

Electronic Theses and Dissertations, 2004-2019

2017

Computer Programming with Early Elementary Students with and without Intellectual Disabilities

Matthew Taylor
University of Central Florida

 Part of the [Special Education and Teaching Commons](#)
Find similar works at: <https://stars.library.ucf.edu/etd>
University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Taylor, Matthew, "Computer Programming with Early Elementary Students with and without Intellectual Disabilities" (2017). *Electronic Theses and Dissertations, 2004-2019*. 5564.
<https://stars.library.ucf.edu/etd/5564>

COMPUTER PROGRAMMING WITH EARLY ELEMENTARY STUDENTS WITH AND
WITHOUT INTELLECTUAL DISABILITIES

by

MATTHEW SCOTT TAYLOR
B.S. Gordon College, 2006
M.Ed. Salem State University, 2010

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the College of Education and Human Performance
at the University of Central Florida
Orlando, Florida

Summer Term
2017

Major Professor: Lisa A. Dieker

© M.S. Taylor 2017

ABSTRACT

Researchers suggest students at the preschool and kindergarten grade levels are active learners and creators and need to be exposed to science, technology, engineering, and mathematics (STEM) curriculum. The need for student understanding in STEM curriculum is well documented, and positive results in robotics, computer programming, and coding are leading researchers and policy makers to introduce new standards in education. The purpose of this single case design study is to research the abilities of kindergarten students, with and without intellectual disabilities (ID), to learn skills in computer programming and coding through explicit instruction, concrete manipulatives, and tangible interfaces. While constructionist methodology is typically used to teach robotics, best practice for students with ID is explicit instruction. For this reason, a group of students with ID and a group of students without ID were taught to program a robot to move in a square, through explicit instruction, and by using the iPad application, Blockly. It was discovered that students in both groups were capable of programming the robot, though students learned at different rates. Introducing STEM to students with and without ID at an early age could prepare students for future STEM careers and encourage students with ID to pursue STEM-related paths.

First and foremost, I dedicate this paper to my Lord and Savior. I would not have been able to complete this program without Him. From the beginning of my journey as a student, as a teacher, and as a doctoral candidate, He has been instrumental in laying the path before me. Thank you God for giving me the mind to think and the will to follow your guidance.

My parents, Jeff and Ellen Taylor, you have never stopped supporting me. Since I was a child, shy and scared to participate in new activities, you both have pushed me to always try, give my absolute best, and never quit. It is with this mindset I have progressed through my life to do what inspires me, and the life I believe I am called to fulfill. I want you both to know how much your love and support mean to me; how much your battles and hardships have moved our family; and how proud I am to stand before anyone and say you are my parents, the reason I am who I am, and that I aspire to be like the both of you someday. I love you both so much.

To my siblings, all *four* of you: Becky my older sister, Ben and Davey my younger brothers, and Andrew the brother I never knew. Our time together has helped shape who each of us is now. Life under the same roof has been amazing at times and difficult at others, but worth every second. You are all wonderful and amazing people, loved by those who know you, and amazing parents to your children. I aspire to be parents as you are, supporting your children with 100% sacrifice and love. And to Andrew, my fourth sibling. You were my older brother, and although you and I never met, I know your life and your passing were pieces of the puzzle guiding me to have a passion for those with disabilities, especially those with Down syndrome. I will be forever grateful to you and how you changed our family and directed my path. I love you all so much.

To my nephews and my niece, you don't know yet how much you inspire me, but I push forward and I push on to make you all proud. My love for all of you cannot be hushed; it spills from me, and everyone who knows me will tell you the same. Always stay focused, do not let life's trials hold you back, but rather use them to spur you on. Remember I am always here for each one of you with loving arms.

To my extended family- Grandfathers and grandmothers, uncles and aunts, cousins, and brother and sisters in-law, thank you for your support and your love. I am blessed to have each and every one of you in my life. I can never say enough about family and the importance of you all. We may not see each other often or talk much, but I know without hesitation I can reach out to any of you, and you will be there.

To my friends new and old, in Massachusetts, Florida, and around the country, I cannot thank you enough for always being there for me and pushing me forward. You all have helped shape the path I have taken, and I owe you all a large debt for that. Those that encouraged me to work with children, those that encouraged me to begin this doctorate program, those that encouraged me to extend so far past my comfort zone- thank you.

And to Riley, you have been one of my inspirations in teaching since the year you were in my kindergarten class. It has been a pleasure to see you grow into the young adult you are today. It has been an honor to know you and your family and be welcomed by them with open arms. You will always be a close friend of mine and one of the driving forces for me in this field.

ACKNOWLEDGMENTS

To my dissertation committee:

Dr. Dieker- Thank you for believing in me, inspiring me, and opening my mind to what the future can be.

Dr. Vasquez- Thank you for pushing me, challenging me, and directing me.

Dr. Hines- Thank you for encouraging me and challenging the constructs of modern education to teach in new and innovative ways to prepare the next generations of students.

Dr. Nickels- Thank you for taking the time with me, and for showing me a new world that can combine computer science and young students of all abilities.

UCF Exceptional Education Faculty- Thank you all for your teaching, guidance, and friendships.

UCF Health and Public Affairs

Drs. Tucker, Pabian, Towson- Thank you for sharing the vision to combine the worlds of special education, physical therapy, and speech and language therapy.

Office of Special Education Programs

Dr. Renee Bradley- Thank you for being my mentor and my friend; you are amazing and inspiring.

Dr. Larry Wexler- Thank you for your guidance and your vision.

Down Syndrome Foundation of Florida- Thank you for the amazing children you support everyday! Thank you for wrapping me in your work.

My UCF Cohort- I love each one of you. I could have never done this without you. You are all special and all near and dear to my heart. I am honored to call you each a friend, and more so, sisters.

And to all my students...

It has been placed on my heart to serve the future of the world, and you are it. Each one of you is special, each one of you is exceptional, and each one of you can do whatever you put your mind to... always give your absolute best and never give up.

TABLE OF CONTENTS

LIST OF FIGURES	xi
LIST OF TABLES	xii
LIST OF ABBREVIATIONS.....	xiii
CHAPTER ONE: INTRODUCTION.....	1
Introduction.....	1
STEM in the Curriculum	2
Computer Programming in Early Education.....	3
Problem.....	4
Purpose.....	5
Research Questions	5
Definitions.....	7
Computer Programming/Coding.....	7
Robotics	8
Concrete Coding	8
Tangible Coding.....	8
Dash	9
Blockly.....	9
Students with intellectual disabilities (SWID).....	9
Students without disabilities (SWOD).....	9
Explicit Instruction.....	10
CHAPTER TWO: LITERATURE REVIEW	11
Introduction.....	11
Students with Intellectual Disabilities Served in STEM and Inclusive Education.....	13
STEM in Early Education.....	15
STEM and Students with Intellectual Disabilities	18
Support for students with intellectual disabilities in STEM curricula.....	20
Computer Science in Early Education	21
Computer Science and Students with Disabilities	33
Studies Pre-2000	33
Studies Post-2000.....	33
Conclusion	38
CHAPTER THREE: METHODOLOGY	41
Introduction.....	41
Problem Statement and Research Questions.....	41
Research Design.....	43
Participants.....	44
Participants with Intellectual Disabilities	44
Participants without Disabilities	45
Setting	47
Instruments.....	48
Physical Manipulatives	48
Tangible coding software.....	48

Dash Robot.....	49
Dependent Measures	50
Frequency checklist	50
KeyMath3 Standardized Assessment.....	51
Documentation of disability.....	51
Independent Variable	51
Procedure	52
Sessions.....	52
Baseline.....	53
Treatment Phase (B)	53
Level 1.....	53
Level 2.....	54
Level 3.....	54
Level 4.....	55
Generalization Phase (C).	56
Inter-observer Agreement	57
Data Analysis	58
Social Validity	58
CHAPTER FOUR: RESULTS	60
Introduction.....	60
Group 1: Students with Intellectual Disabilities	65
Participant 1	65
Participant 2	67
Participant 3	69
Group 2: Students without Disabilities	70
Participant 4	70
Participant 5	72
Participant 6	73
Participant 7	74
Participant 8	76
Participant 9	77
Visual Analysis of Groups 1 and 2	79
Effect size.....	83
Inter-observer Agreement	84
Social Validity	84
Conclusion	85
CHAPTER FIVE: DISCUSSION.....	88
Introduction.....	88
Purpose and Research Questions	89
Research Questions and Discussion for Participants with ID.....	91
Research Question 1	91
Research Question 2	92
Summary Research Question 1 and Research Question 2	93
Research Questions and Discussion for Participants without Disabilities.....	93
Research Question 3	93
Research Question 4	94

Summary Research Question 3 and Research Question 4	95
Social Validity	97
Discussion	97
Reaching all Students.....	101
Implementing Coding/Robotics Activities in Early Elementary Classroom Routines	103
Future Implications	105
Limitations	109
Conclusion	112
APPENDIX A: TREATMENT PHASE B AND GENERALIZATION PHASE SCORE SHEETS AND TREATMENT PHASE B FIDELITY OF IMPLEMENTATION.....	115
APPENDIX B: SOCIAL VALIDITY SURVEYS (PARTICIPANTS, PARENTS, TEACHERS/ADMINISTRATORS)	131
APPENDIX C: INSTITUTIONAL REVIEW BOARD CONSENT.....	135
REFERENCES	137

LIST OF FIGURES

Figure 1. Physical coding manipulatives	48
Figure 2. Blockly tangible software application	49
Figure 3. Dash robot	50
Figure 4. Generalization Phase C introductory tasks.....	63
Figure 5. Participant 1-ID visual data.....	67
Figure 6. Participant 2-ID visual data.....	68
Figure 7. Participant 3-ID visual data.....	70
Figure 8. Participant 4-WOD visual data.....	71
Figure 9. Participant 5-WOD visual data.....	73
Figure 10. Participant 6-WOD visual data.....	74
Figure 11. Participant 7-WOD visual data.....	76
Figure 12. Participant 8-WOD visual data.....	77
Figure 13. Participant 9-WOD visual data.....	78
Figure 14. Participants with ID graphs for visual analysis	80
Figure 15. Participants with ID graphs for visual analysis	81

LIST OF TABLES

Table 1 Evidence-Based Practices for SWID in STEM	21
Table 2 Empirical Studies from 2010-2017 for elementary students and coding.....	27
Table 3 Empirical studies for elementary students with disabilities and coding.....	36
Table 4 Demographics	47
Table 5 Treatment Phase (B) level changes (1-4).....	56
Table 6 Generalization Phase (C)	57
Table 7 Levels in Treatment Phases	65
Table 8 Effect sizes (Tau-U).....	83

LIST OF ABBREVIATIONS

ABC Design: Baseline, Treatment Phase, Generalization Phase
ANOVA: Analysis of variance
ASD: Autism spectrum disorder
CCSS: Common Core State Standards
CRA: Concrete, representational, abstract
DV: Dependent variable
EBP: Evidence based practice
EHA: Education of all Handicapped Children Act
FAPE: Free and appropriate education
ID: Intellectual disability
IDEA: Individuals with Disabilities Education Act
IEP: Individual education program
IQ: Intelligence quotient
IRB: Institutional Review Board
ISTE: International Society for Technology Education
IV: Independent variable
LD: Learning disability
LRE: Least restrictive environment
NCTM: National Council of Teachers of Mathematics
NETP: National Education Technology Plan
NGSS: Next Generation Science Standards
NLTS-2: National Longitudinal Transition Study-2
P#-ID: Participant # with intellectual disabilities
P#-WOD: Participant # without disabilities
STEM: Science, technology, engineering, and mathematics
SWID: Students with intellectual disabilities
SWOD: Students without disabilities
TORTIS: Toddlers Own Recursive Turtle Interpreter System
UDL: Universal design for learning

CHAPTER ONE: INTRODUCTION

Introduction

Widespread consensus in past educational thinking is that subject matter and curriculum in the areas of science, technology, mathematics, and engineering (STEM) are complex educational topics reserved for secondary and post-secondary settings (Faulkner, Crossland, & Stiff, 2013; Goodnough, Pelech, & Stordy, 2014). Researchers today, however, suggest students at the preschool and kindergarten grade levels be active learners and creators being taught STEM curriculum (Lottero-Perdue, Lovelidge, & Bowling, 2010; Moomaw, 2012). Further expanding the range of learners in STEM fields, prominent researchers in the field of special education have also suggested that students with intellectual disabilities (SWID) can access these core subject areas (Browder, Spooner, Ahlgrim-Delzell, Harris, & Wakeman, 2008; Spooner, Knight, Browder, Jimenez, & DiBiase, 2011). Current state government leaders, federal legislators, and researchers recommend embedding STEM content into grade level standards for PreK-12 students (Carr, Bennett IV, & Strobel, 2012; National Council of Teachers of Mathematics, 2000; NGSS Lead States, 2013; U.S. Department of Education, 2016).

The need for early adoption of STEM curriculum is evident due to job shortages in these areas for all populations. The Bureau of Labor estimates five million job openings by the year 2022 in STEM disciplines (Vilorio, 2014). For students with disabilities, the need is critical as less than 4% of people with disabilities hold a job in a STEM-related field (Newman et al., 2011). Introducing and teaching STEM skills in early elementary schools could begin preparing all students for careers driven by STEM (Carr et al., 2012). For SWID, STEM skills could have a dual purpose of supporting their development of adaptive learning and problem solving skills, as well as employment in these areas (Miller, Doughty, & Krockover, 2015).

STEM in the Curriculum

Researchers call for the integration of STEM throughout all grade levels, PreK-12, (Lottero-Perdue et al., 2010; Moomaw, 2012) and standards and curriculum reflect the need to educate all students in these areas (Carr et al., 2012; National Council of Teachers of Mathematics, 2000; National Education Association, 2011; NGSS Lead States, 2013; U.S. Department of Education, 2016). Curriculum and standards in STEM education often are designed with a focus on science and mathematics in the elementary grades and more of a focus on the integration of all four STEM areas as students move through secondary courses (Xie, Fang, & Shauman, 2015). However, Lottero-Perdue, Lovelidge, and Bowling (2010) discuss the need for implementation of STEM standards, specifically engineering, in inclusive elementary classrooms. The authors suggest the five steps of the engineering process (i.e., ask, imagine, plan, create, and improve) provide students with a means to test their own ideas and stretch their learning. A consistent and systematic approach to introducing this engineering process can build students' problem solving skills and understanding of STEM content, by allowing students to be actively involved in learning.

Science, technology, engineering, and mathematics content is driven by hands-on learning experiences, active learning, and engagement (Lee, Lee, & Collins, 2009; Lott, Wallin, Roghaar, & Price, 2013). Positive results in robotics, computer programming, and coding (e.g., Bers, 2010; Kalelioğlu, 2015; Varney, Janoudi, Aslam, & Graham, 2012) are leading researchers to study how to blend all four areas of STEM at the elementary level and introduce new standards in education (U.S. Department of Education, 2016; Whitehouse Office of Science and Technology Policy, 2015).

Computer Programming in Early Education

The commencement of young students being taught computer programming skills coincided with Papert's (1985) LOGO programming language and a constructivist/constructionist framework. Constructivists suggest students build (or construct) their own knowledge by actively interacting with a medium (e.g., robot, programming language) and solving problems to further their knowledge base (Cejka, Rogers, & Portsmore, 2006). Constructionists are similar, but add programming and robotics, allowing children to construct knowledge by inserting themselves in the problem; essentially, they act as the "robot" to understand the code they must then develop to make the robot move (Papert, 1980; Sullivan & Heffernan, 2016). In the 1970s, researchers predicted the impact computers and programming skills would have on the future careers and the knowledge base of students at all grade levels (Papert, 1972, 1980; Perlman, 1974, 1976). During this time period, researchers recognized the difficulty keyboards posed to young students (Papert, 1985; Perlman, 1974). A basic version of tangible coding was introduced, allowing users to enter words or phrases to move a "turtle" (i.e., triangle shape) on a computer screen, but research in this area was limited (Yelland, 1995). Yelland (1995) reviewed studies focused on computer programming and the LOGO coding language developed by Papert. While researchers using LOGO reported positive effects for students' social interactions, mixed results were found in student problem solving skills and cognitive functioning.

The 1990s did not produce much empirical research focused on young students and computer programming, perhaps due to various viewpoints on the use of technology in elementary grades. In the early 21st century, introduction of readily available technology of personal computers and tablets, led researchers to further explore the feasibility of introducing

computer coding to young students. The initial researchers found positive results, but with some limitations (Barker & Ansorge, 2007; Bers, Ponte, Juelich, Viera, & Schenker, 2002; Kalelioğlu, 2015). Several authors, in reviews of the literature completed post-2000, reported constructivist theories may not be best learning frameworks for young students learning computer programming (Lye & Koh, 2014; Sullivan & Heffernan, 2016). The authors noted students learned problem solving skills and cognitive functions when teachers provided explicit instructions, guided student activities, and asked students questions about their work. Prominent researchers in special education found evidence-based practices (EBPs) for SWID accessing STEM curricula (Browder et al., 2008; Spooner et al., 2011), including the importance of using systematic, explicit instruction. Flores, Hinton, Strozier, and Terry (2014) found the use of concrete, representational, abstract (CRA) modeling in mathematics for SWID and students with autism led to learning gains. Using explicit instruction and CRA models may lead populations of students (e.g., young students with ID) to understand STEM curricula.

Problem

While limited research exists in programming and coding conducted with students with learning disabilities (e.g., Atkinson, 1984; Chiang, Thorpe, & Lubke, 1984) and those who are deaf/hard of hearing (e.g., Lange, 1985; Miller, 2009), the skills and learning of students with developmental and severe disabilities (e.g., ID, autism; Lye & Koh, 2014; Yelland, 1995) has only once entered the research paradigm (i.e., Taylor, Vasquez, & Donehower, 2017). The National Education Technology Plan (NETP), implemented by the U.S. Department of Education (2016), provides justification for technology in education, including active learning opportunities and equity in learning for all students. The authors of the New Media Consortium

(2017) discuss the need for STEM, robotics, and programming/coding in K-12 curriculum to prepare students for future careers and learning opportunities. Robotics and programming provide all participants (regardless of disability) interactive, problem-solving activities incorporating all STEM disciplines (Cejka et al., 2006; Sullivan & Heffernan, 2016). While the STEM field continues to grow and the need for personnel continues to increase (Vilorio, 2014), students with disabilities continue to be looked past and are unprepared to enter these fields (Newman et al., 2011).

Purpose

The purpose of this study was to research the abilities of early elementary students, with and without ID, to learn skills in computer programming and coding through explicit instruction (Browder et al., 2008; Devlin, Feldhaus, & Bentrem, 2013), concrete manipulatives, and tangible interfaces (i.e., iPad; Flores, Hinton, Strozier, & Terry, 2014). In this study, students had the opportunity to learn basic coding skills to program a robot to move in a square. Students were taught through explicit instruction and a CRA model over an ABC single subject, changing criterion study design.

Research Questions

The research questions explored in this study were:

1. To what extent does coding ability of early elementary SWID increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by a rubric?

Null Hypothesis: Coding ability of early elementary SWID will not increase through explicit instruction, using physical manipulatives in a one-on-one setting.

2. To what extent does coding ability of early elementary SWID generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a rubric?

Null hypothesis: Coding ability of early elementary SWID will not increase through explicit instruction, using only an iPad application in a one-on-one setting.

3. To what extent does coding ability of early elementary students *without* disabilities (SWOD) increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by rubric?

Null Hypothesis: Coding ability of early elementary SWOD will not increase through explicit instruction, using physical manipulatives in a one-on-one setting.

4. To what extent does coding ability of early elementary SWOD generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a frequency chart?

Null hypothesis: Coding ability of early elementary SWOD will not increase through explicit instruction, using an iPad application in a one-on-one setting.

5. What are the perceptions of students, parents, and school administration regarding the goals, procedures, and outcomes of the explicit instruction on computer programming as measured by unstructured interviews and surveys?

The research design was based on an ABC changing criterion across participants design. In this design, participants began five baseline sessions (A) followed by the treatment phase (B). Treatment Phase B consisted of four levels to explicitly teach participants to use physical

manipulatives to program a robot to travel in a square. Each level of the phase was contingent on the preceding level, and students were required to complete a minimum of three sessions at each level (What Works Clearinghouse, 2011). All participants had the opportunity to independently code the robot to move in a square after each session. Following the treatment phase, participants entered Generalization Phase C. In this phase, participants learned the basic coding pieces and application procedures to program a robot to travel in a square, using tangible coding software, such as Blockly on an iPad. The participants demonstrated how to code the robot to move in a straight line following explicit instruction, and were given the chance to independently code the robot to move in a square, using the iPad application. A sequencing effect was expected due to the nature of coding programs (in treatment and generalization phases) used. The coding blocks used in the intervention phase (B) were physical replications of the coding software used in the Generalization Phase C. This research consisted of two studies following the same research procedures (Baseline, Treatment Phase B, and Generalization Phase C): a) Study 1 with SWID, and b) Study 2 with SWOD.

Definitions

Computer Programming/Coding

Computer programming/coding traditionally reflects writing lines of code using a software program (e.g., C++, JavaScript) and a computer-processing machine. Technological advances in the 2000s led to accessible coding applications (e.g., Blockly, Wonder, Code.org, Scratch) aimed to introduce programming skills to young students. These applications remove long strings of code manually created by the user, and introduces “chunks” of code through pictures and limited text (e.g., Cherp, Lego Mindstorms; Bers, 2010).

Robotics

Robots are an engaging medium for students, providing a physical representation to enact students' programming sentences (Berry, Remy, & Rogers, 2016). Programming with robots is included in robotics kits like Lego Mindstorms (Karp, Gale, Lowe, Medina, & Beutlich, 2010) and Wonder Workshop's Dash and Dot (used in this study). Researchers found a higher understanding of programming languages and purposes amongst students interacting with both robotics and coding software (Sullivan & Heffernan, 2016).

Concrete Coding

Concrete coding is using physical manipulatives as lines of code to teach programming through pictures and limited text. Researchers introduced this type of coding so that young students (PreK-2nd) without reading abilities could access programming languages (Bers, 2010). Physical manipulatives allow students to construct a concrete image of linking code to tell a robot instructions to follow.

Tangible Coding

Tangible coding refers to the process of interacting with software applications to create code for a robot to follow. Physical manipulatives (aside from the actual robot) are removed in this style of coding, and students use applications to click and drag chunks of code, which are replications of the physical manipulatives, to give directions to a robot. Examples include the tablet application, Blockly (used in this study), and the interactive site, Code.org (Bers, 2010; Kalelioğlu, 2015).

Dash

Dash is a robot created by Wonder Workshop (see Figure 3). The robot was designed to engage students through voice, sound, and accessories (e.g., attachments for Legos, smartphone mount). Using software applications (e.g., Blockly, Wonder, Go), students can control the robot through touch-screen devices (e.g., tablets or smartphones).

Blockly

The application “Blockly” (see Figure 2) used in this study was developed by Wonder Workshop specifically for the Dash robot (see description). This application allows students to merge blocks of code to tell the robot to follow a variety of instructions (e.g., movement, sound, repeat). Each block of code represents JavaScript, but can be moved using one finger on an iPad or other touchscreen devices.

Students with intellectual disabilities (SWID)

The formal definition of a SWID is a diagnoses of delayed cognition, IQ scores under 70, and significant intellectual and adaptive learning deficits (American Psychiatric Association, 2013). Examples of intellectual learning include “problem reasoning, problem solving, planning, abstract thinking, judgment, academic learning, and learning from experience... adaptive deficits limit functioning in daily life, such as communication, social participation, and independent living” (American Psychiatric Association, 2013, p. 33).

Students without disabilities (SWOD)

In this study, a SWOD was any student without a documented disability being served on an IEP or 504 plan in the school they currently attended.

Explicit Instruction

Explicit instruction is direct instruction with scaffolding, student practice, and consistent feedback from the instructor (Doabler & Fien, 2013). Explicit instruction is considered an evidence-based practice for students with and without disabilities, especially those with ID (Browder et al., 2008). Explicit instruction in this study was used in a 1:1 setting to directly teach a participant a coding language through physical manipulatives and tangible coding applications.

CHAPTER TWO: LITERATURE REVIEW

Introduction

The need for personnel in the fields of science, technology, engineering, and mathematics (STEM) is well documented (Vilorio, 2014). Clear evidence exists for both the need for and the emphasis on preparing students for STEM careers at a younger age, with officials in the Whitehouse Office of Science and Technology Policy (2015) budgeting over \$3 billion to prepare students in these areas. The Bureau of Labor Statistics estimates over 5.5 million job openings in STEM-related fields by 2022 (Vilorio, 2014), making STEM curricula one of the most prominent educational foci of the 21st century (U.S. Department of Education, 2016). The importance of job creation through more effective STEM education is referenced in numerous, federal legislative actions and state curriculum standards (e.g., Carr, Bennett IV, & Strobel, 2012; National Council of Teachers of Mathematics, 2000; NGSS Lead States, 2013).

Prominent researchers suggest STEM curricula should begin at a young age, a learning process developed and expanded as students progress through elementary and secondary school (Bers, 2010; Varney, Janoudi, Aslam, & Graham, 2012). Students in early elementary school are active learners and creators (Lottero-Perdue et al., 2010) and should be prepared as young as possible to intellectually understand and develop skills in STEM areas (Bers, Flannery, Kazakoff, & Sullivan, 2014; Lott et al., 2013). This same emphasis on and need for preparation for STEM careers also exists for students with disabilities starting at a young age. The National Longitudinal Transition Study-2 (NLTS-2; Newman, 2011) documented that less than 4 % of students with disabilities held jobs in STEM-related careers of computer, mathematical, architecture, engineering, and science (Newman et al., 2011). Moreover, only 1.8% of adults

with developmental disabilities (i.e., autism, intellectual disabilities [ID]) had employment in STEM-related fields.

Preparing people with disabilities for STEM careers begins in the classroom at an elementary age (Varney et al., 2012). All students, regardless of learning ability, must be given the opportunity to learn core educational material and skills (e.g., STEM curricula; Every Student Succeeds Act [ESSA], 2015; Individuals with Disabilities Education Act [IDEA] 2004). Inclusion of students with disabilities in STEM education is accomplished through constructs like Universal Design for Learning (UDL), which highlight students' abilities and learning processes (Basham & Marino, 2013; CAST, 2011). Student-friendly computer programming software such as Cherp or Blockly (Bers, 2010; Sullivan & Heffernan, 2016) allows young students to problem solve, create code, and learn basic STEM skills through an accessible medium (Devlin et al., 2013; Lottero-Perdue et al., 2010).

Intellectual skills, like self-determination and problem solving, are an area of difficulty for students with ID (Cote et al., 2010) and need further research (Agran & Hughes, 2005; Miller et al., 2015). Students with ID often struggle to determine the best possible solution to a given problem and determine their answers based on the easiest and most familiar course of action (Cote et al., 2010; Goharpey, Crewther, & Crewther, 2013; Scruggs & Mastropieri, 1997). Students with and without disabilities can begin learning problem solving skills related to STEM curriculum and careers through basic coding languages, using physical manipulatives and tangible coding software (e.g., tablets; Bers et al., 2014).

With readily available technology, programming languages, software, and computers, programming (e.g., coding) could be introduced as a core component of instruction for young students (Lye & Koh, 2014; Papert, 1985), despite ability or disability (Newman et al., 2011;

U.S. Department of Education, 2016; Vilorio, 2014). Robots are an engaging medium to give students the ability to see the constructs of their written code in action (Berry et al., 2016). Berry, Remy, and Rogers (2016) describe robots as “an ideal artifact for teaching real-world application of math, science, programming, and engineering” (p.43). Robots come in different forms and styles for student use, including kits for students to build their own (e.g., Lego Mindstorms; Karp, Gale, Lowe, Medina, & Beutlich, 2010; Sullivan & Heffernan, 2016) and pre-assembled robots (e.g., Wonder Workshop’s Dash and Dot). These types of basic tools could be and are being used in Pre-K and early elementary settings (Sullivan & Heffernan, 2016) for students with and without disabilities.

Students with Intellectual Disabilities Served in STEM and Inclusive Education

Students with ID have been at a disadvantage in the education realm throughout much of history, especially compared to their general education peers (Polloway, Patton, & Marvalin, 2011). The formal definition of a student with an ID is a diagnoses of delayed cognition, IQ scores under 70, and significant adaptive learning deficits (American Psychiatric Association, 2013). Yet, this population of students, when included with peers, demonstrates proven successes in education (Kemp & Carter, 2006; Sermier Dessemontet & Bless, 2013). Researchers call for interventions and studies related to students’ intellectual skills, like self-determination and problem solving, as this is a vital skill used to access inclusive classrooms, careers, and social engagements (Cote et al., 2010; Miller et al., 2015).

The shift where students are served and what they are taught is an emerging aspect of culture and education in U.S. Society. In the early 1850s, schools and institutions focused on providing basic skills to students with ID simply for community living or, in many cases, to

remain peacefully in isolation in institutions (Potter, 1853). In 1932, Aldrich argued SWID may learn slower than peers, but learning and acquisition of new skills was possible. She stated that SWID need extra time to learn and complete tasks and a structured work environment to reinforce positive behaviors. Similar findings and suggestions for SWID built upon Aldrich's work (Brown et al., 1979; Spooner & Brown, 2011).

Dunn, in 1968, recommended placements, assessments, and instruction for students with disabilities align more with the expectations for all students. Dunn argued for proper assessments to identify different disabilities, not a one size fits all diagnosis based upon a label. Further, he proposed proper student placement is based on assessment results aligned with strengths and deficits. These suggested changes in the approach to education for students with disabilities by Dunn and others in the field (e.g., Reynolds, 1962) provided the framework and foundation for mainstream education and placement of students in the least restrictive environment (LRE; IDEA, 1990).

In 1971, *Wyatt v. Stickney* continued the emphasis of looking to educate students with disabilities like their peers. The case established the right for an adequate education for those institutionalized due to mental disabilities, so they had an opportunity to become effective members of society. *Wyatt v. Stickney* established standards still in effect presently, which guarantees rights of treatment and care for those with disabilities (IDEA, 1990; 2004). A year later, *Pennsylvania Association for Retarded Children v. Commonwealth of Pennsylvania* (1972) led to a ruling guaranteeing special education services to SWID, as well as free and appropriate education. Similar to the civil rights case *Brown v. Board of Education* (1954), this case acted as a springboard to incorporate all students with disabilities under its umbrella. Shortly thereafter, *Mills v. Board of Education of District of Columbia* (1972) judicial court determined special

education services were a right to all students with disabilities. In 1973, Section 504 of the Rehabilitation Act became the first civil rights law to protect the rights of persons with disabilities, stating discrimination against those with disabilities was prohibited. These four court cases and one act provided a substantial foundation for the creation of the Education For All Handicapped Children Act (EHA; 1975). This act was the first of its kind and has been reauthorized several times, namely as the Individuals with Disabilities Education Act of 1990, 1997, and 2004. These acts uphold laws providing all students with disabilities equal access to education, a definition of special education, and requirements for school personnel to evaluate students with disabilities annually.

The importance of educating students with disabilities by the same standards as general education students is highlighted by Yudin and Musgrove's (2015) Dear Colleague Letter "Guidance on FAPE." Goals of students' individualized education programs (IEPs) must align with state standards and "include specially designed instruction necessary to address the unique needs of the child that result from the child's disability and ensure access of the child to the general education curriculum..." (p. 7). Students with disabilities, including ID, should be educated alongside their general education peers in state curriculum, including new initiatives in STEM at the early elementary level (Basham, Israel, & Maynard, 2010; Lottero-Perdue et al., 2010; Moomaw, 2012; Yudin & Musgrove, 2015).

STEM in Early Education

The integration and union of STEM academic areas provides a process for problem solving (Sullivan & Heffernan, 2016). Mitts (2016) explains the components of STEM education allow students to individually answer questions and collectively solve whole-

problems. He suggests science asks why, technology answers how, engineering figures out what, and mathematics describes relationships. Together, the four STEM disciplines provide a well-rounded focus on real-world problem solving (National Media Consortium, 2017).

Xie, Fang, and Shauman (2015) suggest STEM education is defined by grade level. Elementary grades (K-6) focus mainly on the integration of science and mathematics to actively engage students in curricula. As students get older, more choices are available for STEM, such as varying levels of mathematics, sciences, computer science, and social sciences (NCTM, 2000; NGSS, 2013).

Science, technology, engineering, and mathematics are interlocking subject areas (Basham et al., 2010) found intertwined in state standards and national accreditations. The Common Core State Standards (CCSS; National Governors Association Center for Best Practices & Council of Chief State School Officers, 2010) and the National Council of Mathematics Teachers (NCTM; 2000) describe five, core mathematical domains all students in elementary school should have extensive experiences in during their first years of education: a) counting and cardinality, b) operations and algebraic thinking, c) number and operations in base ten, d) measurement and data, and e) geometry. These five areas provide the foundation for all mathematics skills taught between grades one to twelve. The Next Generation Science Standards (NGSS; 2013) embed science and engineering instruction into general curriculum to provide a well-rounded learning space for students. Engineering concepts can be found in most states' curriculum standards, related to science and mathematics (Carr et al., 2012). The International Society for Technology in Education (ISTE; 2007, 2016) introduce technology standards focused on students' creativity, innovation, critical thinking, and problem solving abilities. The National Media Consortium (2017) introduce long, mid, and short-term needs in education

technology and STEM learning, as well as significant challenges and necessary developments in school systems related to STEM learning.

Preparation for STEM content, engagement, and career readiness should begin in students' early elementary years (Bers, 2010; Kalelioğlu, 2015; Varney et al., 2012). Science, technology, engineering, and mathematics content is driven by hands-on learning experiences, active learning, and engagement (Lee et al., 2009; Lott et al., 2013). Lottero-Perdue, Lovelidge, and Bowling (2010) discussed the need for implementation of STEM standards, specifically engineering, in inclusive elementary classrooms. The authors suggest the five steps of the engineering process (i.e., ask, imagine, plan, create, and improve) provide students with a means to test their own ideas and stretch their learning. A consistent and systematic approach to introducing this process can build students' problem solving skills and understanding of engineering. Even the concept of failing is encouraged by the authors, as it gives students the opportunity to test their work and spurs thinking of what needs to be done differently to make a project succeed.

Nadelson and colleagues (2013) reason STEM education is important at the elementary level as it is engaging, explorative, and promotes student enthusiasm. The authors continue their reasoning to suggest engineering is the glue that holds all of the STEM components together, and teachers need to implement curriculum around this subject. Thirty-three teachers participated in Nadelson and colleagues' two-year study, which focused on teachers instructing STEM-related curriculum. The researchers found that a knowledge regarding STEM correlates with the teachers' ability and confidence to teach the subjects. The researchers found most teachers were not knowledgeable of STEM concepts, which directly affected their teaching ability and confidence. The authors contend for professional development and pre-service teacher

instruction to focus on the STEM areas. Implementation of STEM skills, especially engineering, needs further validation for teacher fidelity and student learning.

Engineering is driven by science and mathematics, and is often made accessible to students through technology (Pantoya, Aguirre-Munoz, & Hunt, 2015). DiFrancesca, Lee, and McIntyre (2014) researched the incorporation of engineering into the curriculum for elementary students. The authors suggest students can practice and be knowledgeable of engineering by incorporating the design process (described above) into problem solving during science, mathematics, and literacy. Likewise, Devlin, Feldhaus, and Bentrem (2013) document the need to incorporate technology in all areas of the curriculum. The authors used a mix-method research approach to measure students' attention to, understanding of, and engagement with a STEM assignment, delivered by either an in-person teacher (control group) or a video-based teacher (treatment group). Devlin, Feldhaus, and Bentram found students engaged more with the video-based teacher than the control group with the in-person teacher, as measured by their attention to instructions and completion of the assignment. The authors suggest engaging students early in a lesson is key to keeping them focused for the rest of the class. They emphasize, as part of their findings, students be given the medium they will use constantly and are brought up using (i.e., technology).

STEM and Students with Intellectual Disabilities

The National Education Technology Plan (NETP; U.S. Department of Education, 2016), outlines the need and purpose of technology in education, including active learning and participation and equity in learning. The five main sections of the plan include: a) Learning, b) Teaching, c) Leadership, d) Assessment, and e) Infrastructure. The plan specifically describes

provision of technology accessibility for all students, including those with disabilities. Teachers are tasked with modifying lessons and assessments, using technology to close the achievement gap and equity in education areas (e.g., exams, essays, curriculum). Many subject areas are interwoven using technology, including science, engineering, and mathematics.

Although STEM curriculum is becoming more available for students in early elementary grades, students with disabilities are not often included or are unable to access the information in the format provided (Lye & Koh, 2014). The National Education Technology Plan (NETP; U.S. Department of Education, 2016) outlines the needs, benefits, and next steps for technology in America's school systems. Equity of learning curriculum and access to technologies is specifically called to include students with disabilities (amongst other groups).

In the State of the Union Address (2016), President Obama reiterated the need for students to be college and career ready in specific areas, especially those reflecting STEM (i.e., computer science, mathematics). The passing of ESSA (2015) brings the integration of STEM into the general curriculum. The Every Student Succeeds Act also calls for all students to be in the most appropriate environment for their education, beginning with inclusive practice in general curriculum classes. Thus, students of all abilities are to be taught STEM curriculum and need to show success in their learning. Both ESSA and IDEA (2004) require teachers to use evidence-based practices (EBPs) to help students learn academic skills.

Researchers argue critical components of STEM (i.e., mathematics) are the foundation and early predictors for skills in other academic disciplines, such as literacy (Clements & Sarama, 2008). Teachers need to be prepared to teach all students in these critical areas and adjust their teaching strategies based on students' needs (Wakeman, Karvonen, & Ahumada, 2013). Many teachers find teaching STEM at the elementary level a daunting task (Goodnough

et al., 2014), and have doubts about teaching these concepts to SWID. Teachers also have limited examples to base teaching SWID STEM curriculum, using standards from the CCSS, NCTM, and NGSS (Browder, Treal, et al., 2012). Researchers suggest teachers use EBPs to help students learn and succeed in STEM curriculum (Browder et al., 2012; Devlin et al., 2013; Spooner et al., 2011). Researchers also continue to stress the importance of teaching students with ID problem solving skills, which is an area all four STEM disciplines cover (Hefty, 2015; Miller et al., 2015; Scruggs & Mastropieri, 1997).

Support for students with intellectual disabilities in STEM curricula

Prominent researchers in the field studied EBPs to aide teachers in helping SWID access STEM curriculum (see Table 1 for summary). Browder, Spooner, Ahlgrim-Delzell, Harris, and Wakeman (2008) and Spooner, Knight, Browder, Jimenez, and DiBiase (2011) note EBPs for students with significant disabilities (e.g., ID) in science and mathematics education. The researchers list EBPs for students with ID in mathematics and science, including systematic, explicit instruction, life skills in context (known as *in vivo*), and opportunities to respond. Doabler and Fien (2013) note explicit instruction (i.e., direct instruction with scaffolding, student practice, and consistent feedback) is beneficial for students with disabilities and those struggling with mathematic concepts.

Concrete, representational, and abstract (CRA) models present curriculum and activities in an obtainable manner (Agrawal & Morin, 2016). Agrawal and Morin (2016) describe CRA as a process in which the teacher “guides the student through a mathematical concept... through the use of manipulatives and visual representations that illustrate the concept...” (p. 35). The CRA model is most effective when taught with explicit instruction and provides scaffolding for the student to learn abstract ideas and lessons. Flores, Hinton, Strozier, and Terry (2014) identify

CRA mathematics instruction techniques with 11 students with disabilities, including ID and autism spectrum disorder (ASD). All students were assessed with a pretest, and no significant differences in mathematics abilities were found. Teachers instructed students in over twenty lessons, introducing addition and subtraction concepts through CRA and strategic instruction models. The authors found significant gain scores between CRA levels from pretest to posttest. The concrete-representational-abstract teaching model is most commonly used with students with mathematics' disabilities, not those with ID and ASD. Flores and colleagues suggest CRA might be a viable learning strategy for SWID and others.

Table 1

Evidence-Based Practices for SWID in STEM

Evidence-Based Practice	Research Article
Explicit instruction (supported through scaffolding, student practice, consistent feedback)	Browder et al., 2008; Doabler & Fien, 2013
Life skills in context (in vivo)	Browder et al., 2008
Opportunities to respond	Browder et al., 2008
Time-delay	Spooner et al., 2011
Task analytic instruction	Spooner et al., 2011
Place-based learning	Spooner et al., 2011
Concrete, Representational, Abstract (CRA) models	Agrawal & Morin, 2016; Flores et al., 2014

Computer Science in Early Education

Computer science and computer programming are skill areas of STEM curriculum that drive many careers (Kalelioğlu, 2015). Preparation for these careers needs to begin early in

students' schooling, which is a time to promote engaging activities and develop necessary skills (Bers et al., 2014; Fessakis, Gouli, & Mavroudi, 2013; Kalelioğlu, 2015; Kazakoff & Bers, 2012). These types of activities and experimentation begin by teaching topics related to STEM (Lott et al., 2013; Monari Martinez & Benedetti, 2011). Teachers and instructors can implement programming and coding activities in early elementary school through physical manipulatives and tangible coding software. Physical manipulatives are blocks representing basic code (e.g., Cherp; Bers, 2010). Tangible coding software allows students to interact and build code on a computer device or tablet (e.g., Blockly; Kalelioğlu, 2015; Lye & Koh, 2014). Currently, all articles related to programming and early elementary students are represented through a constructivist/constructionist approach (see description below; see Table 2).

Perlman (1974, 1976) suggests computers are an invaluable experience for children. Recognizing the difficulty a keyboard system poses to young students, a physical coding system called Toddler's Own Recursive Turtle Interpreter System (TORTIS) was developed. Students learned to program a robot-like device (called a "turtle") to follow specific commands using tangible manipulatives. Perlman found young students required teacher suggestions for problems to solve using the turtle (e.g., following a specific path, creating a shape) and instruction on how to begin or further their created programs.

Papert (1972, 1980) researched teaching computer programming to students in elementary school, foreseeing computers as an important advancement in technology as early as the 1960s. In 1972, Papert described technology and computers in the education system as "...something the child himself will learn to manipulate, to extend, to apply to projects, thereby gaining a greater and more articulate mastery of the world, a sense of the power of applied knowledge and a self-confidently realistic image of himself as an intellectual agent"

(p. 245). He continued by describing computers as important machines to active learning and problem solving when taught to students using comprehensible language. “We can give children unprecedented power to invent and carry out exciting projects by providing them with access to computers, with a suitably clear and intelligible programming language and with peripheral devices capable of producing on-line real-time action” (p. 245).

Papert developed the programming language, LOGO, which required the user to enter commands to navigate a pointer (called a “turtle”) on the computer screen. Papert studied under Piaget and the theory of constructivism, in which students experimented in their learning by constructing their knowledge (Bass, 1985). Papert’s vision of computer programming was set in the belief that children had to teach themselves by experimenting with programming, failing, and then adjusting their thinking (Papert, 1985). Papert moved away from Piaget’s theory and coined the term constructionism, which has a similar approach to constructivism, but the participant constructs his or her own learning by using concrete representations of the robot and computer, inserting themselves into the task, and then figures out the programming problem (Bass, 1985; Sullivan & Heffernan, 2016).

In a review of the literature regarding LOGO, Yelland (1995) found positive effects of programming language on students’ problem solving skills, social interactions, and cognitive skills. Yelland identified in past studies results related to social skills, with mixed-results in attainment of problem solving skills and cognitive functioning. In later studies, researchers found students learned programming processes and language to a significantly greater extent when teachers taught skills in a structured and explicit manner (Bers et al., 2014; Kärnä-Lin, Pihlainen-Bednarik, Sutinen, & Virnes, 2006; Lye & Koh, 2014; Ratcliff & Anderson, 2011).

Lye and Koh (2014) reviewed 27 intervention studies focused on teaching computer programming and robotics to K-12 students. The authors found minimal studies after Papert's introduction of LOGO in the 1970s-1980s until the early 2000s. Introduction of easy-to-use programming languages and software renewed interest in teaching students coding skills. Lye and Koh suggest visual programming mimics spoken English and can reduce the need for complicated computer code, to decrease the cognitive load on students. The authors note most researchers' approaches allowed students to actively engage with programming and were supported in their learning.

Lye and Koh (2014) note students need adult guidance for experience and understanding to take place. Lye and Koh found most (if not all) researchers assume students can learn programming from "self-discovery" (p. 58). In some cases, self-discovery worked, but students needed instructor input throughout. For instance, Barker and Ansorge (2007) researched differences on a robotics pre/posttest in 9-11 year olds ($N = 32$), between a treatment group ($n = 14$) receiving robotics instruction in an after school program, and a control group ($n = 18$) without access to robotics instruction or robots. Using Papert's findings on experiential learning (i.e., active learning, learn through experience), the researchers provided students with procedural knowledge and gave students the responsibility to transfer their knowledge to new situations. The authors found students in the treatment group scored significantly higher ($p < .001$) on robotics posttests than those in the control group. Lye and Koh (2014) found students needed the opportunity to reflect on their learning, ask questions, and think about what they were doing as they programmed, rather than simply constructing their own knowledge.

Kalelioğlu and Gülbahar (2014) supported Lye and Koh's (2014) findings using a tangible programming application (i.e., Scratch) with fifth grade participants ($N = 49$). Students

were assessed with a problem solving inventory and focus group interviews. The researchers found no significant gains in problem solving abilities for students learning programming, and the majority of students found difficulty in programming successfully without intervention. Students performed best when they followed teachers' explicit instructions and were supported throughout the coding interventions.

Similarly, Sullivan and Heffernan (2016) conducted a literature review of research studies involving robotics construction kits. The authors used only qualitative and mixed-methods studies in their review to find common themes amongst researchers from 1999-2014. Sullivan and Heffernan identified 21 articles meeting their keywords (e.g., qualitative or mixed method empirical studies, robotics, education, PreK-12) and their designation as either strong or fair use of research methods (scored by a rubric to score trustworthiness of data). Four themes were found, including the use of: a) robotics to learn directly about robots (first-order uses) and robotics to learn concepts about computer programming (second-order uses); b) learning curricula material through active participation with robotics; c) computational thinking and learning in programming; and d) trial and error procedures to learn problem-solving skills. The authors found students learning robotics supported their abilities to learn computer programming and engineering processes.

Table 2 (and Table 3) depicts studies completed from 2010-2017, related to early elementary education (PreK-5) and computer programming. Databases searched were ERIC, PsychINFO, Applied Science & Technology Source, Education Source, Science Citation Index, Academic OneFile, Education Full Text (H.W. Wilson), and ScienceDirect. Studies were also found with a combination of search terms including: elementary, disabilities, students,

kindergarten, computer programming, robotics, and coding. An initial search found 34 studies. Of these results, 18 studies reported empirical data (quantitative, qualitative, or mixed methods).

Table 2

Empirical Studies from 2010-2017 for elementary students and coding

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Sullivan & Bers, 2016	PreK-2 nd grade students. 8-week robotics curriculum	PreK-2 nd	<i>N</i> = 60 PreK <i>n</i> = 15 Kindergarteners <i>n</i> = 18 First graders <i>n</i> = 16 Second graders <i>n</i> = 11	One Group Posttest Design	IV: Robotics curriculum (8 weeks) DV: Assessment Posttests: • Robot Parts task • Solve-It	Kruskal-Wallis H-Test	<ul style="list-style-type: none"> - Robot Parts Assessment: No significant differences between grade levels - Solve-It Assessment: Kindergarten, first, and second graders did significantly better than PreK students on Hard Sequencing (5 instructions) tasks - PreK-2nd graders were able to master programming skills (e.g., sequencing, loops) - Older students (1st and 2nd graders) progressed through curriculum faster than younger students (PreK and kindergarten)
Bartolini Bussi & Baccaglioni-Frank, 2015	First grade students in northern Italy over 4 month time period	First Grade	<i>N</i> = 18	Mixed Methods One Group Posttest Design	IV: Robotics curriculum DV: Posttest tasks given 4 months after IV	Researcher/Teacher interpretation of tasks	<ul style="list-style-type: none"> - Children were able to create rectangles by programming robot (Bee-Bot). - Learned to identify similarities and differences between rectangles and squares

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Kalelioğlu, 2015	Fourth grade students in Turkey	Fourth Grade; 10-years-old	<i>N</i> = 32	Triangulation Design (Mixed Methods)	IV: Programming through code.org DV: Reflective Problem Solving Skills Focus-group interviews	<i>t</i> -Test Interview themes	<ul style="list-style-type: none"> - Were no significant differences between pre/posttest on reflection problems solving skills, although scores did increase (greater increase for females) - Qualitatively, all students felt code.org was beneficial, increase their programming knowledge, or led to increased problem solving ability
Strawhacker & Bers, 2015	Kindergarten students in nine week program	Kindergarten	<i>N</i> = 35 Tangible: <i>n</i> = 14 Graphical: <i>n</i> = 7 Hybrid: <i>n</i> = 14	Mixed Methods	IV: Robotics curriculum using tangible, graphical, or hybrid model DV: Solve-It Assessment (at midpoint and end of study)	One-Way ANOVA, repeated measures test	<ul style="list-style-type: none"> - No significant scores, suggesting all intervention styles had the same effect on students' programming comprehension - Tangible coding groups scored significantly better from mid test to posttest once graphical interface was introduced
Bers, Flannery, Kazakoff, & Sullivan, 2014	Kindergarten students	Kindergarten	<i>N</i> = 53 Students	Within group, Quasi-Experimental	IV: TangibleK Robotics Program DV: Likert Scale scoring to measure students' sequencing, correspondence, debugging, and control flow	Repeated Measures ANOVA	<ul style="list-style-type: none"> - TangibleK curriculum was engaging and appropriate for Kindergarten students - Students demonstrated higher levels of understanding on their final project - On lessons 3-6 of increasing difficulty, achievement scores dropped, indicating perhaps not all basic skills had been developed

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Harlow & Leak, 2014	Third grade students	Third grade	<i>N</i> = 20	Qualitative	IV: Students use of Turtle Art curriculum DV: video records	Video recordings transcribed and coded	<ul style="list-style-type: none"> - Proficiency in programming an image came from teacher-directed conversation - Proficiency in developing novel outputs came from peer-to-peer conversation
Kalelioğlu & Gülbahar, 2014	Fifth grade students in private school in Turkey	Fifth Grade	<i>N</i> = 49	Sequential Mixed Methods (Pretest/posttest, observation, focus group interview)	IV: Scratch programming DV: problem solving skills (Problem solving inventory)	<p>Paired samples <i>t</i>-test</p> <p>Interviews coded and found themes</p> <p>Observations were summarized, least used items found for each participant</p>	<ul style="list-style-type: none"> - No significant increases on problem solving inventory, student' self-confidence in problem solving was low, but improved between pre- and posttest. - Students need teacher direction and support in problem solving tasks and achieving higher-order thinking - Students indicated they enjoyed programming and wanted to increase their skills - Just providing the software application and programming is not enough for students, need effective teacher guidance

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Adams & Cook, 2013	One student with complex communication needs (CCN) and cerebral palsy	12-year-old	<i>N</i> = 1	Descriptive case study (with quantitative and qualitative results)	IV: Lego Robot programming activities in one-on-one setting DV: Assessments <ul style="list-style-type: none"> • Goal attainment scaling (GAS; tracked participation of student using a speech generating device) • Morae Usability Analysis Software 	GAS scores determined after each session by author and corroborated by assistive technology team. Observations made by author Effectiveness and efficiency measured using Morae Usability Analysis Software	<ul style="list-style-type: none"> - Participant needed researcher direction for most programming activities - Participant interacted with classmates in a positive manner, but often did not initiate interactions - Adapting the speech generating device to control robot allowed student to actively participate in the class rather than simply observe
Sullivan & Bers, 2013	Kindergarten students	Kindergarten	<i>N</i> = 53 Males <i>n</i> = 28 Females <i>n</i> = 25	Two factor posttest design	IV: TangibleK Robotics Program DV: Tasks completed during each lesson (six lessons) and ability to debug throughout lessons	Independent t-tests Pearson Product-Moment Correlations	<ul style="list-style-type: none"> - TangibleK Robotics Program equally accessible to kindergarten males and females - Males only significantly outperformed females in two areas: attaching robotic parts and selecting appropriate instructions when programming - Males and females were both equally successful in completing final project, indicating equal ability to use knowledge from all lessons

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Fessakis, Gouli, & Mavroudi, 2013	Kindergarten students in whole class setting	5-6 years-old	<i>N</i> = 10	Case study	IV: Logo style programming language on interactive whiteboard in whole class setting DV: Completion or non-completion of coding tasks	Frequency chart (completed or not completed) for each lesson Observations	<ul style="list-style-type: none"> - Students were able to develop basic programs and engage in problem-solving - Students were engaged in whole-class setting and were able to develop programming skills with teacher guidance
Sullivan, Kazakoff, & Bers, 2013	PreK classrooms and three teachers	PreK	Students <i>N</i> = 37 Teachers <i>N</i> = 3	Qualitative	Programming language “CHERP”	Observations Interviews (students and teachers) Post survey (students and teachers)	<ul style="list-style-type: none"> - Young students can program robots to complete specific tasks - Students supported through scaffolding (teacher guidance and direction) - Students at this age need supports in place as they had difficulty expanding on engineering processes
Kazakoff & Bers, 2012	Kindergarten students	Kindergarten	<i>N</i> = 54 Private school - Treatment <i>n</i> = 11 - Control <i>n</i> = 11 Public school - Treatment <i>n</i> = 15 - Control <i>n</i> = 17	Two Factor Control/Treatment Group Pretest/Posttest	IV: TangibleK Robotics Program DV: Sequencing task	Between Groups ANOVA	<ul style="list-style-type: none"> - Students in intervention groups scored significantly higher than students in control groups on sequencing tasks - Programming instruction may be better taught in small groups

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Kwon, Kim, Shim, & Lee, 2012	First grade students	First grade	<i>N</i> = 24 Treatment <i>n</i> = 10 Control <i>n</i> = 14	Control/ Treatment Group Pretest/Posttest Design	IV: Treatment Group: A-Bricks programming manipulatives and instruction Control Group: Scratch Programming DV: Pre-survey, usability questionnaire, error frequency count, robot programming task	Independent <i>t</i> - tests, achievement comparisons	<ul style="list-style-type: none"> - No statistical difference in scores found between usability of A-Bricks and Scratch programming - Treatment group made half as many errors as control group (72, 150 respectively), suggesting A-Bricks programming has a higher usability than Scratch - Positive effect found A-Bricks manipulatives in creating programs - As tasks grew in difficulty, both groups found difficulty in coding
Louca, Zacharia, & Constantinou, 2011	4 th -5 th grade students	11- to 12- year-olds	<i>N</i> = 38	Qualitative: Case Study	Stagecast Creator (programming software designed for young students to manipulate micro-worlds using graphic/symbol program language)	Triangulation: transcription of whole group discussions, artifact analysis, teacher involvement	<ul style="list-style-type: none"> - Found three modeling frames student coding shifted between: phenomenological description (describe story/problem), operationalization story of system (translating story to programming code), constructing algorithms/evaluations (assessing their code)

Computer Science and Students with Disabilities

Studies Pre-2000

Computer science and programming have been used sparingly in research studies for students with disabilities. In Yelland's (1995) review of the research on LOGO, students with disabilities were never mentioned. In the 1980s, three articles were published with a focus on participants with disabilities and computer programming using LOGO. All three articles were practitioner-based and discussed how LOGO could be adapted and used with certain populations of students. Atkinson's (1984) article was directed towards teachers and students with learning disabilities (LD). He briefly explained how a student with LD could engage with the LOGO language and possibly show learning over time. Similarly, Chiang, Thorpe, and Lubke (1984) discussed strategies to use with students with LD and their peers. Students using the LOGO language who are deaf/hard of hearing were discussed by Lange (1985). She examined Papert's (1980) work, analyzed use of the LOGO programming language, and discussed how students with hearing impairments could access the curriculum.

Studies Post-2000

More recently, authors discussed the use of robotics and coding with students with disabilities (Israel, Wherfel, Pearson, Shehab, & Tapia, 2015; Kärnä-Lin et al., 2006; Nickels, 2014), but only four studies were published on empirical research done with students with disabilities (i.e., Adams & Cook, 2013; Miller, 2009; Ratcliff & Anderson, 2011; Taylor et al., 2017; see Table 3). Miller (2009) worked with one participant with profound deafness and limited language skills (i.e., no oral language, no sign language). Through observations and interactions over a period of three months, Miller found the participant could learn skills related

to STEM and communication (e.g., interpersonal communication, programming language).

Miller found the participant's disability in hearing and spoken language did not disrupt his ability to learn coding processes and language. Rather, the internal language processes associated with programming allowed the participant to learn strings of commands, words, and symbols.

Ratcliff and Anderson (2011) observed and interacted with fourth-grade students with disabilities' (i.e., dyslexia, ADHD, dysgraphia, and dyscalculia) experiences with LOGO programming over three, 90-minute sessions. The researchers found students learned more when given a specific task and were taught through explicit instruction than with Papert's constructivist design. Ratcliff and Anderson advised using:

A mediated teaching approach...in a carefully planned and structured manner, using strategies such as setting academic goals, sequencing tasks, asking higher-order questions, giving feedback, discussing errors and common misunderstandings, providing examples of how to apply skills in other contexts, and facilitating awareness and use of planning and problem-solving processes. (p. 248).

Adams and Cook (2013) tracked learning, engagement, and usability of coding software (i.e., Lego Mindstorms) in a 1:1 setting and whole class activities with one participant. The 12-year-old in their descriptive case study was diagnosed with complex communication needs and cerebral palsy. The researchers acted as the instructors for the participant in a 1:1 format and found constant direction was needed for most programming activities. In whole class activities the participant was able to interact with classmates, although he did not initiate most interactions. A speech-generating device allowed the student to actively participate with researchers and peers rather than only observe.

Taylor, Vasquez, and Donehower (2017) researched the abilities of elementary students with Down syndrome (1st-2nd grade; $n = 3$) to learn skills in programming through explicit instruction using a single subject design study. Although the constructionist approach is typically used to teach coding/programming, the researchers concluded this type of instruction is not considered best practice for SWID. Any type of practice in robotics and programming was not realized for SWID until Taylor and colleagues investigated its use with this population. All students in the authors study worked in a 1:1 setting and mastered skills to arrange physical blocks of code to move a robot through four specific levels, representing the four sides of a square. The authors focused primarily on student acquisition of learning the coding blocks and were not provided with a generalization or maintenance phase.

Table 3

Empirical studies for elementary students with disabilities and coding

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Taylor, Vasquez, & Donehower, 2017	Three students with Down syndrome	1 st , 1 st , 2 nd	<i>N</i> = 3	Single subject changing criterion design	IV: explicit instruction in basic coding skills using Wonder Workshop's Dash Robot and physical manipulatives DV: Assessments <ul style="list-style-type: none"> • Key-Math 3 • Percent correct responses 	Visual Analysis Effect size measured by Tau_{novlap} (= 0.982)	<ul style="list-style-type: none"> - All participants followed researcher's instructions to move Dash robot in a square over four levels in treatment phase - All participants learned coding blocks and their placement to move robot - All participants and parents indicated instruction in computer programming was beneficial
Adams & Cook, 2013	One student with complex communication needs (CCN) and cerebral palsy	12-year-old	<i>N</i> = 1	Descriptive case study (with quantitative and qualitative results)	IV: Lego Robot programming activities in one-on-one setting DV: Assessments <ul style="list-style-type: none"> • Goal attainment scaling (GAS; tracked participation of student using a speech generating device) • Morae Usability Analysis Software 	GAS scores determined after each session by author and corroborated by assistive technology team. Observations made by author Effectiveness and efficiency measured using Morae Usability Analysis Software	<ul style="list-style-type: none"> - Participant needed researcher direction for most programming activities - Participant interacted with classmates in a positive manner, but often did not initiate interactions - Adapting the speech generating device to control robot allowed student to actively participate in the class rather than simply observe

Citation	Participants	Age/Grade	<i>N, n</i>	Design/Method	Variables	Analysis	Notable Results/Conclusions
Ratcliff & Anderson, 2011	Students with dyslexia, dyscalculia, dysgraphia, and/or ADHD	4 th -grade students	<i>N</i> = 1	Qualitative	IV: Three 90-minute programming sessions over four weeks DV: Structured interviews, observations		<ul style="list-style-type: none"> - Students were engaged in LOGO programming activities - Intrinsic value to learning skills and problem solving - Benefitted from teacher/researcher instruction and guidance
Miller, 2009	One student with pre-lingual deafness	13-years-old	<i>N</i> = 1	Qualitative	IV: LOGO programming language over three months	Observations	<ul style="list-style-type: none"> - Participant learned skills related to cognition, communication, and programming - Students with learning difficulties in spoken language “may be capable of internalizing spoken language as an abstract symbolic system” (p. 80) - Although participant did not possess language skills, he could understand symbolic language of programming language

The authors cited in the review of the literature (Tables 2 and 3) suggest students in elementary school (PreK-5), with and without disabilities, can learn STEM skills, like basic computer programming. Students were successful when teachers or instructors guided their learning through EBPs, including explicit instruction, scaffolding, and student practice (Doabler & Fien, 2013; Lye & Koh, 2014) and CRA models (Flores et al., 2014). Despite some emerging themes, empirical studies completed with elementary students with disabilities (see Table 3), and specifically with SWID is lacking.

Conclusion

The rise of careers and employment opportunities in STEM creates an educational need to prepare all students with skills and knowledge in STEM content and employment areas in today's society (National Media Consortium, 2017; Vilorio, 2014). Currently, STEM education integrates four subject areas to provide students with active learning opportunities in real-world problem solving (Devlin et al., 2013). Teachers at the elementary level tend to expose students more to science and mathematic subject areas (Xie et al., 2015), but all areas of STEM are to be integrated into the curriculum in most state standards (Carr et al., 2012). The NGSS (2013), NCTM (2000), CCSS (2010), and ISTE (2016) outline standards and objectives to prepare all students for college and career readiness, including skills in STEM. The National Media Consortium (2017) discusses the need to prepare students with STEM related skills as the country's economy and careers are moving in a technological direction. The report outlines the needs and challenges that must be overcome in introducing STEM standards into PreK-21 classrooms.

The introduction of computers and programming technology in the 1960s and 1970s led researchers to study computer programming learning abilities of early elementary students (i.e., Papert, 1972, 1980; Perlman, 1974). Papert (1985, 1980) created the programming language, LOGO, to allow students to explore and construct their own understanding of computer coding. While many studies focused on LOGO and its implications for students' understanding of mathematical concepts (Yelland, 1995), a limited number focused on students with disabilities (i.e., hearing impairment, learning disabilities; Atkinson, 1984; Chiang et al., 1984; Lange, 1985). New programming languages and accessible technology introduced in the early 2000s renewed researchers' interest in teaching coding to students at the elementary level (Bers, Ponte, Juelich, Viera, & Schenker, 2002; Lye & Koh, 2014). Students were afforded the opportunity to use physical manipulatives and tangible coding to "speak" to robots and see their programming work in action. Computer programming, development of code, and problem solving are skill areas integral to STEM education.

Even with the abundance of computer programming languages and student-friendly software available to researchers and teachers, studies targeting the use of these skills with students in early elementary education with developmental disabilities is found in only one study (i.e., Taylor, Vasquez, & Donehower, 2017). Researchers in the field of special education suggest students with significant disabilities (e.g., ID, ASD) can access STEM curriculum when teachers use EBPs (e.g., explicit instruction, scaffolding, CRA teaching model; Agrawal & Morin, 2016; Doabler & Fien, 2013; Spooner & Browder, 2015; Spooner et al., 2011). Science, technology, engineering, and mathematics curricula teaches students to learn and apply problem solving skills (Hefty, 2015), an area of deficit by definition found in students with ID, but a teachable area (Cote et al., 2010; Miller et al., 2015; Scruggs & Mastropieri, 1997). All students,

regardless of their learning ability, need the opportunity to learn core educational material, including critical STEM curricula (Every Student Succeeds Act, 2015).

Newman and colleagues (2011), in the NLTS-2, document a significant difference in percentage of employed people with ID (63%) to those with high incidence disabilities (78%; i.e., speech/language, learning disability, or other health impairments). The majority of people with ID are employed in food-related careers (25.1%) making less per hour than any other disability (\$7.90/hour). The percentage of students with ID in STEM careers is not reported because the number is close to zero. Students with ID need the same opportunities as their peers, including career choices in STEM disciplines, or their unemployment or underemployment rate will continue to be dismal (Basham et al., 2010). Students with significant disabilities are underrepresented in STEM fields (Newman et al., 2011) and research on students gaining skills in STEM-related areas (i.e., computer programming) does not currently exist. Research in areas of computer programming and robotics may be a viable option to build a foundation of understanding in STEM and problem solving skills for students with ID in early elementary school. The purpose of this study is to assess the abilities of young students with ID in learning basic coding skills through explicit instruction.

CHAPTER THREE: METHODOLOGY

Introduction

The researcher in Chapter 3 presents the research methods for this study. The research consisted of two, single subject design studies, identical in all aspects, except the first study focused on kindergarten students with intellectual disabilities (SWID) and the second study focused on kindergarten students without disabilities (SWOD). All students learned, in a 1:1 setting, to code a robot to move in a square through explicit instruction from the researcher. Students learned to code using physical manipulatives in the first half of both studies and tangible coding (i.e., software application on a tablet) in the second half.

Problem Statement and Research Questions

The competencies of students with developmental disabilities (i.e., ID) in computer programming have only been researched in one study (Kärnä-Lin et al., 2006; Lye & Koh, 2014; F. R. Sullivan & Heffernan, 2016; Taylor et al., 2017). Prominent researchers in the field suggested STEM concepts (e.g., computer programming; Bers, 2010) could be taught beginning in early elementary school (Nadelson et al., 2013). Early intervention for SWID is important and providing early intervention in science, technology, engineering, and mathematics (STEM) is just emerging as a consideration in the field. The need for early STEM preparation aligns with future career options. Students with disabilities are underrepresented in STEM careers (Newman, 2011) and the practicality of this population accessing the STEM field needs further research. Any type of practice in robotics and programming is not yet realized for SWID, as only one other study (i.e., Taylor et al., 2017) currently exists aligned with teaching SWID robotics. The intent of this study was to use an evidence-based practice (EBP; i.e., explicit

instruction) to determine the potential success for both SWID and SWOD. The research questions proposed in this study align with this pressing need for SWID in STEM. The questions explored are:

1. To what extent does coding ability of early elementary SWID increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by a rubric?

Null Hypothesis: Coding ability of early elementary SWID will not increase through explicit instruction, using physical manipulatives in a one-on-one setting.

2. To what extent does coding ability of early elementary SWID generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a rubric?

Null hypothesis: Coding ability of early elementary SWID will not increase through explicit instruction, using only an iPad application in a one-on-one setting.

3. To what extent does coding ability of early elementary SWOD increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by rubric?

Null Hypothesis: Coding ability of early elementary SWOD will not increase through explicit instruction, using physical manipulatives in a one-on-one setting.

4. To what extent does coding ability of early elementary SWOD generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a frequency chart?

Null hypothesis: Coding ability of early elementary SWOD will not increase through explicit instruction, using an iPad application in a one-on-one setting.

5. What are the perceptions of the students, parents, and school administration regarding the goals, procedures, and outcomes of the explicit instruction on computer programming as measured by unstructured interviews and surveys?

The computer programming with both groups of students began following IRB approval, parent approval, and participant approval of the methods and video recording to capture each session. Video recordings were used to review any sessions for fidelity of implementation and inter-observer agreement. All recordings were stored on password-protected devices, including an iPad for recording purposes, and the researcher's computer. An inter-observer viewed recordings using password-protected devices. The researcher will keep all videos for a minimum five years, following completion of the study.

Research Design

The research study was comprised of two, single-subject changing-criterion designs. Both Study 1 and Study 2 were completed as ABC designs, utilizing one baseline phase (A), one treatment phase (B), and a generalization phase (C). The baseline phase consisted of students learning to piece together computer code to move the robot in a square without explicit instruction from the researcher. In the treatment phase (B), participants were taught with physical manipulatives, representing basic computer code, using pictures and limited words. The generalization phase (C) consisted of participants using only a tangible application (i.e., iPad

application Blockly) without aide of the physical manipulatives, and the opportunity to independently code Dash to move in a square.

Participants

Participants with Intellectual Disabilities

Participants with ID were recruited and had to meet the following criteria: 1) diagnosed with a mild intellectual disability (IQ score range from 55-70; deficits in adaptive functioning skills; American Psychiatric Association, 2013); 2) ability to recognize basic shapes (i.e., square); 3) no prior use of computer coding manipulatives or software; and 4) ability to use a computer tablet (i.e., iPad). Exclusionary criteria included participants with a hearing or language processing disability. Six SWID were recruited to participate in Study 1, three of which did not meet inclusionary criteria, which left three participants remaining to participate in the study (see Table 4).

Participant 1 (P1-ID) is a Caucasian, female student who at the time of the study was five-years old and diagnosed with Down syndrome. She is from a two-parent family living in Central Florida with four older siblings. P1-ID attended a private school in Central Florida and was fully included in all classroom activities and lessons. Her KeyMath-3 assessment indicated below kindergarten level knowledge in mathematics (<K.0).

Participant 2 (P2-ID) is a Caucasian, male student who at the time of the study was four-years old and diagnosed with Down syndrome. He is from a two-parent family living in Central Florida. He has one older sibling and one younger sibling. P2-ID attended a full-inclusion, public preschool. His KeyMath-3 assessment described him as below level in kindergarten

mathematics knowledge (<K.0). This score is expected as P2-ID was only in preschool at the time of the study.

Participant 3 (P3-ID) is a Caucasian, female student who at the time of the study was seven-years old and diagnosed with Down syndrome. She is from a single-parent family living in Central Florida without siblings, but spends most days a week with her younger cousin. P3-ID attended a public school and was in first-grade. She was removed from the general education setting for the majority of her school day, receiving most academic lessons in a self-contained classroom. It should be noted P3-ID attended a full-inclusion kindergarten class the previous year, but moved to a new school in first grade. P3-ID scored below kindergarten level on the KeyMath-3 mathematics assessment (<K.0).

Participants without Disabilities

Study 2 comprised of kindergarten student participants without disabilities. Students in this study were required to meet the following criteria: 1) *not* diagnosed with a disability; 2) ability to recognize basic shapes (i.e., square); 3) no prior use of computer coding manipulatives or software; and 4) ability to use a computer tablet (i.e., iPad). Six SWOD were recruited for Study 2 (see Table 4).

Participant 4 (P4-WOD) is a Hispanic, female student who at the time of the study was five-years old. P4-WOD is from a single-parent family living in Central Florida. She has one younger sibling. P4-WOD attended kindergarten at a private school. She scored at grade level (K.5) on the KeyMath-3 mathematics assessment.

Participant 5 (P5-WOD) is a Caucasian, female student who at the time of the study was five-years old. She is from a two-parent family with one older sibling living in Central Florida.

P5-WOD attended a private school kindergarten class. Her KeyMath-3 assessment indicated she was at grade level in mathematics (K.7).

Participant 6 (P6-WOD) is a Caucasian, male student who at the time of the study was five-years old. P6-WOD is from a two-parent family living in Central Florida. He does not have any siblings. P6-WOD was in kindergarten at a private school. He scored below grade level (K.2) on the KeyMath-3 mathematics assessment.

Participant 7 (P7-WOD) is a Caucasian, male student who at the time of the study was five-years old. He lives with one parent and stepmother in Central Florida, but sees his mother on a regular basis. He has one stepsibling of the same age. P7-WOD attended kindergarten at a private school. His KeyMath-3 assessment was at grade level (K.9).

Participant 8 (P8-WOD) is a Caucasian, male student who at the time of the study was five-years old. He is from a two-parent home living in Central Florida. He has two siblings, one older and one younger. P8-WOD attended kindergarten at a private school. His mathematics score from the KeyMath-3 assessment was below grade level (K.2).

Participant 9 (P9-WOD) is a Caucasian, male student who at the time of the study was five-years old. P9-WOD lives with his mother and stepfather in Central Florida. He does not have any siblings, but regularly sees his extended family (cousins). He attended kindergarten at a private school in Central Florida. P9-WOD scored above grade level in mathematics (1.4) on the KeyMath-3 assessment.

Table 4

Demographics

	Gender	Grade Level	School	Diagnosis	Key Math 3 (Grade equivalent)
<u>Study 1</u>					
P1-ID	Female	K	Private	Down Syndrome	<K.0
P2-ID	Female	1 st	Public	Down Syndrome	<K.0
P3-ID	Male	Pre-K	Public	Down Syndrome	<K.0
<u>Study 2</u>					
P4-WOD	Female	K	Private	-	K.5
P5-WOD	Female	K	Private	-	K.7
P6-WOD	Male	K	Private	-	K.2
P7-WOD	Male	K	Private	-	K.9
P8-WOD	Male	K	Private	-	K.2
P9-WOD	Male	K	Private	-	1.4

Setting

All participants ($N = 9$) took part in the baseline, intervention, and generalization phases of the study, in a 1:1 format (i.e., researcher and participant), in a room or office space at their school or home. The space used for the research was set up to be free from distractions to the extent possible. On the floor of each research room was a large square (40 or 50 cm²) outlined in tape. The purpose of the square was to give the participants a visual cue to the path and shape the robot, Dash, would complete. The researcher and participants coded while sitting on the floor next to the large square. Manipulatives and software applications were utilized while sitting on the floor.

Instruments

Physical Manipulatives

The primary intervention used physical coding blocks (see Figure 1) to represent code for the students. Blocks were used because the research shows students with ID develop knowledge with more ease using concrete objects (i.e., physical blocks) rather than abstract concepts (e.g., iPad application; Flores et al., 2014; Jimenez, Browder, & Courtade, 2008; Witzel, Mercer and, & Miller, 2003). Each block had a picture or color representing a block of code (i.e., arrow forward, arrow left turn, green for go, red for stop), the word it represents in the coding process (i.e., forward, turn left, go, stop) and was fitted so that participants could slide the blocks together.



Figure 1. Physical coding manipulatives

Tangible coding software

Many coding programs and tools allow elementary students, using digital blocks that drag and drop, to create a program (Kalelioğlu, 2015). In this study, the application, Blockly

(see Figure 2), was used, which was developed by Wonder Workshop specifically for Dash and Dot robots (see description below). This application allowed students to merge blocks of code to tell the robot to follow a variety of instructions (e.g., movement, sound, repeat).

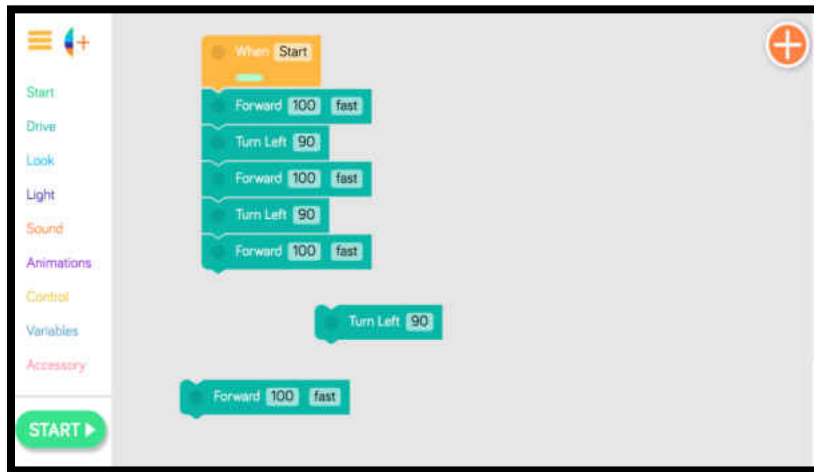


Figure 2. Blockly tangible software application

Dash Robot

Dash is a robot created by Wonder Workshop (see Figure 3). The robot was designed to engage students through voice, sound, and accessories (e.g., attachments for Legos, smartphone mount). Using software applications (e.g., Blockly, Wonder, Go), students control the robot through touch-screen devices (e.g., tablets or smartphones).



Figure 3. Dash robot

Dependent Measures

Frequency checklist

A frequency checklist was used to mark participants' correct or incorrect responses to the researcher's explicit directions (see Appendix A). A total of 25 responses were recorded over four intervention levels. Each level depicted a series of steps, in developing code, to create a portion of a square. When the fourth level was completed, the participants finished the code to program the Dash robot to move in a square. Each level was measured by taking the number of correct directions followed and dividing by the total number possible (25). For example, in Level 1, participants could only get a maximum of four directions correct. Thus, for Level 1, a maximum of 4 out of 25 could be scored (or 16% correct; see Table 5). The following levels (2-4) built upon the preceding level. The participant's score was the total number of frequency checks possible from the previous level(s) plus their score on the current level out of the total number of steps possible (25). For instance, a participant in Level 2 was scored by adding the total possible frequency checks from Level 1 (4) to their score in Level 2 (maximum 7). There was a maximum of 11 points in Level 2 out of the total of 25 for the whole study (or 44%).

KeyMath3 Standardized Assessment

The KeyMath3 standardized assessment was developed to test the mathematical knowledge of students, 4.6-years old to 21 years-old. Internal reliability was considered high (.95) for all grade levels (Kindergarten-12th). Construct validity was measured using correlations with several standardized assessments, including Kaufman's Test of Education Achievement and Iowa Tests of Basic Skills. Content validity was conducted with state mathematical standards and publications from NCTM. The researcher administered the KeyMath3 to all participants.

Documentation of disability

Participants in Study 1 (SWID) needed to provide documentation of a disability to be considered for this study. An intellectual disability is defined as an IQ less than 70, decrease in cognition processes, and significant impairment in adaptive learning skills (American Psychiatric Association, 2013). Children with Down syndrome are diagnosed, either prenatally or at birth, through a series of blood tests, measuring chromosome counts. Adaptive learning deficits and intellectual impairments are common in almost all cases of those with Down syndrome (National Human Genome Research Institute, 2016).

Independent Variable

The intervention for this study was expert, explicit instruction in computer programming to create code to move the Dash robot in a square (see Appendix A). The explicit instruction followed a scripted procedure to introduce, teach, and assess students' learning of basic computer programming skills over Treatment B (see Procedure below). Through single step statements and questions, students received a tick mark for either completing the step accurately, or a tick mark for completing the step inaccurately. In the instance that a step was completed

inaccurately, the researcher reviewed the step again with the student and isolated the section of code (i.e., coding block or tangible code) the student was learning. Upon completing the step accurately, the participant and researcher moved to the next step (no point was awarded because the step was first completed inaccurately).

Procedure

This study used a changing-criterion design. In this type of study, participants began baseline phases at the same time, acting as their own control. After stable level and trend were established over at least five sessions, participants began treatment phases. To successfully move through treatment phases, participants met specific criteria over four levels. Each level was designed with the purpose of moving participants through programming the Dash robot to move $\frac{1}{4}$ of a square. To successfully move from one level to the next, participants demonstrated understanding by following explicit instruction in three sessions, with a maximum of one incorrect response per session. Students with ID and SWOD moved through Baseline A, Treatment Phase B, and Generalization Phase C.

Sessions

All students were greeted before any session began and introduced to the robot, Dash. Participants had the opportunity to interact with Dash (e.g., pick up, move, turn on) if they chose. Dash was turned off once baseline/intervention began. Students were directed to sit with the researcher on the floor next to the 40 or 50 cm² square created prior to the sessions. Upon completion of creating the code for Dash using coding blocks (Treatment Phase B) or coding software (Generalization Phase C), students had the opportunity to turn on and place the robot at a corner of the square. Students then started the code they created.

Baseline

Participants began the study with five baseline sessions. Each baseline session consisted of the researcher giving the participant a goal for the session (i.e., program the robot to move in a square using coding blocks) and the necessary materials to create the code (i.e., Phase A- physical coding blocks). Participants were not guided any further in the session and were scored using a nine-point rubric (see Appendix A). To obtain a score above zero, students had to begin any coding sentence with “Start” or “Go,” otherwise the code would not run.

Treatment Phase (B)

In Treatment Phase (B), participants learned to create code using physical coding blocks (see Figure 1) and following explicit instructions from the researcher. Participants moved through this phase by successfully completing criteria at four levels. Each level (Levels 1-4) was designed to teach the participants to develop code to create $\frac{1}{4}$ of a square and built upon the level preceding it (see Table 4 for calculation of baseline percentages and Levels 1-4 percent changes). In Level 4, the participants created a code to move Dash in a full square.

Level 1.

Level 1 consisted of four steps to develop code to guide the robot in a straight line. At this level, participants were explicitly taught and guided through identifying coding blocks (i.e., Go, Forward, Stop) and aligning them to create a code for the Dash robot to follow. Participants were told the goal of the session (i.e., we will tell Dash to go in a straight line), shown the coding blocks needed for the code individually, and then had to identify coding blocks needed (Steps 1-3). Participants then followed the researcher’s directions to make the code (Step 4; “Find Go, now find and add forward, finally find and add Stop”). If the student followed all four steps

correctly, he or she scored a maximum of four out of twenty-five completed steps (twenty-five equals total possible steps to follow in all four levels). To move to Level 2, participants were required to follow a minimum of three steps correctly (i.e., only one incorrect) over a minimum of three sessions.

Level 2.

Level 2 built upon the previous level (Level 1) and consisted of the researcher teaching the participant to create code to move the Dash robot forward, turn left, and forward again (i.e., $\frac{1}{2}$ of a square). The participant built upon their understanding in Level 1 by completing seven steps, consisting of identifying the coding blocks needed to move the robot forward, turn left, and forward again (i.e., Go, Forward, Turn Left, Forward, Stop), and following the researcher's explicit instructions to create the code to move the Dash robot. If the participant followed all seven steps correctly, he or she scored a maximum of eleven (four previous steps in Level 1 plus seven steps in Level 2) out of twenty-five (44% completion). Participants had to successfully complete a minimum of three sessions of Level 2, with a minimum of six steps followed correctly (i.e., one incorrect) in the sessions to move to Level 3.

Level 3.

Level 3 built upon a participant's cumulative knowledge developed in Levels 1 and 2 to move the robot in $\frac{3}{4}$ of a square. The participant was told the goal of the level (i.e., to move Dash in a straight line, turn left, straight line again, turn left, and a final straight line) and then instructed on coding procedure to program the Dash robot. This level consisted of seven steps. The first six steps were focused on the participant correctly identifying coding blocks (i.e., Go, Forward, Turn Left, Stop). In the final step, the participant followed the researcher's explicit

instructions to create the code to move the Dash robot in $\frac{3}{4}$ of a square. If the participant followed all seven steps correctly, he or she scored a maximum of eighteen (11 steps from Levels 1-2 plus seven steps from Level 3) out of twenty-five (72% completion). Students were required to successfully complete three sessions of Level 3, with a minimum of six steps followed correctly (i.e., one incorrect) in each session to move to Level 4.

Level 4.

Level 4 was the final level in Treatment Phase (B) and consisted of the participant following explicit directions to code the Dash robot to travel in a square (i.e., Go, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Stop). Level 4 consisted of seven steps that built upon student knowledge from the previous Levels 1-3. Students identified necessary coding blocks in Steps 1-6 and then followed the researcher's explicit instruction to create code to move the Dash robot in a square. If the participant followed all seven steps correctly, he or she scored a maximum of twenty-five (eighteen steps from Levels 1-3 and seven steps from Level 4) out of twenty-five (100% completion). Students had to successfully complete three sessions of Level 4, with a minimum of six steps followed correctly (i.e., one incorrect) in each session to complete Treatment Phase (B).

At the conclusion of each session in Levels 1-3, the participants were given the opportunity to independently code the Dash robot to move in a square. It was not mandatory for the participant to complete this portion of the study, but it provided an opportunity to extend learning. Data points were grouped according to how much of the square the participant correctly coded without researcher intervention (i.e., $\frac{1}{4}$ or 25%, $\frac{1}{2}$ or 50%, $\frac{3}{4}$ or 75%). Participants were not provided the chance to independently code the robot to move in a square during Level 4.

Table 5

Treatment Phase (B) level changes (1-4)

Intervention Phase	Possible Points	Cumulative Possible Points	Points needed for level change	Cumulative points needed	Calculation for Percentage	Percent Criterion Change
Baseline	9	9	-	-	$\frac{0 \text{ to } 9}{9} \times 100$	0-100
Level 1	4	4	3-4	3-4	$\frac{3 \text{ or } 4}{25} \times 100$	12-16
Level 2	7	11	6-7	10-11	$\frac{10 \text{ or } 11}{25} \times 100$	40-44
Level 3	7	18	6-7	17-18	$\frac{17 \text{ or } 18}{25} \times 100$	68-72
Level 4	7	25	6-7	24-25	$\frac{24 \text{ or } 25}{25} \times 100$	96-100

Generalization Phase (C).

The Generalization Phase (C) consisted of steps for the participant to develop code to guide the robot in a straight line and then independently code the robot to move in a square, using a tangible coding application (i.e., Blockly). At this level, participants were explicitly taught and guided through identifying tangible coding blocks (i.e., Go, Forward, Turn Left, and Play) as an introduction to the iPad application and were not scored. Participants learned to align the blocks to create a code for the Dash robot to follow a straight line (e.g., “Find Go, now find and add forward, now play the program). After completing the code for a straight line, participants were given the opportunity to independently code Dash to move in a square. A participant’s score was calculated by the portion of the square (four parts) they were able to complete and multiplied by 100 for a percentage: 0/4 score was zero percent (could not create code), 1/4 was 25 percent (coded Dash to move in 1/4 of square), 1/2 was 50 percent (coded Dash to

move in $\frac{1}{2}$ square), $\frac{3}{4}$ was 75 percent (coded Dash to move in $\frac{3}{4}$ of square), or $\frac{4}{4}$ was 100 percent (coded Dash to move in full square; see Table 6).

Table 6

Generalization Phase (C)

Intervention Phase	Possible Points	Fraction of Square Coded	Calculation for Percentage	Percent Change	Criterion
	0	0	$\frac{0}{4} \times 100$	0	
	1	$\frac{1}{4}$	$\frac{1}{4} \times 100$	25	
Generalization	2	$\frac{1}{2}$	$\frac{2}{4} \times 100$	50	
	3	$\frac{3}{4}$	$\frac{3}{4} \times 100$	75	
	4	$\frac{4}{4}$	$\frac{4}{4} \times 100$	100	

Inter-observer Agreement

An inter-observer validated participants' results and fidelity of implementation. The observer was trained to use scoring rubrics following the step-by-step directions given by the researcher. The observer coded a minimum of three sessions with the researcher and asked questions during these training sessions to ensure all procedures were clear and understood. Next, the inter-observer coded 33% of the sessions for students with disabilities and 33% of sessions for SWOD in baseline, treatment, and generalization phases (What Works Clearinghouse, 2011). A random number generator was used to determine which recordings the inter-observer scored. The inter-observer used the same checklist the researcher used for delivering explicit instruction in coding (see Appendix A). Inter-observer agreement was determined using a point-by-point method. To find percentage of points agreed upon, the total

number of agreements between the observers was divided by the total number of agreements plus disagreements, and multiplied by 100.

Data Analysis

A visual analysis of graphs and Tau-U were used to calculate effect size of the data collected for each participant. Participant success was determined by completion of Levels 1-4 in Treatment Phase B and Generalization Phase C. Visual analyses for single subject research designs consist of six measurements: (1) level (mean), (2) trend (slope), (3) variability (range), (4) immediacy of effect (change in data once intervention is introduced), (5) overlap (proportion of data overlap preceding phase), and (6) consistency of data patterns across similar phases (Kratochwill et al., 2010).

Effect sizes were calculated through Tau-U using graphs, data points, and an online single subject effect size calculator (singlecaseresearch.org; Vannest, Parker, & Gonen, 2016). Tau-U is considered to obtain the strongest statistical power and has the greatest sensitivity (Parker, Vannest, & Davis, 2011). The effect size for SWID were calculated separately from the effect size for the study done with SWOD. As this study had several experiments due to groups of participants, individual and overall effect sizes per group were calculated. Tau-U for treatments of less than 50% is considered unreliable, 50%-70% is questionable, 70%-90% effective, and greater than 90% is considered highly effective (Parker et al., 2011).

Social Validity

Participants and their parents answered questions on a short survey (Appendix B) to measure their satisfaction with the goals, procedures, and outcomes of the study (i.e., Research

Question 5; Wolf, 1978). Participants answered questions using emoticons to aide engagement and understanding represented on a three-point Likert scale: (a) sad face is disagree, score of 1; (b) face without smile or frown is neither agree nor disagree, score of 2; and (c) smiling face indicates agree, score of 3. Parents answered questions using a five-point likert scale: (a) strongly disagree, score of 1; (b) disagree, score of 2; (c) neither nor disagree, score of 3; (d) agree, score of 4; and (e) strongly agree, score of 5.

CHAPTER FOUR: RESULTS

Introduction

Students with intellectual disabilities (SWID) are diagnosed with a disability in intellectual and adaptive learning (e.g., problem solving, decision-making skills) and have IQ scores under 70 (American Psychiatric Association, 2013). Science, technology, engineering, and mathematics (STEM) education, including computer programming and robotics, is designed to help students work through problem solving processes, sequencing, and error-correcting techniques (Lott et al., 2013; Lottero-Perdue et al., 2010; Nadelson et al., 2013). Providing SWID and SWOD skills in computer programming at a young age may help students strengthen problem-solving abilities and gain interest in STEM fields (Lott et al., 2013; Lottero-Perdue et al., 2010). The research and results of this study focused on working with young students (PreK-1st), with and without ID, in computer programming/coding and robotics as a way to set the foundation for future success in STEM curricula and careers. The research questions addressed were:

1. To what extent does coding ability of early elementary SWID increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by a rubric?

Null Hypothesis: Coding ability of early elementary SWID will not increase through explicit instruction, using physical manipulatives in a one-on-one setting.

2. To what extent does coding ability of early elementary SWID generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a rubric?

Null hypothesis: Coding ability of early elementary SWID will not increase through explicit instruction, using only an iPad application in a one-on-one setting.

3. To what extent does coding ability of early elementary students *without* disabilities (SWOD) increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by rubric?

Null Hypothesis: Coding ability of early elementary SWOD will not increase through explicit instruction, using physical manipulatives in a one-on-one setting.

4. To what extent does coding ability of early elementary SWOD generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a frequency chart?

Null hypothesis: Coding ability of kindergarten SWOD will not increase through explicit instruction using an iPad application in a one-on-one setting.

5. What are the perceptions of the parents, students, and school stakeholders regarding the goals, procedures, and outcomes of the explicit instruction on computer programming as measured by unstructured interviews and surveys?

The study was divided into two, single subject studies; the first focused on SWID ($n = 3$) and the second focused on SWOD ($n = 6$). All participants were taught skills in basic coding in a 1:1 setting, with the researcher using explicit instruction. The results for each participant are discussed and presented through visual analysis of graphs.

All sessions were video-recorded for the purpose of inter-observer agreement. Participants began the first phase of the study in five baseline sessions to establish stability and trend in data. Participants were scored in Baseline according to the pieces of code they were able

place correctly (i.e., Start, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Stop) without researcher instruction. Participants then began intervention (Treatment Phase B), learning to code the Dash robot in a square, using four levels, of physical manipulatives and explicit instruction. In Generalization Phase C, students learned skills in coding through use of the iPad application, “Blockly”, to direct Dash.

In Treatment Phase B, participants learned to code the robot to travel in a square through completion of levels (See Appendix A; see Tables 5 and 7). Each level represented one leg of the square and was a minimum of three sessions (i.e., Level 1 was $\frac{1}{4}$ of a square, represented by a straight line; Level 2 was $\frac{1}{2}$ a square, represented by the code straight line, turn left, straight line; Level 3 was $\frac{3}{4}$ of a square represented by straight line, turn left, straight line, turn left, straight line; and Level 4 was completing the whole square; What Works Clearinghouse, 2014). Each level was designed to build upon skills learned in the previous level(s). Participants were only allowed one mistake per session, throughout a level, to move to the next level.

Treatment Phase B consisted of 25 tasks over the four levels. In Level 1, participants completed four tasks, while in each of the Levels 2-4, they completed seven tasks. Each level built upon the level preceding it. Students were scored by adding the preceding level(s) total possible tasks completed to their current level tasks completed and dividing by the total number of tasks through the four levels (25). For example, a student in Level 2 that completed all tasks successfully would receive a score of the previous level tasks (Level 1, four tasks) plus their current levels tasks (Level 2, seven tasks). This total was then divided by 25 (total number of tasks in all four levels of Treatment Phase B) to find a percentage of tasks completed. In this example, a total of 11 tasks were completed (four from Level 1, seven from Level 2) and divided by the total tasks (25) to get a percentage of correct responses, or 44%. All participants were

given the opportunity to independently code the robot to move in a square after completion of explicit instruction tasks.

In Generalization Phase C, participants completed five introductory tasks over one level to move Dash in a straight line (not scored; see Figure 4). Each part of the level was designed to allow participants to become familiar with the iPad application (i.e., learn coding blocks, how to move blocks, and how to connect Dash to iPad). Participants were then given the opportunity to independently code Dash to move in a square using the iPad. Participants were scored according to amount of the square they were able to independently code Dash to travel (0-100%; see Tables 6 and 7).

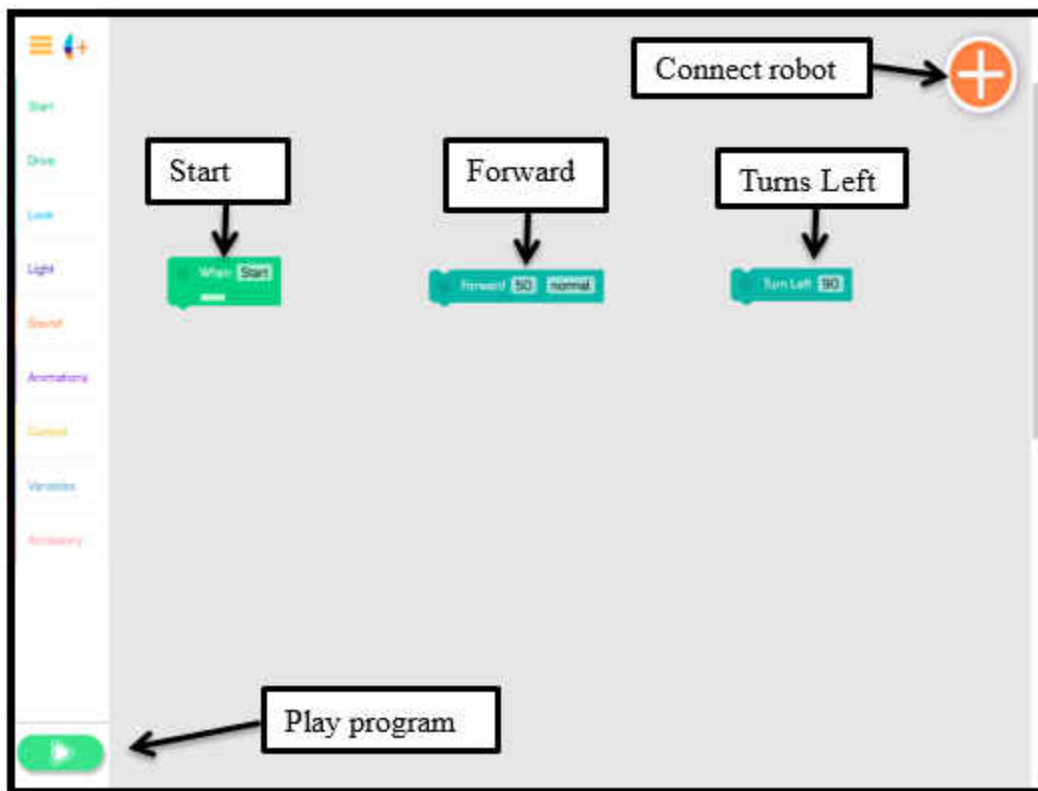


Figure 4. Generalization Phase C introductory tasks

All students with and without ID were able to progress through the baseline phases, treatment phase, and generalization phase; demonstrating stability in level, trend, and variability. Participants with ID required more sessions overall than those without disabilities. In the generalization phase, the participants with ID were unsuccessful in independently coding the robot to travel in a square using the iPad application, Blockly. Two of the participants with ID were able to independently code the robot to travel in a straight line. Participants without disabilities completed each level of the treatment phase in three sessions and were able to independently code the robot to move in a square during the generalization phase in at least one session.

Social validity was collected using a Likert scale survey for the participants, parents, and administration/teachers involved in the study. The majority of students found the programming lessons and work enjoyable and indicated they wished to continue instruction if possible. All parents and administrators/teachers indicated agreement or strong agreement regarding the work completed by their children and students.

Table 7

Levels in Treatment Phases

Treatment Phase	Level	Fraction of square	Code needed to complete
Phase A Baseline	Baseline	4/4	Start, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Stop
Phase B Intervention	1	1/4	Start, Forward, Stop
	2	1/2	Start, Forward, Turn Left, Forward, Stop
	3	3/4	Start, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Stop
	4	4/4	Start, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Stop
Phase C Generalization	-	0/4	Start
	-	1/4	Start, Forward
	-	1/2	Start, Forward, Turn Left, Forward
	-	3/4	Start, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward
	-	4/4	Start, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward
	-	4/4	Start, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward

Group 1: Students with Intellectual Disabilities

Participant 1

Participant 1 (P1-ID) was a 5-year-old female student at a private school in Central Florida. In the five baseline sessions (Phase A), she was unable to code Dash to move in a square (0%). Level 1 of Treatment Phase B consisted of moving Dash in a straight line (i.e., 1/4 of square) and P1-ID completed it over six sessions (8%, 8%, 8%, 16%, 12%, 16%). Level 2

consisted of moving Dash in a straight line, turn left, and moving in a straight line again (i.e., $\frac{1}{2}$ of square). Participant 1 completed this level over four sessions (36%, 40%, 40%, 44%). P1-ID began Level 3 (i.e., moving Dash in $\frac{3}{4}$ of square), making two mistakes in the first session (64%), but successfully completed the level over the following three sessions (68%, 68%, 72%). In the final level of Treatment Phase B, P1-ID was required to follow directions to move Dash in a full square and completed the level in three sessions (96%, 96%, 96%). In Level 4, on the occasion she did make a mistake in identifying the correct block of code for Dash, she was able to identify and fix the error without researcher intervention. In the generalization phase, P1-ID was able to follow the researcher's explicit instructions to code Dash to move in a straight line (not scored); however, she was unable to independently code Dash to move beyond a straight line to create a partial or complete square.

P1-ID demonstrated stable level and trend in the baseline sessions. During the treatment phase, an upward slope is visible with a stable trend and some variability as P1-ID progressed through each level. In Session 11, her percentage dropped because she made one mistake. Once the intervention was introduced a noticeable difference is observed in P1-ID's data points, increasing in a consistent pattern through each level. P1-ID was unable to independently code the robot to move in any part of a square when the intervention was removed (Generalization Phase C). Her effect size was measured by Tau-U, which measured the overlap and non-overlap of data points between baseline and the treatment and generalization phases. P1-ID's data were found to be effective (0.86). Results of this visual analysis can be seen in Figure 5.

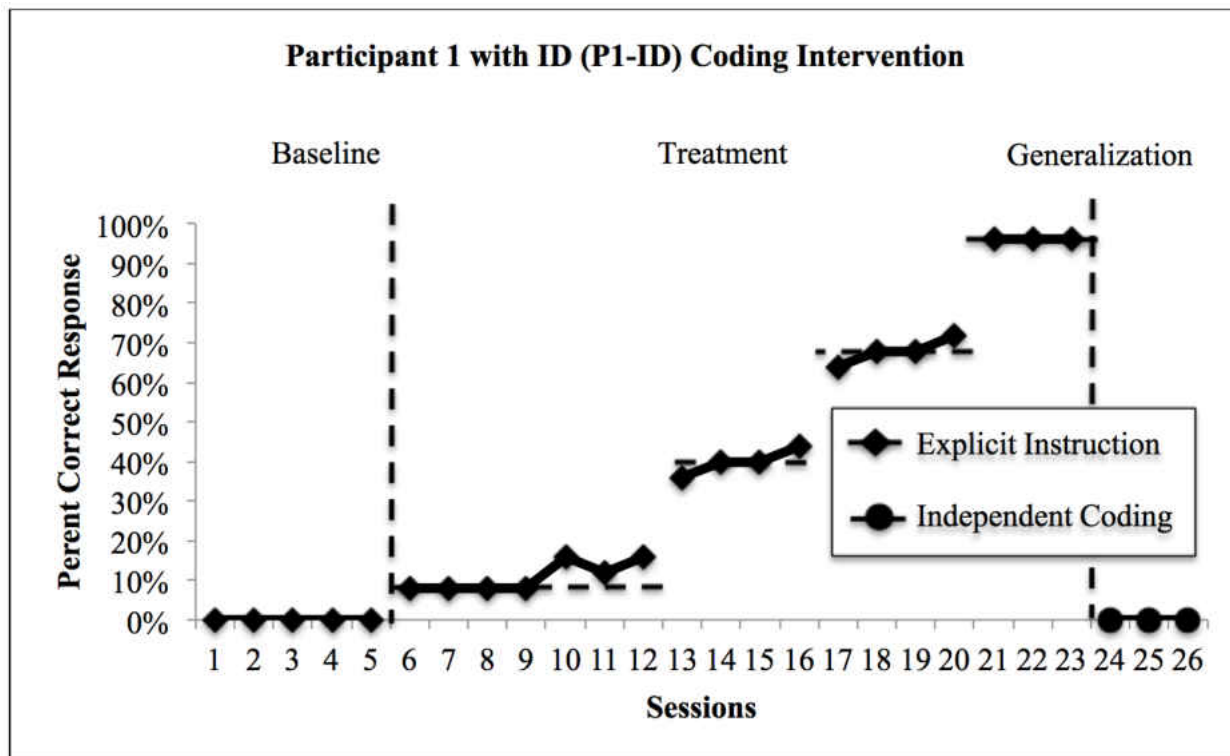


Figure 5. Participant 1-ID visual data

Participant 2

Participant 2 (P2-ID) was a four-year old male student attending a public school in Central Florida. During the five baseline sessions, P2-ID was unable to code Dash to move in a square (0%). He then coded Dash successfully over six sessions (4%, 12%, 8%, 8%, 12%, 12%) in Level 1. Mistakes were due to either incorrectly identifying a piece of code or choosing the wrong piece when making the code. In Level 2, P2-ID coded Dash to move in a ½ square in three sessions (40%, 40%, 40%). P2-ID began Level 3 making two mistakes in the first two sessions (64%, 64%), but was able to follow the sequence for the correct code in the final three sessions (72%, 72%, 68%). P2-ID completed Level 4 to move Dash in a full square in three sessions (100%, 96%, 100%). He was then able to follow researcher instructions on the iPad to

move Dash in a straight line (not scored), but could only generalize tasks learned to independently move Dash in a straight line during one session.

P2-ID demonstrated stable level and trend in the baseline sessions. A stable trend continued in the treatment phase, with an upward slope in data points and immediacy of effect once the intervention was implemented. P2-ID made two mistakes in sessions eight and nine, a mistake in Session 19, and a mistake in Session 21, which may have caused some variability in data. He was unable to independently code the robot to move beyond a straight line (Session 24) during the generalization phase. His effect size was measured by Tau-U and found to be highly effective (0.92). Results of this visual analysis can be seen in Figure 6.

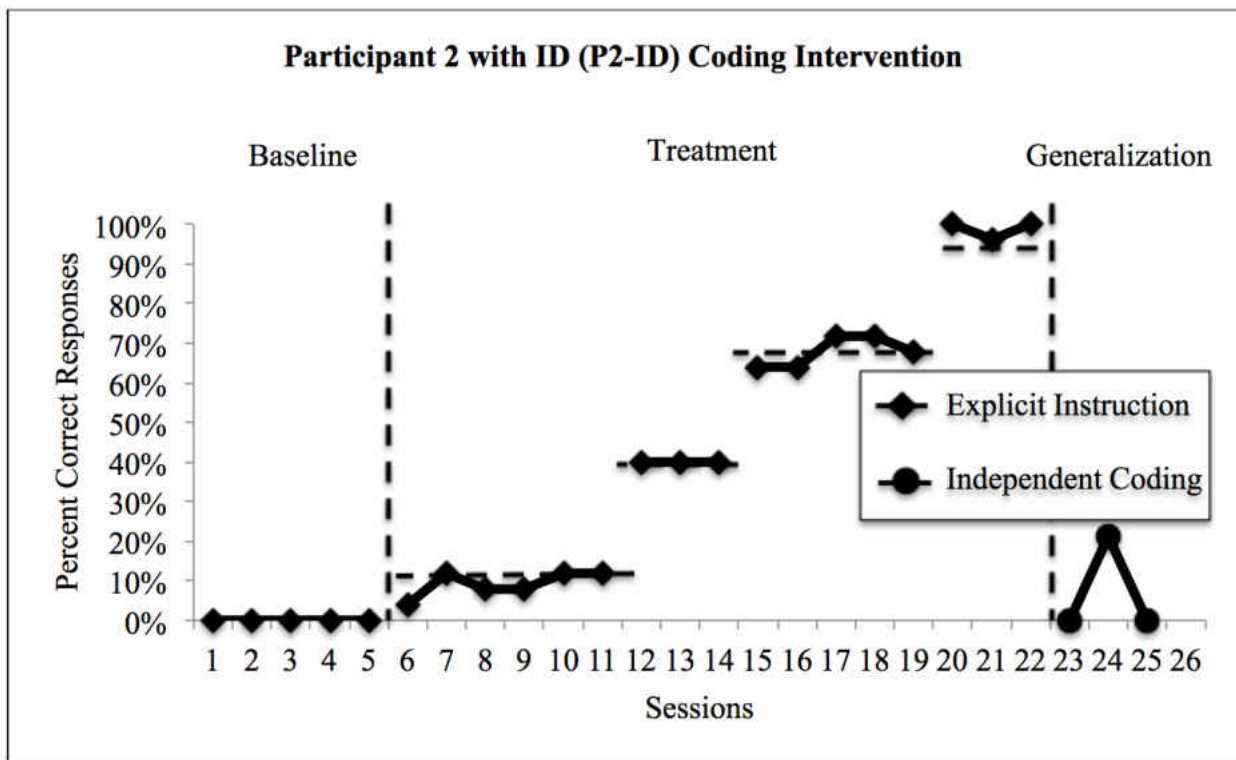


Figure 6. Participant 2-ID visual data

Participant 3

Participant 3 (P3-ID) was a six-year-old female attending a public school in Central Florida. She completed five baseline sessions, unable to code Dash to move in a square (0%). P3-ID completed Level 1 over five sessions (8%, 16%, 8%, 12%, 16%). In Level 2, she followed directions to code Dash to move in $\frac{1}{2}$ a square over three sessions (40%, 40%, 40%), making one mistake in each of the sessions (mistakes were in identifying the correct piece of code). P3-ID completed Level 3 in three sessions (72%, 72%, 68%), making only one mistake in the final session. She completed Level 4 in three sessions (100%, 100%, 96%), with one mistake in the third session. In all three sessions of the Generalization Phase, P3-ID was able to move Dash in a straight line without researcher intervention.

P3-ID demonstrated stable level and trend in the baseline sessions. An immediacy of effect was observed when the intervention was introduced in Treatment Phase B, continuing stability in level, trend, and variability. During Session 8, P3-ID made two mistakes, resulting in a slight downward slope. She also made one mistake in Sessions 16 and 19. P3-ID independently coded the robot to move in a straight line in all three sessions of the generalization phase. Tau-U was used to measure effect size and found to be highly effective (1.0). Results of this visual analysis can be seen in Figure 7.

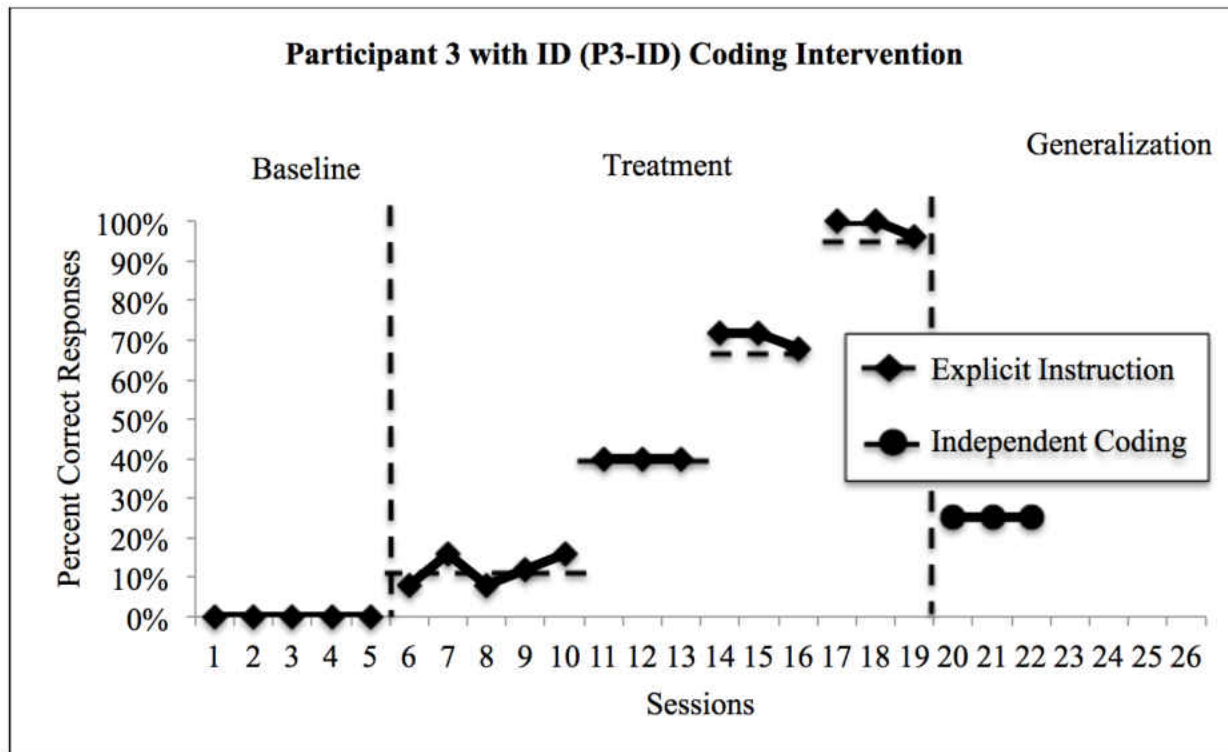


Figure 7. Participant 3-ID visual data

Group 2: Students without Disabilities

Participant 4

Participant 4 (P4-WOD) is a five-year-old female student in a private school in Central Florida. She was unable to code Dash to move in a square in any of the baseline sessions (0%). She completed Level 1 in three sessions (12%, 16%, 16%), Level 2 in three sessions (44%, 44%, 44%), Level 3 in three sessions (72%, 72%, 72%), and Level 4 in three sessions (100%, 100%, 100%). During Level 1 sessions, P4-WOD was unable to code Dash to move in a square independently, but over two sessions in Level 2, she coded Dash to move in three-quarters of a square, while in two sessions in Level 3, she coded Dash to move in a whole square. P4-WOD completed Generalization Phase C over three sessions, as she followed explicit directions to code

Dash to move in a straight line and then independently coded Dash to move in a square in three consecutive sessions.

P4-WOD demonstrated stable level and trend through baseline sessions. Immediacy of effect was apparent in Level 1 and continued through the remainder of the treatment and generalization phases. Level, trend, and variability all positively increased through the phases without any data point overlap. Data patterns were evident through baseline, treatment, and generalization phases, as P4-WOD built upon her learning through each level of the intervention. Effect size was measured by Tau-U and was found to be highly effective (1.0). Results of this visual analysis can be seen in Figure 8.

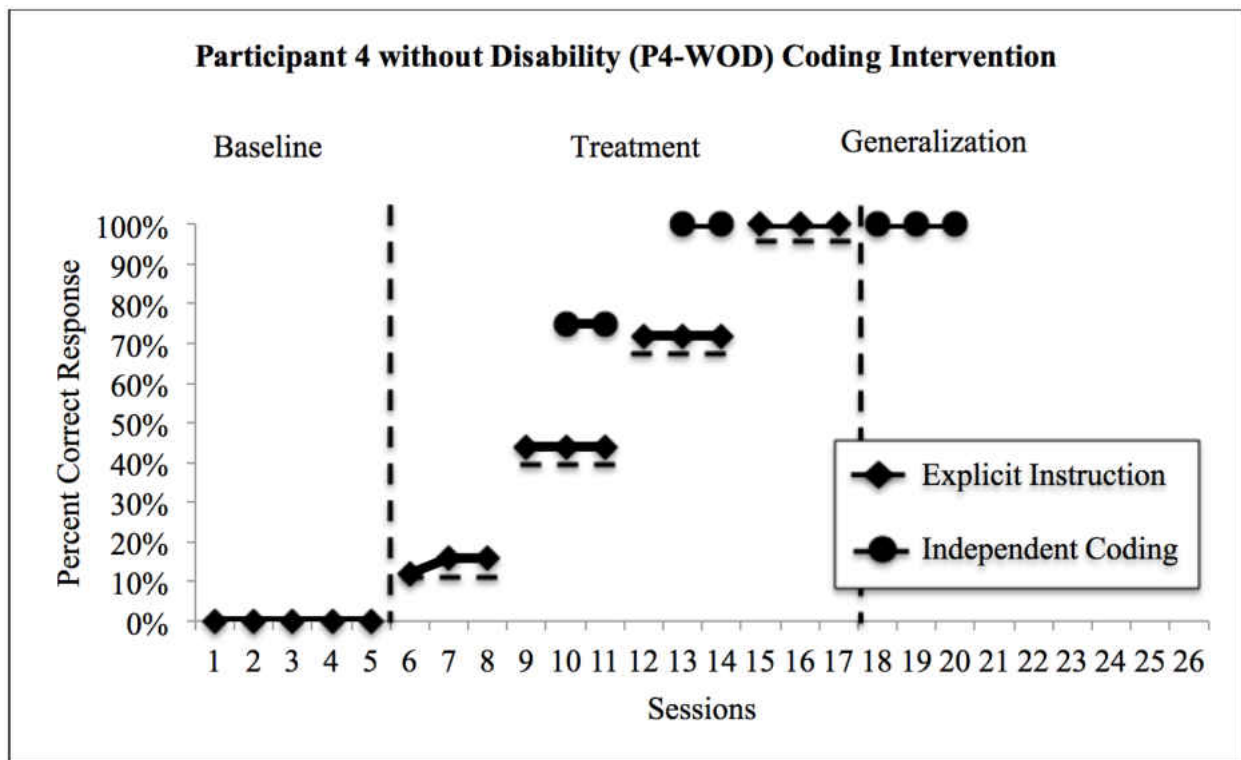


Figure 8. Participant 4-WOD visual data

Participant 5

Participant 5 (P5-WOD) is a five-year old female student in a private school in Central Florida. She began the baseline sessions with correctly placing “Start” and “Forward” in the correct sequence (22%), but was unable to correctly place coding blocks in the last four sessions (0%). P5-WOD completed all four levels of the intervention in three sessions per level, only making one mistake in the first level identifying a coding piece (Level 1: 12%, 16%, 16%; Level 2: 44%, 44%, 44%; Level 3: 72%, 72%, 72%; Level 4: 100%, 100%, 100%). She was unable to independently code Dash to move in a square in any session. In the generalization phase, P5-WOD followed researcher’s explicit instructions to move Dash in a straight line (not scored) and then coded Dash to move in a full square once over three sessions (0%, 0%, 100%).

P5-WOD demonstrated stable level, trend, and variability in baseline, although in Session 1 she was able to place two pieces in the correct order without intervention, but was unable to replicate this progress again in this phase. Immediacy of effect was evident through the treatment phase, with positive increases in level, trend, and variability. In the generalization phase, P5-WOD was unable to independently code the Dash robot to move in any part of a square until Session 20. Tau-U was used to measure effect size and was found to be effective (0.84). Results of this visual analysis can be seen in Figure 9.

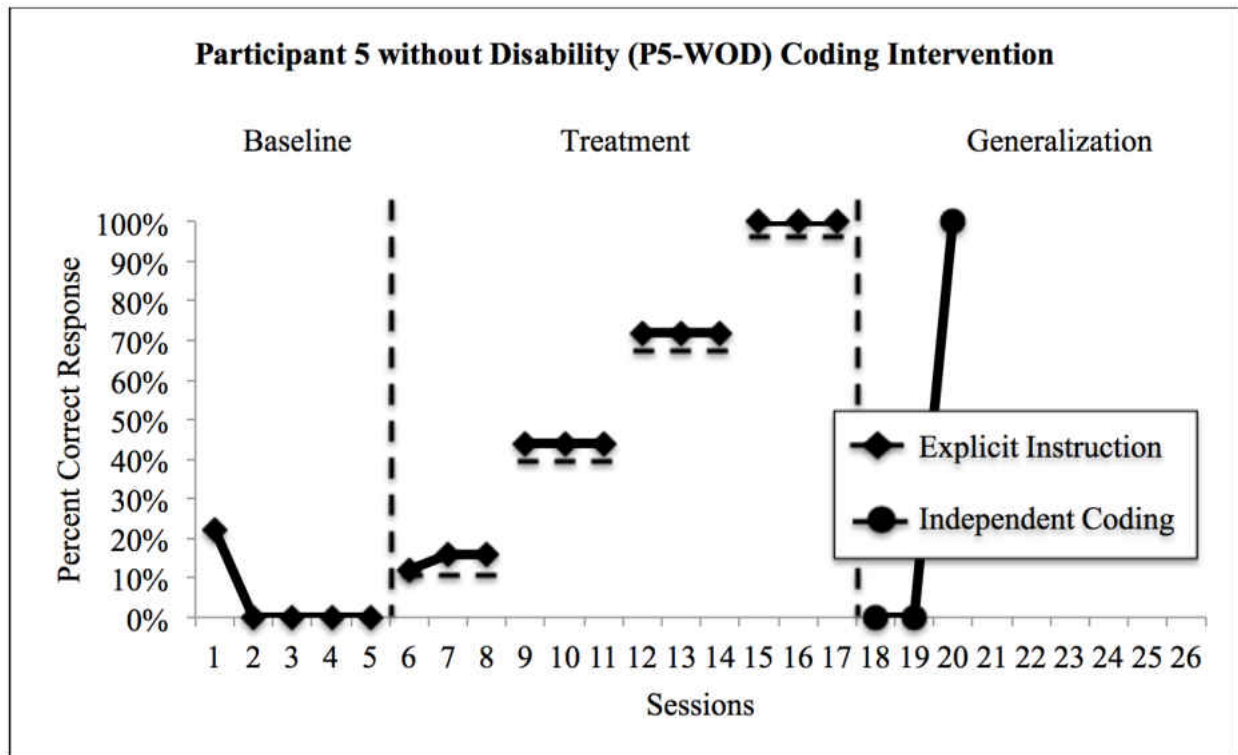


Figure 9. Participant 5-WOD visual data

Participant 6

Participant 6 (P6-WOD) is a five-year old male student in a private school outside of Central Florida. P6-WOD was unable to align code to make Dash move in any part of a square during the five baseline sessions (0%). He then completed all levels of Treatment Phase B in three sessions per level (Level 1: 16%, 16%, 16%; Level 2: 44%, 44%, 44%; Level 3: 72%, 72%, 72%; Level 4: 100%, 100%, 100%). In Levels 2 and 3 of Treatment Phase B, P6-WOD was able to use the coding blocks to independently code Dash to move in a square (100%) in every session except one (Session 13 in Level 3). In the generalization phase, P6 was able to follow the researcher’s explicit directions to move Dash in a straight line over three sessions, and then

independently coded Dash to move in square using the iPad application in two of three sessions (100%).

P6-WOD demonstrated stable level, trend, and variability through baseline and the treatment phases. Immediacy of effect was evident once the intervention was introduced. P6-WOD was able to independently code the robot to move in a square in two of three sessions (i.e., 18, 20) in the generalization phase. Effect size was measured using Tau-U and found to be highly effective (1.0) for P6-WOD. Results of this visual analysis can be seen in Figure 10.

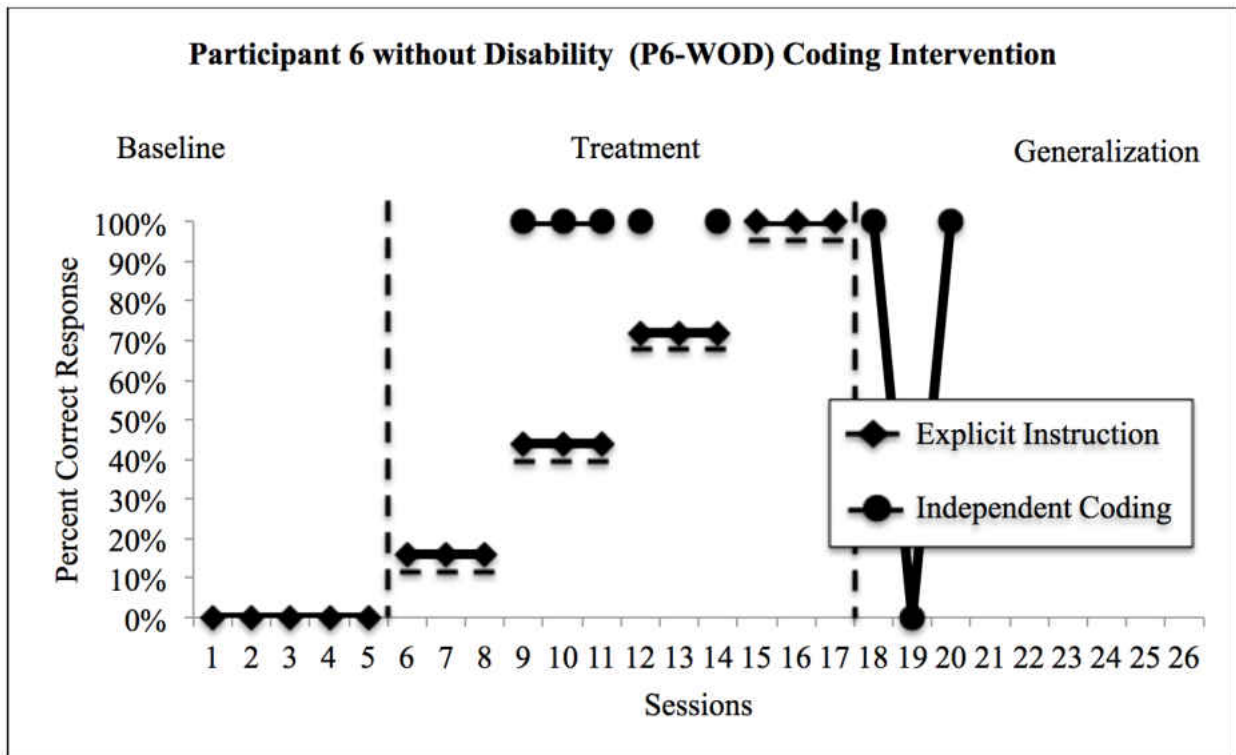


Figure 10. Participant 6-WOD visual data

Participant 7

Participant 7 (P7-WOD) is a five-year old male student in a private school in Central Florida. In his first baseline session, he was able to put the coding block “Start” at the beginning of the code (11%), but was unable to place any of the other blocks in the correct order. In the

final four baseline sessions, he was unable to move Dash in any part of a square (0%). P7-WOD then used three sessions in each of the four levels to successfully complete Treatment Phase B (Level 1: 16%, 16%, 16%; Level 2: 44%, 44%, 44%; Level 3: 72%, 72%, 72%; Level 4: 100%, 100%, 100%). When given the opportunity to move Dash in a square independently, P7-WOD was able to code Dash to move in three-quarters of a square in two sessions of Level 2; while in Level 3, he coded Dash to move in a full square over all three sessions. In the generalization phase, P7-WOD learned to use the iPad application, Blockly, to move the Dash robot. P7-WOD was able to generalize his learning over three sessions to first follow explicit instructions to move Dash in a straight line (not scored), and then independently coded Dash to move in a square.

P7-WOD demonstrated stable level, trend, and variability during the baseline phase. During Session 1 of this phase, he was able to place one piece correctly without intervention, but was unable to replicate this progress again during the remaining four sessions of baseline. Stable level, trend, and variability continued through the treatment and generalization phases, with immediacy of effect evident once intervention was introduced. In the generalization phase, P7-WOD independently coded the robot to move in a square during all three sessions. Effect size was measured using Tau-U and found to be highly effective (1.0). Results of this visual analysis can be seen in Figure 11.

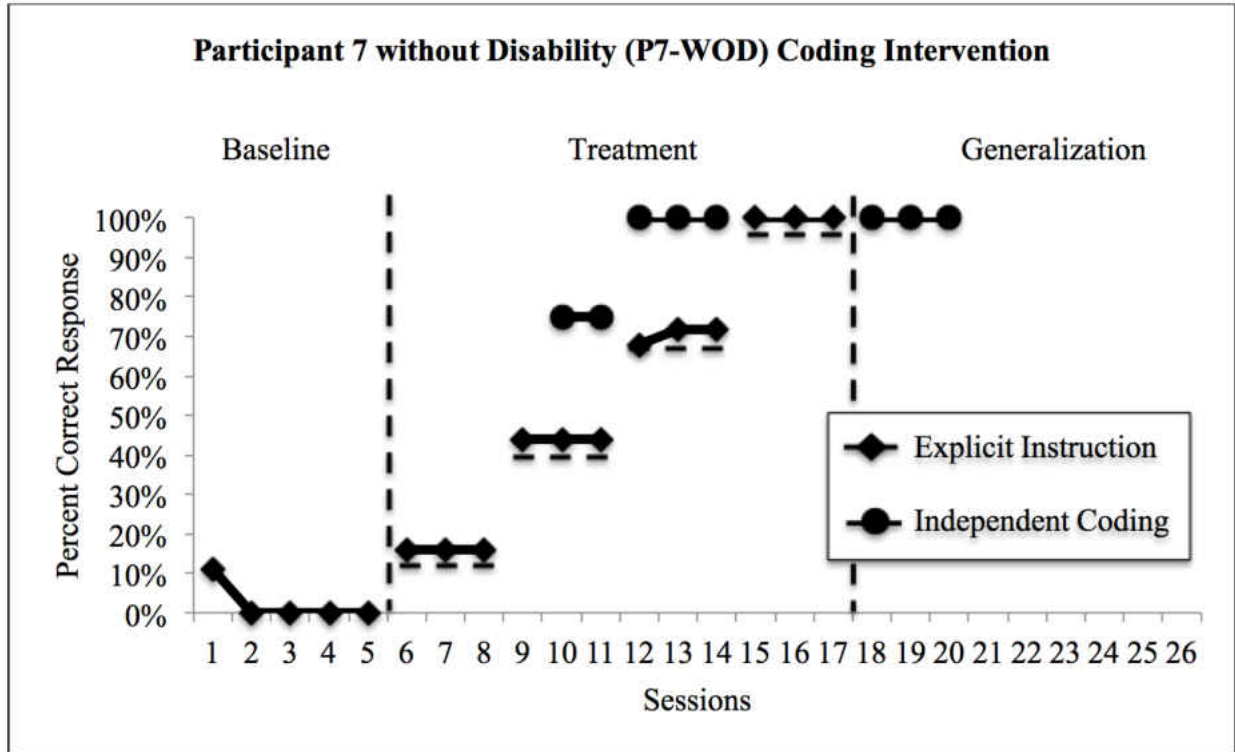


Figure 11. Participant 7-WOD visual data

Participant 8

Participant 8 (P8-WOD) is a five-year old male student in a private school in Central Florida. He was unable to code Dash to move in a square during the five baseline sessions (0%). He learned the code to move Dash through each of the four levels in only three sessions per level (Level 1: 16%, 16%, 16%; Level 2: 44%, 44%, 44%; Level 3: 72%, 72%, 72%; Level 4: 100%, 100%, 100%). After each session, P8-WOD had the opportunity to independently code Dash to move in a square. After the third session of the first level, he coded Dash to move in a ½ square (50%). In the first two sessions of the second level, he coded Dash to move in the full square (100%, 100%). In the third level, he coded Dash in the second session to move in a full square (100%). P8-WOD completed the generalization phase using the iPad, following explicit

instruction to move Dash in a straight line, and then moved Dash in a full square in two out of three sessions.

P8-WOD demonstrated stable level, trend, and variability through the baseline and treatment phases. Immediacy of effect was noticeable once the intervention began. During the generalization phase, P8-WOD was able to independently code the robot to move in a square in two of three sessions. Tau-U was used to measure P8-WOD’s effect size and found to be highly effective (1.0). Results of this visual analysis can be seen in Figure 12.

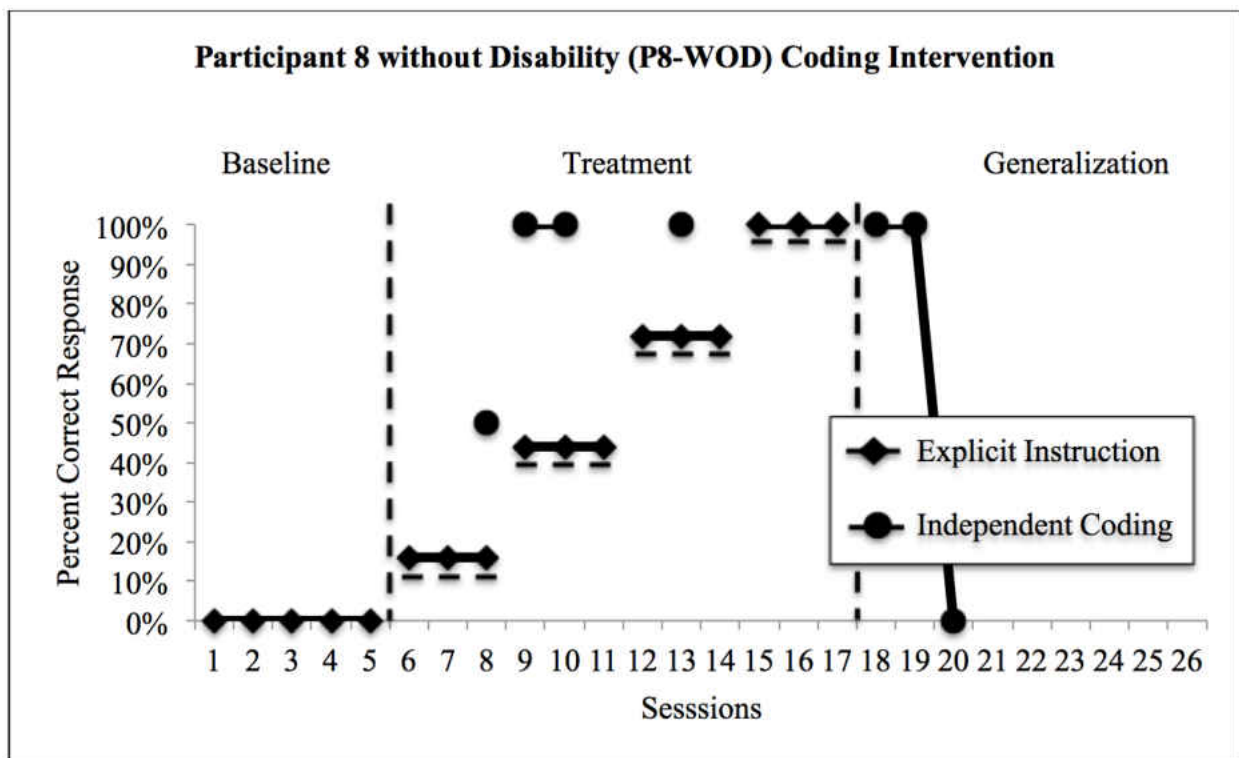


Figure 12. Participant 8-WOD visual data

Participant 9

Participant 9 (P9-WOD) is a five-year old male student in a private school in Central Florida. He was unsuccessful in moving Dash in any portion of a square during the baseline sessions (0%). In the Treatment Phase, P9-WOD successfully completed each level, over three

sessions, without any mistakes (Level 1: 16%, 16%, 16%; Level 2: 44%, 44%, 44%; Level 3: 72%, 72%, 72%; Level 4: 100%, 100%, 100%). P9-WOD was able to independently code Dash to move in a square in the second and third sessions of Level 1, three-quarters of a square in the third session of Level 2, and a full square in all three sessions of Level 3. He then generalized his learning to the iPad application, first following researcher explicit instruction to move Dash in a straight line, and then independently coding Dash to move in a square with 100% accuracy over three sessions.

P9-WOD demonstrated stable level, trend, and variability through the baseline, treatment, and generalization phases. Immediacy of effect was evident once the intervention was introduced. He was able to independently code the robot to move in a square during all three sessions of the generalization phase. Effect size was measured using Tau-U and found to be highly effective (1.0). Results of this visual analysis can be seen in Figure 13.

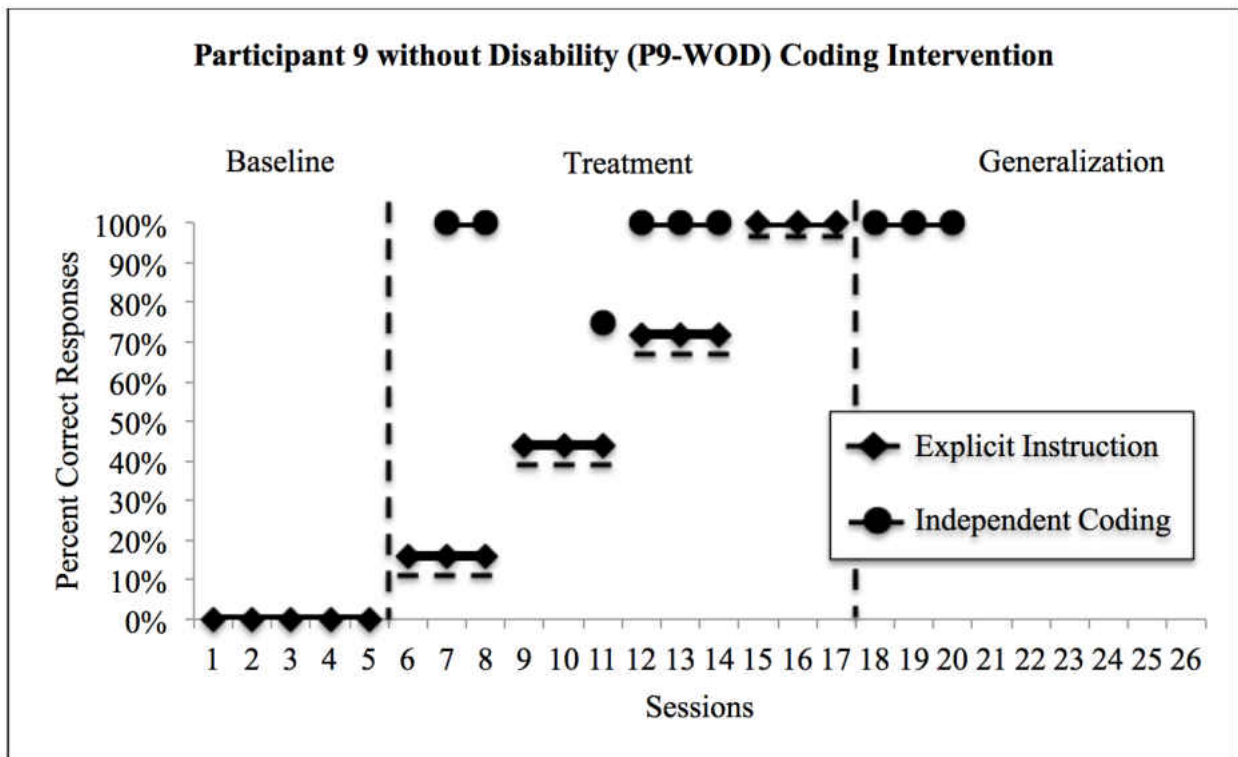


Figure 13. Participant 9-WOD visual data

Visual Analysis of Groups 1 and 2

Visual analysis for single subject research designs consists of six measurements: (1) level (mean), (2) trend (slope), (3) variability (range), (4) immediacy of effect (change in data once intervention is introduced), (5) overlap (proportion of data overlap preceding phase), and (6) consistency of data patterns across similar phases (see Figures 14 and 15; What Works Clearinghouse, 2014). For all students in studies 1 and 2, baseline showed stability in level, trend, and variability. Only two participants (i.e., P5-WOD and P7-WOD) were able to arrange any code correctly, but were only able to do so in one session. In Treatment Phase B, upward trend through the four levels of intervention depicted students' learning and understanding of the coding sequence (see Figures 5-13). Immediacy of the effect is observable between the baseline and treatment phases for all students, as each progressed through the treatment phase once the intervention (i.e., explicit instruction in coding) was introduced. The generalization phase varied in level and trend, as students were required to independently demonstrate their learning. Immediacy of effect can be seen as the intervention was removed from this portion of the study and the majority of SWOD were able to code independently. Participants with ID demonstrated the most difficulty in the generalization phase. There was no overlap between the baseline session and the treatment and generalization sessions for SWOD. Overlap occurred for SWID between the baseline and generalization phases. Effect size relating to overlapping and non-overlapping pairs also was measured using Tau-U (see next section and Table 8) and was found to be highly effective.

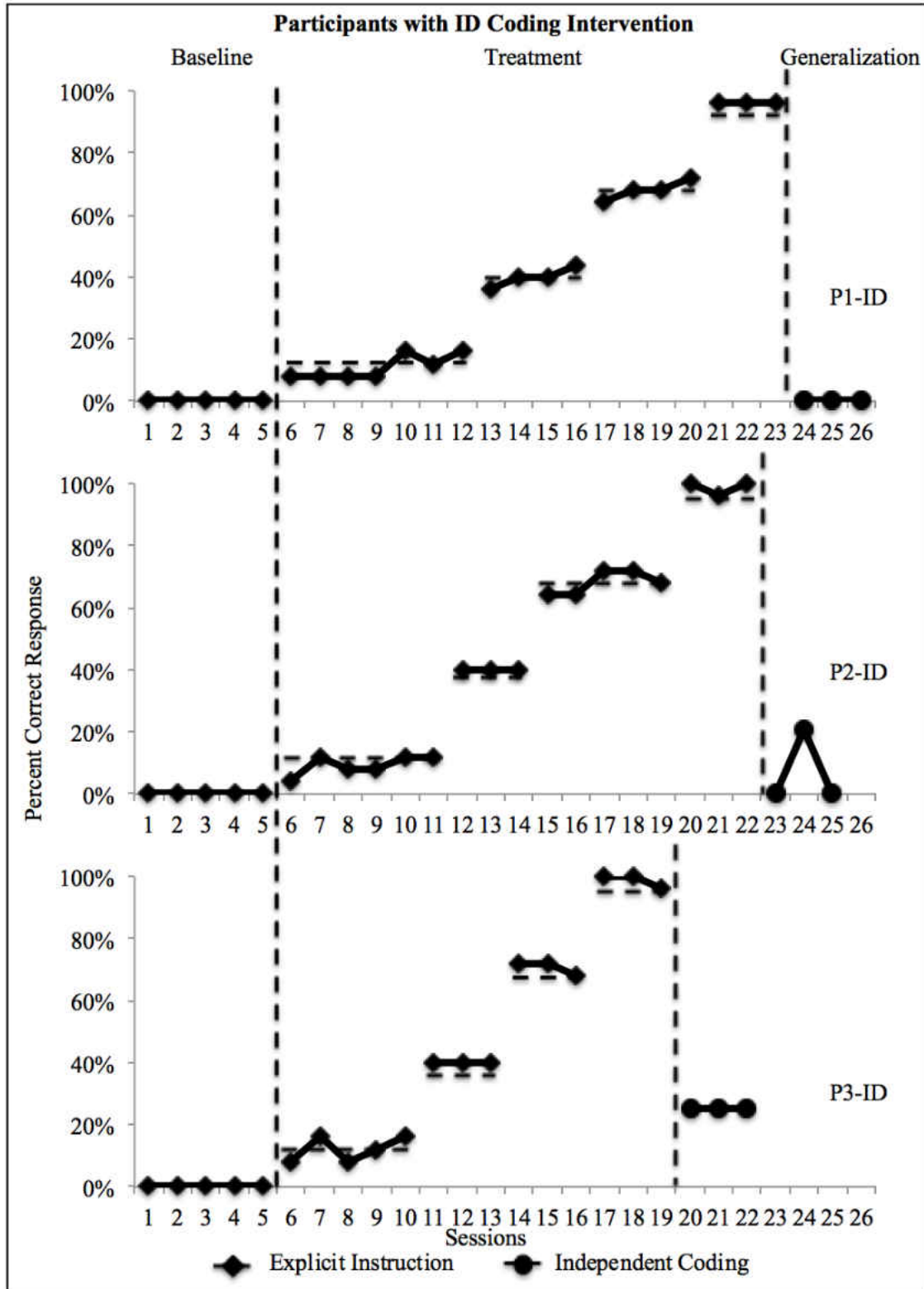


Figure 14. Participants with ID graphs for visual analysis

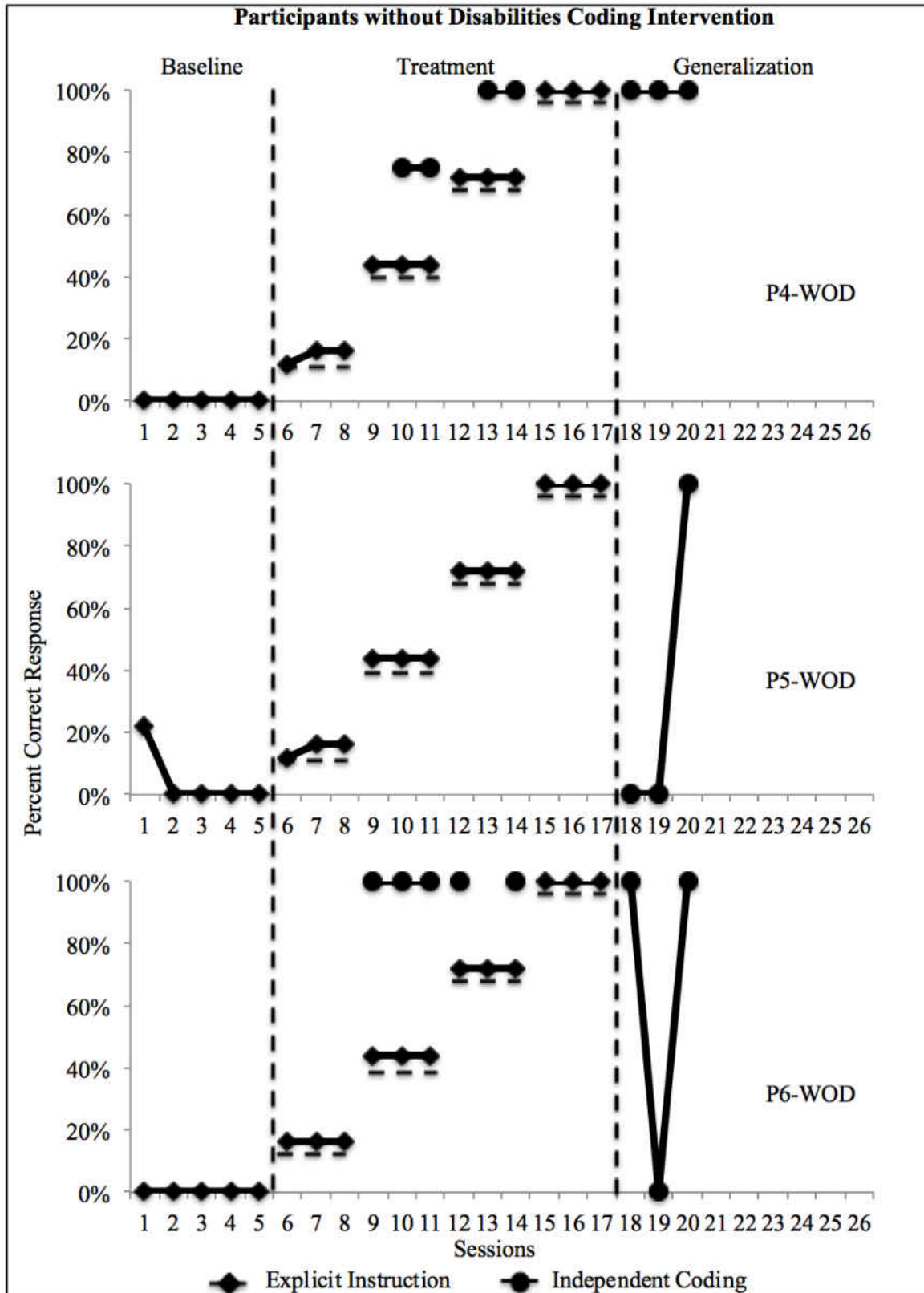
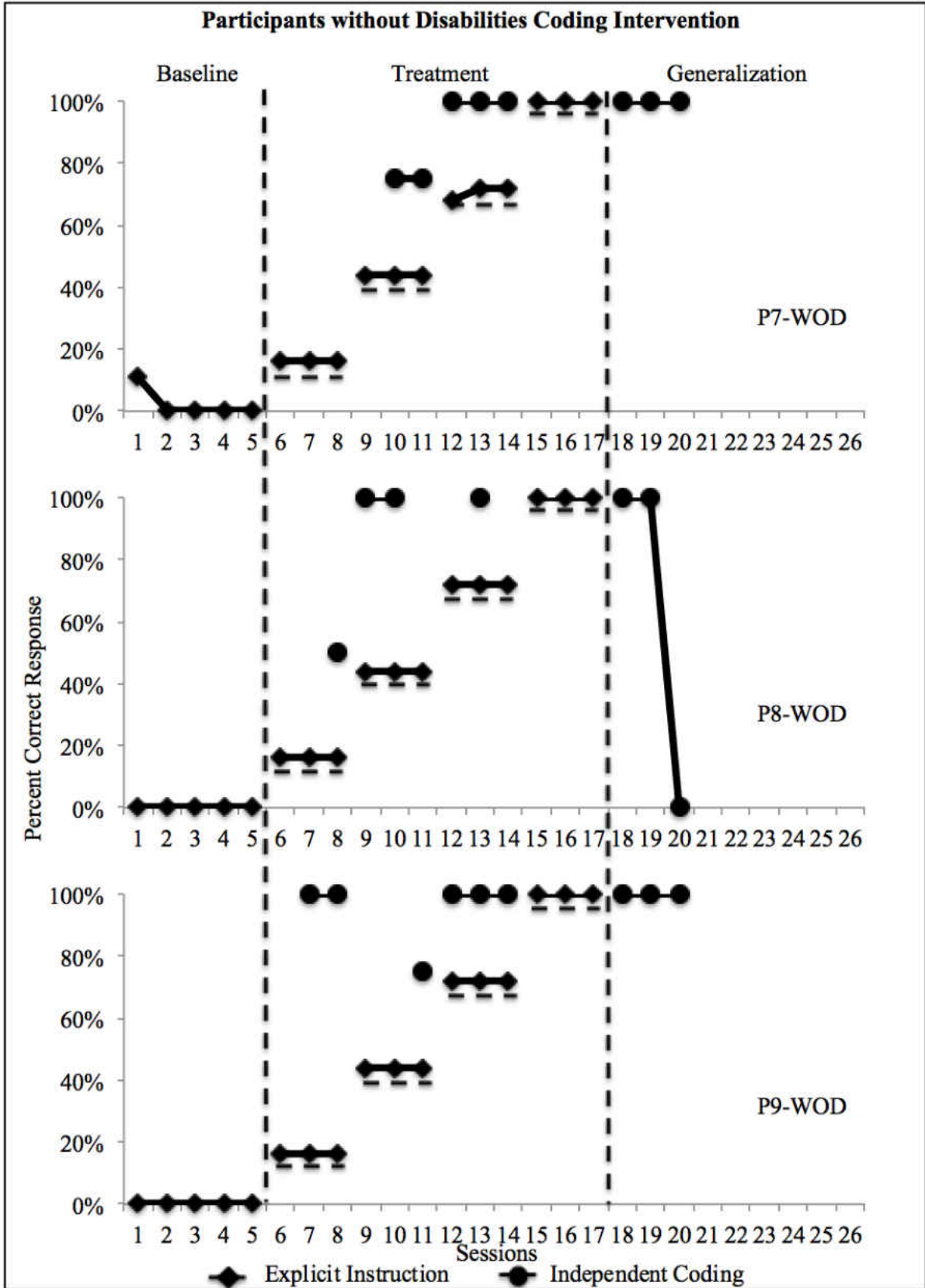


Figure 15. Participants with ID graphs for visual analysis



(Figure 15 continued)

Effect size

Effect size was calculated through Tau-U using graphs, data points, and an online single subject effect size calculator (Vannest et al., 2016). Tau-U is considered to obtain the strongest statistical power and has the greatest sensitivity, measuring overlap and non-overlap of data (Parker et al., 2011). Effect size was calculated for the first study with SWID, and a separate calculation for effect size was calculated for SWOD (see Table 8). As this research consisted of two individual studies due to groups of participants, individual participant effect sizes and group effect sizes were calculated separately (see Table 8). Tau-U for treatments of less than 50% is considered unreliable, 50%-70% is questionable, 70%-90% effective, and greater than 90% is considered highly effective (Parker et al., 2011). All participant effect sizes were found to be effective (P1-ID, P5-WOD) or highly effective (P2, P3, P4, P6, P7, P8, P9). Combined effect size for both groups of students was considered highly effective (SWID = 0.90; SWOD = 0.97).

Table 8

Effect sizes (Tau-U)

Group	Participant	Effect Size (Tau-U)
Students with ID	P1-ID	0.86
	P2-ID	0.92
	P3-ID	1
	Combined-ID	0.90
Students without Disabilities	P4-WOD	1
	P5-WOD	0.84
	P6-WOD	1
	P7-WOD	1
	P8-WOD	1
	P9-WOD	1
	Combined-WOD	0.97

Inter-observer Agreement

An inter-observer validated participants' results and fidelity of implementation. The observer was trained to use scoring rubrics following the step-by-step directions given by the researcher. The observer coded a minimum of three sessions with the researcher and was able to ask any questions for clarity of directions. The inter-observer then coded 33% of sessions for students with disabilities and 33% of sessions for SWOD, including baseline and generalization phases (What Works Clearinghouse, 2014). The inter-observer used the same checklist the researcher used for delivering explicit instruction in coding (see Appendix A).

Inter-observer agreement was determined using a point-by-point method. To find percentage of points agreed upon, the total number of agreements between the observers was divided by the total number of agreements, plus disagreements, and multiplied by 100. A minimum of 80% agreement was required to be obtained for this study, with a preferred agreement of 90%. A random number generator was used to determine which 33% of recordings the inter-observer scored. Agreement was found to be 97.5%. Fidelity of implementation was calculated on 33% of sessions (total steps completed accurately by total number of implemented x 100) to address this limitation. Fidelity was found to be high percentage for both groups (Students with ID= 97.14%; SWOD= 98.45%; Combined percentage= 97.99%).

Social Validity

Participants, their parents, and their teachers/administrators (as applicable) were asked to answer questions on a short survey (Appendix B) to measure their satisfaction with the goals, procedures, and outcomes of the study (i.e., Research Question 5; Wolf, 1978). Participants answered questions using emoticons to aide engagement and understanding represented on a

three-point Likert scale: (a) sad face was disagree, score of 1; (b) face without smile or frown was neither agree nor disagree, score of 2; and (c) smiling face indicated agree, score of 3. Average score of SWID ($n = 3$) was 2.7, while the average score for SWOD ($n = 6$) was 2.89. Scores from both groups indicated overall agreement with statements on the survey.

Parents/teachers/administrators answered questions using a five-point likert scale: (a) strongly disagree, score of 1; (b) disagree, score of 2; (c) neither agree nor disagree, score of 3; (d) agree, score of 4; and (e) strongly agree, score of 5 (if in a two-parent home, only one parent from each household responded to the survey). Overall average score for the parents of SWID ($n = 3$) was 5, indicating strong agreement. The average of parents of SWOD ($n = 6$) was 4.87, indicating agreement or strong agreement to the survey. The classroom teacher and the school administrator for all kindergarten participants (SWID $n = 1$, SWOD $n = 6$) also filled out the social validity survey with an average score of 5, indicating strong agreement.

Conclusion

Students with and without disabilities accessed robotics and computer coding through explicit instruction, physical manipulatives, and a tangible coding program on an iPad. Students with ID ($n = 3$) demonstrated understanding between baseline sessions and Treatment Phase B. Stable level, trend, and variability are observed in the participants' graphic data (Figures 5-7), depicting skills in basic coding of a robot to travel in a square. In Generalization Session C, participants were tasked with independently coding the robot to travel in a square, using an iPad application (i.e., Blockly). Participant 1-ID was unable to independently code, while P2-ID and P3-ID were able to code the robot to move in a straight line in at least one session.

Students without disabilities ($n = 6$) established gains between baseline sessions, Treatment Phase B, and Generalization Phase C. All participants followed explicit directions to successfully move through the treatment phase in the minimum required sessions. Five of the participants learned to independently code the robot to move in a square by Level 2. All participants completed the study by independently coding the robot to travel in a square, using an iPad, in at least one of three generalization sessions. All SWID and SWOD indicated positive agreement to social validity questions regarding their feelings towards the study, although two participants specified they would not like to continue learning the coding intervention. Parents, teacher, and administrator all agreed or strongly agreed to questions about the goals, procedures, and outcomes of the study.

Effect size was calculated using Tau-U through an online, single subject effect size calculator (Vannest et al., 2016). Individual effect sizes were calculated and found to be either effective ($P1 = 0.86$, $P5 = 0.84$) or highly effective ($>.90$) for all participants. Group effect sizes also were calculated for SWID (0.90) and SWOD (0.97) and found to be highly effective. Inter-rater agreement and fidelity of implementation was calculated on 33% of baseline, treatment, and generalization sessions. Inter-rater agreement was found to be 97.5%, while fidelity of implementation was 97.99%.

Early elementary students with and without disabilities have demonstrated skills in basic coding (Strawhacker & Bers, 2015; Taylor et al., 2017). Robotics and computer programming/coding incorporate all areas of STEM (Lottero-Perdue et al., 2010) and can help strengthen intellectual skills (e.g., problem solving) in students (Miller et al., 2015). Students in this study were taught skills in basic coding through explicit instruction, which is an evidence-

based practice for SWID (Browder et al., 2012). Further discussion of participant results, including similarities, differences, and future implications are provided in Chapter 5.

CHAPTER FIVE: DISCUSSION

Introduction

In this chapter, the researcher presents the discussion, limitations, and future implications of a research study, focused on teaching basic skills in computer programming to students in PreK-1 settings, with and without disabilities. This study was completed to build upon the fact that many future careers will require skills in science, technology, engineering, and mathematics (STEM); and all students should be prepared for use of these skills in citizenship and careers (Vilorio, 2014). Students with disabilities, specifically those with intellectual disabilities (ID), are grossly underrepresented in both STEM courses and careers (Newman et al., 2011). To prepare students with ID (SWID) for STEM-related careers, an early introduction to computer programming/coding is a viable option to consider for strengthening problem solving and adaptive learning skills while providing a strong foundation for future college and career options (Miller et al., 2015).

In this study, nine students in early elementary grades (PreK-1) participated in learning basic skills related to computer programming, using explicit instruction, the Dash robot, physical manipulatives, and an iPad application (i.e., Blockly). Three of the participants were diagnosed with Down syndrome, and six participants had no documented disability. All participants completed the baseline, treatment, and generalization phases, through explicit instruction in a 1:1 setting, with the researcher. Results from each group of students (i.e., with Down syndrome or without disabilities) are discussed and findings are compared and contrasted with current literature in the field.

Purpose and Research Questions

The purpose of this study was to research the abilities of young students (PreK-1st), with and without ID, to demonstrate basic skills related to computer programming and coding taught through explicit instruction (Browder et al., 2008; Devlin et al., 2013), concrete manipulatives, and tangible interfaces (i.e., iPad; Flores et al., 2014).

The researcher explored the following questions:

1. To what extent does coding ability of early elementary students with intellectual disabilities (SWID) increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by a rubric?
2. To what extent does coding ability of early elementary SWID generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a rubric?
3. To what extent does coding ability of early elementary students without disabilities (SWOD) increase through explicit instruction, using physical manipulatives in a one-on-one setting, as measured by rubric?
4. To what extent does coding ability of early elementary SWOD generalize to independently code, using only an iPad application in a one-on-one setting, as measured by a frequency chart?
5. What are the perceptions of the students, parents, and school administration regarding the goals, procedures, and outcomes of the explicit instruction on computer programming as measured by unstructured interviews and surveys?

Robotics and computer programming/coding simultaneously incorporate all four areas of STEM while providing students with active learning and problem solving experiences (Kalelioğlu, 2015). Previous researchers have demonstrated early elementary students' learning and mastery of basic skills in computer programming/coding are possible (Barker & Ansorge, 2007; Bers et al., 2002; Cejka et al., 2006; Lye & Koh, 2014). In a review of the literature, Lye and Koh (2014) note young students benefit from adult guidance in programming to have a meaningful experience and facilitate understanding. Most robotics and coding curriculum is based on constructionist approaches (Papert, 1980), allowing students to control their own learning. These approaches in robotics are a way to provide students with hands-on activities to test their thinking and results through trial-and-error and problem solving (Sullivan & Heffernan, 2016), but at times, students need explicit instruction to move forward in the coding process (Kalelioğlu, 2015; Lye & Koh, 2014).

As this study focused on SWID, the evidence-based practice (EBP) of explicit instruction (Browder et al., 2012; Devlin et al., 2013) was used to teach basic skills in coding/programming. At this time, constructivist/constructionist approaches are not EBPs or best practice to use in developing the learning and knowledge of SWID. The researcher chose to use a practice validated in the literature for SWID and to also apply this same principle to SWOD to see the impact of explicit instruction on both populations. The researcher firmly believes best practice for teaching coding/programming is a constructivist/constructionist approach, but currently this type of instruction would not be considered best practice for SWID. Yet, any type of practice in robotics and programming is not yet realized for SWID, as only one other study currently exists aligned with teaching SWID robotics. The intent of this study was to use an EBP to determine the potential success of explicit instruction for both SWID and SWOD and then to further

explore a more constructionist approach to instruction for SWID in the future. A large group design study is being explored to compare the constructivist/constructionist approach to explicit instruction for SWID at the early grade levels in the fall of 2017.

While numerous researchers depict work in coding and robotics with students at early grade levels, only a minimal number of researchers discuss empirical data for students with disabilities (Miller, 2009; Ratcliff & Anderson, 2011; Taylor et al., 2017). Of these studies, only one focused on students with ID (Taylor et al., 2017). As demonstrated by Taylor, Vasquez, and Donehower (2017), SWID in elementary school can demonstrate basic skills related to computer programming. The research suggests future exploration on using an iPad application (e.g., Blockly) to introduce a generalization phase to monitor students' independent skills in coding.

Research Questions and Discussion for Participants with ID

Research Question 1

Research question 1 explored if participants could demonstrate coding ability in basic programming at the early elementary level. Although the participants with ID ($n = 3$) in this study differed in grade level (Pre-K, kindergarten, first grade), they were similar in mathematical content knowledge (<K.0) as measured by the KeyMath3 assessment. These three participants demonstrated basic skills in coding and robotics, through explicit instruction, over four levels. The number of sessions required for mastery of this group was similar between participants and previous research (i.e., Taylor et al., 2017). The first level of Treatment Phase B was expected to be the most difficult for all participants, as they were introduced to the coding process for the first time. The SWID reached mastery of the first level after receiving five to eight sessions using explicit instruction.

As the participants progressed through the next three levels, the number of sessions needed to complete each level decreased. Each of the participants identified as ID completed Levels 2 and 4 in three sessions. P1-ID completed Level 3 in three sessions, but P2-ID and P3-ID needed additional time (five and four sessions respectively). The three participants were able to recognize mistakes in their code by the final level of Treatment Phase B. P1-ID often commented, “Oops, that’s the wrong one!” when she chose the incorrect piece of code, shaking her head and laughing aloud. She almost always appropriately fixed the code when she realized her error by independently replacing the incorrect choice with the correct piece of coding. Recognizing errors and fixing code is an important component of programming (Bers et al., 2014). Students with ID in this study were unable to independently code the robot to move in any part of a square during sessions in Treatment Phase B, but were able to demonstrate knowledge of the coding pieces and follow researcher instructions to make Dash move.

Research Question 2

Research Question 2 focused on the abilities of SWID to generalize knowledge of concrete manipulatives to the tangible code, using the iPad application Blockly (Generalization Phase C), an area of research not yet explored (Taylor et al., 2017). Each session of the generalization phase began with the participants identifying the coding blocks (i.e., Start, Forward, Turn Left, Play) on the application. All three participants with ID were consistently able to identify the blocks when directed by the researcher. The participants were then asked to independently code the robot to move in a square. P1-ID was unable to make the robot move during any of the sessions, P2-ID coded the robot to move in a straight line (1/4 of square) in one session, and P3-ID coded the robot to move in a straight line (1/4 square) during all three

sessions. This outcome shows mixed and limited generalization of skills for SWID, but leaves several points for further research and discussion.

Summary Research Question 1 and Research Question 2

Explicit instruction in novel curriculum for SWID, especially in material that is deemed outside of their understanding due to age, grade, or disability, should be explored. This study replicated work by Taylor, Vasquez, and Donehower (2017), but added generalization sessions to help researchers further understand student learning. Although SWID were unable to generalize coding a square without researcher explicit instruction, the students showed gains in understanding verbal instructions between all four levels in Treatment Phase B with this type of instruction. Further, SWID were then able to identify coding blocks and follow explicit instructions to move the robot using the tangible coding application, Blockly. Students appeared to have a gap in their abilities to independently apply or generalize basic coding skills, which is a weakness considered in part of the identification process to classify a student as having an ID. Future research may need to include additional, next steps to generalize coding skills or more levels of repetition to overcome or to solidify the need for scaffolded, explicit instruction for this population of students in coding.

Research Questions and Discussion for Participants without Disabilities

Research Question 3

Research Question 3 focused on the basic skills in coding of students without disabilities (SWOD; $n = 6$), taught through explicit instructions, and use of concrete coding blocks. The participants were in kindergarten and were found to be similar in mathematical skill levels (K.2-1.4) as measured by the KeyMath3 assessment. Participants completed Treatment Phase B in

three sessions per level. All participants, except P5-WOD, were able to independently code the robot to move in at least $\frac{3}{4}$ of a square by Level 2 (P5-WOD was unable to independently code the robot to move in any part of a square during Treatment Phase B). Many of the participants attempted to code Dash to move in a square in Level 1 and Level 2, but were unsuccessful because they included too many coding blocks or put the blocks in the wrong order. In Level 3, all participants, except P5-WOD, were independently coding the robot to travel in a square. This ability to generalize through three levels appears different from initial observations of SWID. Future research should explore this difference in generalization.

Research Question 4

Research Question 4 was designed to measure the ability of SWOD to generalize their understanding from concrete manipulatives to the iPad application, Blockly. All participants were able to identify the coding pieces they needed and could follow the researcher's instructions to move Dash in a line. The students questioned why the piece representing "Stop" was not present on the iPad application, but they came to realize it was embedded in the functions of the program. Participants were then asked to independently code Dash to move in a square. While the students differed in their consistency of constructing the correct code (i.e., made mistakes in either the length of code or in the order of code), all participants independently made Dash move in a square in at least one session. If a participant incorrectly coded Dash, they were asked if they could find the mistake they made and correct it (this session was scored as 0% if first attempt was wrong). In the majority of cases, the students recognized a mistake in the code when the robot did not follow the path they planned. The students were surprised and would make comments that the path was not correct and were then able to look at their constructed code and alter it to correctly move the robot. As stated earlier, fixing mistakes and trying new code is

a necessary skill in programming (Bers et al., 2014). The SWOD were more proficient at fixing their code after the robot did not follow their intended path than SWID.

Summary Research Question 3 and Research Question 4

All SWOD demonstrated knowledge of the coding procedures in Treatment Phase B in the minimum required sessions (three) per level (four). In this phase, the students differed in their ability to generalize their knowledge to independently code Dash to move in a full square. The participants made mistakes in either the length of the code (e.g., made Dash travel too far) or the order of the code (e.g., made Dash turn instead of go forward), but overall, SWOD were successful in generalizing their understanding of the verbal directions during the four levels of the treatment phase. In Generalization Phase C, participants were able to identify and use the tangible coding pieces and procedures needed to move Dash in a straight line on the iPad application. When P7-WOD was given directions and told he was going to code Dash without the researchers help in the first session of the generalization phase, he commented, “I don’t know if I can do that!” He then proceeded to surprise himself and complete each session in the phase appropriately. P9-WOD quickly understood the coding procedure on the iPad to the extent he was able to modify code to make the robot move either faster or slower, as well as to make the robot make noises and the lights change color.

The demonstration of understanding in Treatment Phase B and Generalization Phase C for SWOD was expected, as similar studies completed by researchers have depicted student learning of robotics and programming with kindergarten participants (Berry et al., 2016; Bers et al., 2014). For example, Bers and colleagues (2014) studied the abilities of kindergarten SWOD in understanding and developing code, including fixing mistakes, completing challenges, and choosing the correct lines of code for the robot to follow. Teachers led students through 60-90

minute constructionist lessons, with the majority of time focused on independent work. Students were able to ask questions of their peers and teacher if they were unsure of their code or next steps. Students demonstrated learned skills and understanding over time, through a final project. Sullivan and Bers (2013) studied the comparison of learning of male and female kindergarten SWOD. Students were taught over six lessons, using a tangible coding robotics program. Both groups performed equally, demonstrating equal ability to complete a final project and incorporating knowledge from all lessons.

The participants in this study were taught through explicit instruction, whereas robotics and programming is typically taught through constructivist/constructionist methods (Sullivan & Heffernan, 2016). Explicit instruction, an EBP for SWID (Browder et al., 2012; Doabler & Fien, 2013), was used to keep the two groups in this study as similar as possible (aside from documented ID). Ratcliff and Anderson (2011) found fourth grade students with high incidence disabilities (e.g., ADHD) learned programming software to a greater extent when the lessons were short, focused on specific tasks, and supported through explicit instruction.

In Treatment Phase B, all participants were given the opportunity to demonstrate learning by independently coding the robot to travel in a square after an intervention session was completed. This opportunity was the only time the students were given to code without researcher intervention during the treatment phase and was implemented to avoid a ceiling effect. Five of six SWOD were able to independently code the robot to travel in at least $\frac{3}{4}$ of a square by the second level; only one SWOD was unable to independently code Dash to move in any portion of a square. Researchers implementing constructionist teaching methods observed student learning over lessons and application of skills in several final projects (Fessakis et al., 2013; Kazakoff & Bers, 2012; Strawhacker & Bers, 2015). Future researchers may want to

focus on student learning through explicit instruction in coding with a final, novel, project to demonstrate learning. Comparison studies between constructionist robotics curriculum and explicit instruction have not yet been researched. Future studies should focus on the differences in student learning in robotics and coding, dependent upon type of intervention (i.e., constructivist/constructionist vs. explicit).

Social Validity

Participants and educational stakeholders (i.e., parents, teachers, administrators) responded to a social validity survey to measure the goals, procedures, and outcomes of this study (Wolf, 1978). Most participants responded positively to the intervention, including engagement with the robot and programs used to move the robot (engagement was observational data and the opinion of this researcher). Only two students (one with ID and one without disabilities) indicated no desire to continue learning with the robot or the programming languages. The two students' responses were not surprising, as computer programming is not a subject area of interest to all students. All parents and school administration either agreed or strongly agreed with statements or questions regarding the study, including the benefits of the robotics study/curriculum for kindergarten students.

Discussion

The following discussion points emerged from the results of this exploratory study from the two different groups (SWID and SWOD), but these points also are presented through the eyes of the lead researcher, a PreK-2nd grade general and special education teacher. Students with ID and SWOD demonstrated coding ability to move a robot in a square through explicit instruction from the researcher. Students with ID required more sessions than SWOD. This

finding was an anticipated result, as students with ID have supposedly significantly lower IQ scores and intellectual learning (American Psychiatric Association, 2013). Interestingly, though, was the varied length and number of sessions between the two groups. Students with ID, on average, used 17.3 sessions to complete the study, while those without ID, on average, used 12 sessions (only Treatment Phase B sessions were counted, as all students completed a mandatory five baseline sessions and three generalization sessions). The difference in averages of sessions is most likely attributed to the first level of coding, which took longer for students with ID to complete. Students with ID averaged 17 mistakes per participant, whereas SWOD averaged less than 0.2 mistakes. All students, regardless of disability, were able to identify problems in their code (either when prompted or independently) when following the researcher's explicit instructions.

Students with ID have a disability in intellectual functioning, as their diagnosis suggests (American Psychiatric Association, 2013). Researchers proposed several ways to address student learning in STEM content for SWID and other disabilities (Israel et al., 2015; Wakeman et al., 2013). Israel Wherfel, Pearson, Shehab, and Tapia (2015) suggest teachers implement Universal Design for Learning (UDL), student collaboration and cooperation, and experimenting with coding applications and software that best suits students' needs. Wakeman, Karovnen, and Ahumada (2013) encourage teachers to recognize when a student is struggling and make necessary changes to instruction and content to affect student success. While SWID in this study made more mistakes than SWOD, perhaps researchers need to look at new ways to present programming lessons and support students of all abilities in their learning (Israel et al., 2015).

The most important difference between the SWID and the SWOD appeared to be the ability to generalize knowledge to independently code the robot to travel in a square, during both

Treatment Phase B and Generalization Phase C. This finding was an expected difference, as generalization is an intellectual and adaptive skill, both part of the diagnosis of ID. Students with ID also were not able to independently code the robot in the treatment phase, and only two were able to generalize their learned skills to the iPad application. The majority of SWOD (five of six) were able to move the robot independently by the second level of the treatment phase. All SWOD were able to make the robot move in a square during the generalization phase on the iPad. The SWOD all identified a pattern in the code they created (i.e., Forward then Left Turn, four times). The majority also expressed they knew how to move the robot in a full square because they only needed to follow the pattern. Students with ID never verbally identified the coding pattern, although P1-ID and P3-ID often knew which piece of code came next before the researcher's instructions.

The diagnosis of ID includes the terms “intellectual and adaptive functioning deficits in conceptual, social, and practical domains” (American Psychiatric Association, 2013, p. 33). Intellectual functioning includes reasoning, problem solving, planning, abstract thinking, judgment, academic learning, and experiential learning. Adaptive behavior refers to communication, social skills, independent living, and school/work functioning. These skills align with the complexity of coding, and in this study, students with ID demonstrated knowledge of parts of these skills through explicit instruction. Yet, for many teachers, the diagnosis of an ID may stir up feelings of inadequacy, worry, anxiety, and inability to teach these children (Ruppar, Neeper, & Dalsen, 2016). Teachers should consider looking at this definition in terms of what SWID can do, rather than simply what they cannot. This inclusivity of thinking must include the mastery of critical skills in STEM areas that are predicted to align with future employment outcomes (Vilorio, 2014).

Students with ID can demonstrate abilities of basic skills in coding, as evidenced in this study and others (Taylor et al., 2017). The purpose of this study was not to prepare all students to become computer scientists and engineers, but rather to demonstrate SWID can do more when given the right supports and instruction in STEM. Students demonstrated understanding through explicit instruction, although robotics is generally taught through a constructionist methodology (Sullivan & Heffernan, 2016). In this study, explicit instruction was used for all students, especially beneficial to those participants with ID as an EBP (Browder & Cooper-Duffy, 2003; Doabler & Fien, 2013). Teachers can implement a successful robotics curriculum in their classroom grounded in constructionism (Sullivan et al., 2013), but some students, especially SWID, will fall behind and not understand the concepts. Yet, teaching all students robotics through 1:1 explicit instruction, as occurred in this study, is not feasible for a PreK-2nd grade teacher. A balance in teaching methods may help all students achieve in lessons, through small groups, peer-to-peer modeling, and teacher support (Lye & Koh, 2014; Strawhacker & Bers, 2015).

All students have dreams and desires, strengths and weaknesses, which are often vastly different than their peers. The idea behind this study, and behind the use of robotics, coding, programming, and STEM instruction for this population, was to provide students with the opportunity to learn skills in reasoning, problem solving, planning, abstract thinking, judgment, academic learning, and experiential learning; which consequently are the areas of significant need for students identified with the label of ID (American Psychiatric Association, 2013; Devlin et al., 2013; DiFrancesca et al., 2014). Students in this study were not expected to learn all constructs of programming in kindergarten (or early elementary grades), for the same reason students are not expected to understand calculus in kindergarten. These students were expected

and encouraged to demonstrate knowledge of the basic foundation and building blocks of robotics to have the opportunity to nurture an area of learning that is exciting, engaging, and activity-based (Bers, Flannery, Kazakoff, & Sullivan, 2014) to build upon for the future.

All students should be provided with the opportunity to learn (ESSA, 2015; Yudin & Musgrove, 2015). Explicit instruction is an EBP for teaching SWID (Browder & Cooper-Duffy, 2003; Doabler & Fien, 2013), and teaching SWID revolves around reaching them in a way that they best learn. Teaching is the chance to introduce new concepts, scaffold education through active learning and supports, and provide students with the option to make mistakes and to learn to fix them. This adaptability was observed in this study and provides evidence that, even at an early age with proper supports and scaffolding, SWID can and do adapt their thinking while learning a new skill. The pace may differ, but the ability to do so should be a consideration in practice.

Reaching all Students

When this study began, not one student with *or* without a disability could make sense of the coding language placed in front of them. Students averaged almost zero percent correct during baseline sessions. Only two students placed any of the blocks in the correct order during baseline, which was found to be by chance as they did not replicate the placement again. The students could not independently formulate a way to arrange the blocks of code as to move the Dash robot with a purpose. But, when given explicit instruction, told the meaning of each coding block, and how to arrange them to “talk” to the robot, *all* students demonstrated understanding. After the initial instruction, they not only understood, they excelled. Students with ID and SWOD developed knowledge of the system of coding at different rates and levels of understanding, but they both learned. The differences in ability or disability did not suggest one

group can do something the other could not (Browder et al., 2012). Rather, these variances showed the students learned differently.

Browder and colleagues (2012) demonstrated a difference in student learning through a study with teachers, who had students with moderate and severe developmental disabilities, to teach grade level (i.e., secondary) science curriculum. The researchers helped teachers break-down content-specific standards, collaborate with general education colleagues, and develop specific learning goals tied to EBPs (e.g., task analysis, graphic organizers, systematic prompting, and feedback; Spooner et al., 2011). Students showed increased knowledge in the curriculum they were taught, and teachers felt the training and intervention was useful, practical, and supported their instruction for students with mild to severe developmental disabilities. Browder and colleagues suggested future studies should research students' abilities to generalize learned subject matter to real world scenarios, as well as the extent to which student learning can be in the general education classroom. Fostering the growth of skills in robotics and programming for both SWID and SWOD through new interventions, including application to real-world problem solving, are next steps in implementing accessible, STEM curriculum in early grade levels.

Students with ID may choose not to become computer scientists or engineers. They may not even excel in learning to code, but in this study, they were given the chance to explore an arena of academic curriculum and potentially future employment not currently talked about or explored for SWID at their age/grade level (Faulkner et al., 2013; Goodnough et al., 2014). Students with ID can learn novel information (like coding) when presented in a context that is understandable, relatable, and meets their learning needs (Clements & Sarama, 2011; Freeman et al., 2014). A gap existed for SWID in the generalization of skills, from concrete manipulatives to

independently structuring of the code, but this gap could be remediated through further scaffolding (Doabler & Fien, 2013).

Implementing Coding/Robotics Activities in Early Elementary Classroom Routines

Implementation of a coding/robotics curriculum in the early elementary grades can be used as a stand-alone subject area or tied to other daily lessons (e.g., reading, mathematics, science; Kazakoff & Bers, 2012). Teaching has always evolved, but most progressions have not considered the learning temperaments of all students, but rather just the majority (Gardner, 1997). The evolution of education in STEM for SWID could lie in the use of Universal Design for Learning (UDL; CAST, 2011). Universal Design for Learning is a framework for teaching and learning, focused on the unique differences amongst students. Students do not always have the same strengths, the same skills, or the same abilities, as evidenced in this study. The introduction of UDL to teaching provides students opportunities to engage in learning and demonstrate their knowledge according to their strengths and interests. Coding and robotics are options for students to practice STEM-related skills through a UDL construct, including active learning, concrete and abstract materials, and demonstration of their knowledge using a robot to follow code (Flores et al., 2014; Israel et al., 2015; Lott et al., 2013; Lottero-Perdue et al., 2010). Universal Design for Learning could be a feasible way for teachers to implement programming/coding in early elementary classrooms and include all students.

Teachers are presented with daily challenges including time, collaboration, and lesson planning. Embedding a robotics curriculum into students' school lessons may seem a daunting task, hindered by worldly constructs of time and money. One short-term goal of launching a robotics curriculum in elementary school is collaborating with teachers to develop opportunities

to embed the curriculum in other lessons. Mathematics is perhaps an obvious subject area to include robotics, as students can program the robot and learn about directionality, problem solving, real-world problems, collaboration, and engineering design (Nickels, 2014). Students also may benefit from using programming/coding in language arts, including sequencing tasks and letting the robot tell a story through actions and movement (Kazakoff & Bers, 2012). Many students also are afforded time in a computer lab, which could be used to work with robots or online coding programs (e.g., scratch, code.org). A medium-ranged goal may include working with school districts to purchase robots and technology to be used in elementary classrooms. Students could work in small groups to solve problems during lessons or learning stations. A long-range goal could be to work with students throughout elementary school with and without disabilities in learning about technologies that will undoubtedly drive careers (Vilorio, 2014), while also help present active learning, engagement, intrinsic satisfaction, and problem solving skills (Nickels, 2014). Students learning the ability to generalize their learning from programming/coding tasks to real world problems (Miller et al., 2015) may take time and money, but the effects could be invaluable.

Students with and without disabilities, as early as preschool, have been successfully taught basic skills in computer programming/coding (Bers et al., 2014; Kazakoff & Bers, 2012; Taylor et al., 2017). When designing lessons including robotics and coding, the curriculum can be embedded in the engineering design of ask, imagine, plan, create, and improve (Lottero-Perdue, Lovelidge, & Bowling, 2010; Nickels, 2014). Engineering design provides students the opportunity to actively engage with the robot, learn the pieces of coding through games, and learn to position the blocks through trial and error while learning concepts and skills in STEM areas. For students needing extra guidance, scaffolding explicit instruction may be appropriate.

This mode will allow the teacher to give directions, to the extent necessary, for the student they are working with, while allowing the student to try and apply what they learned. Currently, research related to early childhood and robotics is embedded in constructionist teaching methodology (e.g., Lye & Koh, 2014; A. Sullivan & Bers, 2013; Sullivan & Heffernan, 2016). Only one study focuses on the needs of SWID (i.e., Taylor et al., 2017), and the literature in the field of special education (Browder et al., 2012, 2008; Devlin et al., 2013) clearly shows that this population of students learn best through the EBP of explicit instruction. A comparison study of teaching programming/coding with SWID using either constructionist or explicit instruction methodologies may help researchers and teachers understand how this population of students can demonstrate their knowledge in this subject area best. Robotics and computer coding are topics related to all four areas of STEM and can be embedded into students' daily schedules and routines when students are given the support they need (Cejka et al., 2006; Sullivan & Heffernan, 2016).

Future Implications

The abilities of students in early elementary (PreK-2nd) grade levels to learn basic code were studied by researchers as early as the 1970s (Perlman, 1976), and throughout the 1980s (Papert, 1980). After a brief period of no published research in coding in the 1990s (Lye & Koh, 2014; Yelland, 1995), a resurgence of coding literature and studies emerged in the late 2000s (e.g., Bers et al., 2002; Bers et al., 2014; Taylor et al., 2017). Even with focus on STEM and robotics curriculum since the early 2000s, a limited number of studies exist that focused on students with disabilities (specifically SWID) and computer programming at the early elementary grade levels (Adams & Cook, 2013; Miller, 2009; Ratcliff & Anderson, 2011; Taylor

et al., 2017). Researchers suggest implementing STEM, robotics, and programming/coding in elementary school (Berry et al., 2010; Lottero-Perdue, Lovelidge, & Bowling, 2010) and most states already have engineering standards embedded into their curriculum (Carr et al., 2012; Nickels, 2014; NGSS Lead States, 2013). As there is only one study focused on SWID (i.e., Taylor et al., 2017), SWID can not move forward in programming/coding because researchers have not evaluated how this population of students learn best in this constructionist supported field. Research is needed to bridge the potential gap in both research and practice (Taylor et al., 2017) for SWID and how they can achieve independence in this emerging field of robotics. Researchers could examine the appropriate age to introduce basic coding to SWID and students with other disabilities, as their academic age may limit their abilities (e.g., generalization, concrete versus abstract concepts). Researchers also could focus on the “right” or most effective way to teach students with specific disabilities or with specific gaps in their ability to learn (Israel et al., 2015). For example, how is coding possible for a student with dyslexia, versus a student who is blind, versus a SWID?

Robotics and coding are most often taught through constructionism (Papert, 1980). These approaches allow students to interact with the coding materials and teach themselves through trial and error, problem solving, and questioning (Papert, 1980). In this study, a constructionist approach was not used, as researchers have clearly shown SWID benefit from explicit instruction (Agrawal & Morin, 2016; Browder et al., 2012, 2008; Spooner & Browder, 2015; Spooner & Brown, 2011). Instead, all students, including those without disabilities, were taught through the use of concrete manipulatives (e.g., Agrawal & Morin, 2016; Flores et al., 2014) and explicit instruction supported through scaffolding, student practice, and consistent feedback (Doabler & Fien, 2013). This type of instruction is best practice for SWID, but it also

appeared to be effective for general education SWOD. Future researchers might consider comparing the use of constructionist approaches and explicit instruction in robotics and programming, with SWID and SWOD at early ages, to determine the student's best learning method. A large scale study may show differences in learning through various types of instruction, including abilities to generalize coding skills to novel projects.

Science, technology, engineering, and mathematics are on the forefront of education for the 21st century (U.S. Department of Education, 2016; Vilorio, 2014) and prominent researchers suggest these content areas should be integrated into elementary education (Bers et al., 2002; Bers, 2010; Devlin et al., 2013). Robotics and programming/coding focus on all four areas of STEM, incorporating problem solving, critical thinking, sequencing, and design tasks through active learning and engagement (Geist, 2016; Sullivan & Heffernan, 2016). Currently, systems and software are in place to help students of young ages access programming (e.g., WonderWorkshop Dash, Lego WeDo, Scratch, www.code.org). A next step may be to introduce coding into elementary curriculum, allowing students access to physical robots, tangible code, and the overlap of STEM subjects (Geist, 2016).

Current laws and governmental policies mandate involvement of all students, regardless of disabilities (ESSA, 2015; U.S. Department of Education, 2016). Science, technology, engineering, and mathematics education can be presented using robotics and programming/coding for all students at PreK-1st grades (Sullivan & Bers, 2016; Taylor et al., 2017). Sullivan and Bers (2016) researched implementation of a robotics curriculum for PreK-2nd grade students. The reserachers found all grade levels progressed in their understanding of robotics, with older students (1st and 2nd grade) understanding and implementing the coding strategies faster than the younger students (PreK and kindergarten). Students as young as PreK

can access robotics and programming technology (Bers, 2010), but the right systems (e.g., symbolic code rather than text-based code) may need to be in place to meet their learning level.

Physical coding blocks were used in this study as concrete objects for SWID to understand the abstract construction of programs through code. Young students with and without disabilities may benefit from coding blocks with symbols rather than text (e.g., Cherp) to identify the correct code and support reading skills (Bers et al., 2002). Researchers suggest SWID may benefit from learning sight words rather than individual letters and sounds because their visuo-spatial skills are stronger than their auditory skills, and they recognize the visual shape of the word, rather than the sounds it creates (Abbeduto, Warren, & Conners, 2007). Touchscreen devices support coding software, like Blockly used in this study, but only represent coding pieces through a heavily text-based system. Manufacturers may want to consider adding symbol-based code to their applications to foster the learning of younger students.

Physical coding blocks that can be used to program a robot without the researcher or teacher intervention to enter the code in a touchscreen or computer interface also may need to be developed. Currently, Fisher Price has developed a robot caterpillar (i.e., Code-a-pillar™) that allows children to build code by connecting pieces with symbols. Primo developed a robot called Cubetto, focused on teaching students as young as three-years-old to code without need of a tablet or computer (no research was found involving either of these robots). Young students are target populations for learning technology and material in programming and coding, which may prepare them for college and careers (Whitehouse Office of Science and Technology Policy, 2015).

Both formal and informal education provides students with a foundation for future learning, preparation in careers, and groundwork in life skills. The advancement of technology,

specifically in computers, has increased exponentially in the past 50 years, and will continue to advance over the lifetime of students currently in early elementary school (Cassidy et al., 2014; Rücker & Pinkwart, 2016). Students should be prepared to access and apply STEM constructs as a means to secure and advance in careers, as well as navigate day-to-day life (National Media Consortium, 2017; Vilorio, 2014; Whitehouse Office of Science and Technology Policy, 2015). Robotics and computer programming/coding are technologies with potential to help all students develop skills in problem solving and other cognitive functions that may affect the rest of their lives.

Limitations

Results from this single subject study should be interpreted with caution, as only three participants with ID and six participants without disabilities were involved. Future studies should focus on larger groups, comparison groups between explicit and constructivist instruction for SWOD, and inclusion of participants with other disabilities (e.g., ASD). Currently, most empirical studies in robotics focus on SWOD (Lye & Koh, 2014; Sullivan & Heffernan, 2016; Taylor et al., 2017)

All participants were taught skills in basic coding through 1:1 instruction in either a small, school classroom or in a home office in the afternoon. The difference of setting is a limitation in terms of variability (e.g., student comfort levels, distractions from objects or family members). Due to the location of students, the researcher accommodated for the families, including travel time and time of day.

The researcher may have been a limitation in this study, as the lead author conducted all student sessions. Fidelity of implementation was calculated on 33% of sessions (total steps

completed accurately by total number of implemented x 100) to address this limitation. Fidelity was found to be high percentage for both groups (Students with ID= 97.14%; SWOD= 98.45%; Combined percentage= 97.99%).

At the onset of this study, parents were advised to refrain from discussing the coding intervention with their children, as well as to keep coding software and games inaccessible until the end of the study. The purpose was to keep additional variables from possibly altering the students' knowledge in robotics and coding. One set of parents of a SWOD were computer scientists, so their background could have affected the participant. While they claimed they did not ask about the intervention or introduce any variables during the study, their knowledge of computers and programming may have influenced their child prior to the study or in every day conversation.

School personnel and parents also were asked if the participants had any delays in receptive language. Any participant unable to hear or understand spoken language would be at a severe disadvantage in this study, as directions were given through verbal explicit instruction. While no participants were identified as delayed in receptive language, the potential of students inability to process language is still a limitation. Some students with intellectual disabilities have language delays and require speech/language therapy (Martin, Klusek, Estigarribia, & Roberts, 2009), so if the students did not understand the directions that may be why they did not demonstrate generalization of coding in this study.

Another limitation includes students' abilities in reading, mathematics, and sequencing/problem solving. Students were similar in mathematics content knowledge, but the assessment used, KeyMath3, was not developed for students younger than kindergarten. While only one participant fell outside of this grade range, all students with ID and without disabilities

scored similar on the test. Students' ability in reading may have affected acquisition of knowledge regarding the coding blocks. Future researchers may want to implement a reading assessment before the study begins.

All concrete manipulatives were created to be similar to the iPad application Blockly, but pictures were added to aide student understanding. To account for students' inability to read the text and understanding it (e.g., words like "Start", "Forward", "Turn Left"), the researcher included picture context clues on the physical manipulatives (e.g., Start was green, Forward had a forward arrow). Differences and understanding of the iPad application may be attributed to students' ability or inability to read the text, as the application did not have picture clues.

Further, accessing the iPad posed problems to all of the SWID, particularly P1 and P3. Likely, due to underdeveloped fine motor skills (Hartman, Houwen, Scherder, & Visscher, 2010), the students showed difficulty accessing the iPad application. The application was designed for the user to only use one finger to move the coding blocks in the program, but the SWID (and some without disabilities) often tried using two fingers, or placed their opposite hand on the iPad, as if they were holding a piece of paper. The students were visibly frustrated, which may account for mistakes in producing correct code. Apple's iPads and other touchscreen devices have accessibility features for using the devices. While they were not introduced in this study, the features may benefit SWID in future research. Students also may benefit from support of an occupational therapist to guide learning and functional use of touchscreen technology. Some SWOD also had trouble accessing the iPad application. The students often wanted to use more than one finger to move the Blockly pieces, but doing so caused the application to malfunction.

A maintenance session was not utilized in this study to record student learning without researcher intervention after Treatment Phase B. Originally, this study was developed with two treatment phases (B and C). The first treatment phase (B) was the same as this study, using physical manipulatives to teach coding. The second treatment phase (C) was developed to have participants transfer skills developed in Treatment Phase B to the iPad application used in coding. Students with and without disabilities automatically generalized their understanding from Treatment Phase B to the iPad, recognizing all pieces needed to make the Dash robot go in a square on the iPad (i.e., Start, Forward, Turn Left). The researcher decided to alter Treatment Phase C into a generalization phase to focus on the students' ability to transfer skills from concrete blocks to the abstract iPad application (Flores et al., 2014). As discussed earlier, SWID were unable to generalize to independently code the robot to travel in a square using the iPad. Future researchers may want to consider focusing on students' abilities to use only the iPad application without Treatment Phase B, as well as assessing how to help students with ID transfer their skills to independently code.

Conclusion

This research study furthered the explorations of past researchers (e.g., Bers et al., 2002; Bers, 2010; Taylor et al., 2017), working with both SWID and SWOD, to demonstrate abilities in basic coding and programming at a very young age. All students should be prepared to enter careers upon completion of high school or higher education. Science, technology, engineering, and mathematics skills are interwoven into many careers today and are projected to be the forefront of future employability (National Media Consortium, 2017; Vilorio, 2014). Students with ID (and other disabilities) may not have opportunities for employment in STEM careers

unless further research is conducted in this area. Students with ID are diagnosed due to low intellectual and adaptive skills (American Psychiatric Association, 2013), which is an area coding and robotics is focused on helping students develop (Lye & Koh, 2014; Sullivan & Heffernan, 2016). While all SWID may not want to enter a STEM-focused field of study, they could benefit from learning skills in these curricula areas to help with science inquiry, problem solving, engineering, and active learning opportunities (Clements & Sarama, 2008; Devlin et al., 2013; DiFrancesca et al., 2014; Nadelson et al., 2013). The outcome of more effective problem-solving skills could lead to stronger, future outcomes both in life and in careers for this often unemployed or underemployed population of students (Goharpey et al., 2013; Scruggs & Mastropieri, 1997).

For too long, children with disabilities have been restricted in their education and learning due to diagnoses and preconceived notions. Without question, parents, researchers, and teachers have fought for the rights and inclusion of students with disabilities (e.g., Aldrich, 1932; *Brown v. Board of Education*, 1954; *Wyatt v. Stickney*, 1972; Dunn, 1968; Huey, 1913; Spooner & Brown, 2011). In the 1950s, *Brown v. Board of Education* spurred parents to fight for the inclusion and equal opportunity of their children with disabilities. In 1975, the Education of all Handicapped Children Act brought forth rights for students with disabilities in public education that altered the role of teaching and preparing youth. Throughout the last 20 plus years, educational laws and acts (e.g., ESSA, 2015; IDEA, 2004) have broken down barriers for all students, regardless of ability or disability, to access a free and appropriate education (Yudin & Musgrove, 2015). Yet, with all these gains and positive movements in education, there remains a stigma towards students with disabilities, especially those with ID. A stigma is emerging against introducing STEM concepts in the early elementary grades, which has a far-reaching

implication for future education, college, and career options for all students (Cooper et al., 2015; Lye & Koh, 2014).

The time is now to disrupt the educational system in positive ways that cause teachers, administrators, and legislators to rethink how we teach children, both with and without disabilities (National Media Consortium, 2017). The time is now to embed problem-solving curricula, like robotics, into early childhood to spur students' desires to engage with lessons, make mistakes, and try again (National Media Consortium, 2017). The time is now to give students with and without disabilities in early elementary grades the chance to demonstrate what they can accomplish in STEM areas of coding/robotics to set the stage for use of this skill in life. This emphasis in STEM curricula areas needs to include all students, including those identified with ID.

APPENDIX A:
TREATMENT PHASE B AND GENERALIZATION PHASE SCORE SHEETS AND
TREATMENT PHASE B FIDELITY OF IMPLEMENTATION

Treatment Phase B: Researcher and Inter-Rater Score Sheets

Student ID: _____ Date: _____ Inter-observer: YES NO

Baseline	Direction	Completes	Yes	No
Student tries to tell Dash to go in a square without any teacher interference or guidance	“Do you think you can put these coding blocks in order to tell Dash to move in a square?”			
Teacher explains to student they want to move the robot, Dash, in a square, like the one on the ground. To do that they must talk to the robot using the coding blocks.	“I can’t help you, try your best.”	Puts “Start” first <i>If student does not put start first, mark all as “no” and score “0”</i>		
<i>Student must put “Start” first to receive any score above “0”.</i>		Puts “Forward”		
		Puts “Turn Left”		
		Puts “Forward”		
		Puts “Turn Left”		
		Puts “Forward”		
		Puts “Turn Left”		
		Puts “Forward”		
		Puts “End”		
		Total:		

Student ID: _____ **Date:** _____ **Inter-observer:** YES NO
 Please mark “Yes” or “No” for students response to question posed.

Goal 1	Direction	Completes	Yes	No
1. Teach student to make Dash go forward in a straight line	a. Teacher: Which one do you think is “Go”? <i>(Show “Go” and “Stop”)</i> <i>(If wrong, show the right one, repeat direction)</i>	Identifies Go		
Researcher will teach student how to make Dash go in a straight line.	b. Teacher: Which one do you think is “stop”? <i>(Show “Stop” and “Forward”)</i>	Identifies Stop		
Will present student with statements about what they are doing today, then ask questions followed by statements regarding making Dash go in a line.	c. Teacher: Which one do you think is “Forward”? <i>(Show “Forward” and “Turn”)</i>	Identifies Forward		
	d. Teacher: Show me Go. Add Forward. Add stop. <i>(If wrong, show right one, repeat direction)</i>	Makes code: “Go, Forward, Stop”		
<i>Repeat Directions until Participant completes with maximum 1 wrong over in a session over three consecutive sessions, then continue to Step 2</i>		Total:		
		Total percent correct: Total Yes/ (Total Yes + Total No)		/25
Students given opportunity to code a square without teacher intervention. <i>Count as yes as long as they make that part of the square</i>		1/2		
		3/4		
		Full		

Student ID: _____ Date: _____ Inter-observer: YES NO

Please mark “Yes” or “No” for students response to question posed.

Goal 2	Question/Direction	Completes	Yes	No
2. Teach participant to make Dash go in ½ of a square. Dash will go forward in a straight line, turn left, go in a straight line	a. Teacher: How do we always start our code? <i>Go</i>	Says or points to Go		
	b. Teacher: Which one is “Go?” <i>(Show “Go, Stop”)</i>	Identifies Go		
	c. Teacher: How do we always end our code? <i>Stop</i>	Says or points to Stop		
	d. Teacher: Which one is “Stop?” <i>(Show “Stop, Forward”)</i>	Identifies Stop		
	e. Teacher: Which one is “Forward?” <i>(Show “Forward, Turn Left”)</i>	Identifies Forward		
	f. Teacher: Which one do you think is “Turn?” <i>(Show “Turn Left, Repeat”)</i>	Identifies Turn Left		
	We are going to make Dash go forward, turn left, and go forward again.	g. Teacher: Show me Go. Add forward. Add Turn left. Add another turn forward. Add stop.	Makes code: “Go, Forward, Turn Left, Forward, Stop”	
Repeat Directions until Participant completes with maximum 1 wrong over in a session over three consecutive sessions, then continue to Step 3		Total:		
		Total percent correct: Total Yes/ (Total Yes +Total No)	/25	
Students given opportunity to code a square without teacher intervention. <i>Count as yes as long as they make that part of the square</i>		3/4		
		Full		

Student ID: _____ Date: _____ Inter-observer: YES NO

Please mark “Yes” or “No” for students response to question posed.

Session Date:				
Goal 3	Question/Direction	Completes	Yes	No
3. Teach participant to make Dash go in $\frac{3}{4}$ of a square. Dash will go forward in a straight line, turn left, go in a straight line, turn left go in a straight line	a. Teacher: How do we always start our code? <i>Go</i>	Says or points to Go		
	b. Teacher: Which one is “Go?” <i>(Show “Go, Stop, Forward”)</i>	Identifies Go		
	c. Teacher: How do we always end our code? <i>Stop</i>	Says or points to Stop		
	d. Teacher: Which one is “Stop?” <i>(Show “Stop, Forward, Turn Left”)</i>	Identifies Stop		
	e. Teacher: Which one is “Forward?” <i>(Show “Forward, Turn Left, Turn Right”)</i>	Identifies Forward		
	f. Teacher: Which one is “Turn?” <i>(Show “Turn Left, Repeat”)</i>	Identifies Turn		
	We are going to make Dash go forward, turn left, go forward again, turn left, and go forward one more time.	g. Teacher: Show me Go. Add forward. Add Turn left. Add another forward. Add another turn left. Add another forward. Add Stop.	Makes code: “Go, forward, turn left, forward, turn left, forward, stop”	
Repeat Directions until Participant completes with maximum 1 wrong over in a session over three consecutive sessions, then continue to Step 4		Total:		
		Total percent correct: Total Yes/ (Total Yes +Total No)	/25	
Students given opportunity to code a square without teacher intervention. <i>Count as yes as long as they make that part of the square</i>		Full		

Student ID: _____ Date: _____ Inter-observer: YES NO
 Please mark “Yes” or “No” for students response to question posed.

Session Date:				
Goal 4	Question/Direction	Completes	Yes	No
4. Teach participant to make Dash go in a full square. Dash will go forward in a straight line, turn left, go in a straight line, turn left, go in a straight line, turn left, and go in a straight line.	a. Teacher: How do we always start our code? <i>Go</i>	Says or points to Go		
	b. Teacher: Which one is “Go?” <i>(Show “Go, Stop, Forward”)</i>	Identifies Go		
	c. Teacher: How do we always end our code? <i>Stop</i>	Says or points to Stop		
	d. Teacher: Which one is “Stop?” <i>(Show “Stop, Forward, Turn Left”)</i>	Identifies Stop		
	e. Teacher: Which one is “Forward?” <i>(Show “Forward, Turn Left, Turn Right”)</i>	Identifies Forward		
	f. Teacher: Which one is “Turn?” <i>(Show “Turn Left, Repeat”)</i>	Identifies Turn		
	We are going to make dash go in a whole square. Can you walk in a square? We will have dash go forward, turn left, go forward again, turn left again, go forward again, turn left again, and go forward and turn left again.	g. Teacher: Show me Go. Add forward. Add Turn left. Add another forward. Add another turn left. Add another forward. Add another turn left. Add another forward. Add Stop.	Makes code: “Go, forward, turn left, forward, turn left, forward, turn left, forward, stop”	
		Total:		
		Total percent correct: Total Yes/ (Total possible)	/25	

Generalization Phase C: Researcher and Inter-Rater Score Sheets

Student ID: _____ Date: _____ Inter-observer: YES NO

Please mark “Yes” or “No” for students response to question posed.

Goal 1	Direction	Completes	Yes	No
1. Teach student to make Dash go forward in a straight line using Blockly App	<i>a. Do you think you can make Dash move in a square using the iPad without my help?</i>	1/4 of square		
Researcher will teach student how to make Dash go in a straight line.		1/2 of square		
Will then give student opportunity to independently code Dash to move in a square using the Blockly App.		3/4 of square		
		Full square		
	Total percent correct	1/4 of square	25%	
		1/2 of square	50%	
		3/4 of square	75%	
		Full square	100%	

Treatment Phase B: Fidelity of Implementation
 Fidelity of Implementation
 Programming with KIDS

Inter-rater Initials: _____ Session: _____ Date: _____

Goal 1: Teach student to make Dash go forward in a straight line			
Step	What teacher does	Verbal Direction	Completed
1	Teacher tells student goal of the session	Today we are going to tell dash to go forward in a straight line. Are you ready?	
2	Teacher tells student how to start talking to Dash and shows block	Look at this block. This block says 'Go'. We always have to start by telling Dash to 'Go'. Can you point at 'Go'?	
3	Teacher shows student blocks 'Go' and 'Stop'	Which one do you think is 'Go'? <i>(If student said and pointed to 'Go' in previous direction, mark as correct)</i>	
4	Teacher tells student how to stop Dash and shows block	We always have to tell Dash to stop when we are done. Look at this block. This block says 'Stop.' Can you point at 'Stop'?	
5	Teacher shows student blocks 'Stop' and 'Forward'	Which one do you think is 'Stop'? <i>(If student said and pointed to 'Stop' in previous direction, mark as correct)</i>	
6	Teacher tells student we need to tell Dash to move forward	This block says "Forward". See the arrow showing "Forward"? Can you point at "Forward"?	
7	Teacher shows student blocks 'Forward' and 'Turn Left'	Which one do you think is "Forward"?	
8	Teacher tells student that we will now tell Dash to go forward and stop. Shows student all pieces used during session.	Now we are going to tell Dash to go forward in a straight line and then stop.	
9	Shows student all pieces used during session.	Can you find 'Go'?	
10	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	
11	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Stop'?	

12	<i>If needed:</i> Teacher and student read line of code together.	Let's read the blocks together. Go, Forward, Stop.	
----	--	---	--

Inter-rater Initials: _____ Session: _____ Date: _____

Goal 2: Teach participant to make Dash go in ½ of a square. Dash will go forward in a straight line, turn left, go in a straight line			
Step	What teacher does	Verbal Direction	Completed
1	Teacher tells student goal of the session	Today we are going to make Dash go forward, turn left, and go forward again. Are you ready?	
2	Teacher asks student how we always start talking to Dash while showing block.	How do we always start our code? Can you point at the block or say its name?	
3	Teacher shows student blocks 'Go' and 'Stop'	Which one is 'Go'? <i>(If student said and pointed to 'Go' in previous direction, mark as correct)</i>	
4	Teacher asks student how we always end talking to Dash and shows block.	How do we always end talking to Dash? Can you point at the block or say its name?	
5	Teacher shows student blocks 'Stop' and 'Forward'	Which one is 'Stop'? <i>(If student said and pointed to 'Stop' in previous direction, mark as correct)</i>	
6	Teacher tells student we need to tell Dash to move forward. Teacher shows student blocks 'Forward' and 'Turn Left'	Do you remember which block says "Forward"?	
7	Teacher tells student we need to tell Dash to turn left.	This block says "Turn Left." See the arrow turning left? Can you point at "Turn Left"?	
	Teacher shows student 'Turn Left' and another block not used (e.g., 'Repeat')	Which block is 'Turn Left'?	
8	Teacher tells student that we will now tell Dash to go forward, turn, go forward, and stop. Shows student all pieces used during session.	Now we are going to tell Dash to go forward in a straight line, turn left, go forward again, and then stop.	
9	Shows student all pieces used during session.	Can you find 'Go'?	
10	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	
11	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Turn Left'?	
12	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	

13	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Stop'?	
14	<i>If needed:</i> Teacher and student read line of code together.	Let's read the blocks together. Go, Forward, Turn Left, Forward, Stop.	

Inter-rater Initials: _____ Session: _____ Date: _____

Goal 3: Teach participant to make Dash go in $\frac{3}{4}$ of a square. Dash will go forward in a straight line, turn left, go in a straight line, turn left go in a straight line			
Step	What teacher does	Verbal Direction	Completed
1	Teacher tells student goal of the session	Today we are going to make Dash go forward, turn left, and go forward again. Are you ready?	
2	Teacher asks student how we always start talking to Dash while showing block.	How do we always start our code? Can you point at the block or say its name?	
3	Teacher shows student blocks 'Go' and 'Stop'	Which one is 'Go'? <i>(If student said and pointed to 'Go' in previous direction, mark as correct)</i>	
4	Teacher asks student how we always end talking to Dash and shows block.	How do we always end talking to Dash? Can you point at the block or say its name?	
5	Teacher shows student blocks 'Stop' and 'Forward'	Which one is 'Stop'? <i>(If student said and pointed to 'Stop' in previous direction, mark as correct)</i>	
6	Teacher tells student we need to tell Dash to move forward. Teacher shows student blocks 'Forward' and 'Turn Left'	Do you remember which block says "Forward"?	
7	Teacher tells student we need to tell Dash to turn left.	This block says "Turn Left." See the arrow turning left? Can you point at "Turn Left"?	
	Teacher shows student 'Turn Left' and another block not used (e.g., 'Repeat')	Which block is 'Turn Left'?	
8	Teacher tells student that we will now tell Dash to go forward, turn, go forward, and stop. Shows student all pieces used during session.	Now we are going to tell Dash to go forward in a straight line, turn left, go forward again, and then stop.	
9	Shows student all pieces used during session.	Can you find 'Go'?	
10	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	
11	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Turn Left'?	
12	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	
13	Shows student all pieces used during	Can you find 'Stop'?	

	session. Student interlocks pieces.		
14	<i>If needed:</i> Teacher and student read line of code together.	Let's read the blocks together. Go, Forward, Turn Left, Forward, Stop.	

Inter-rater Initials: _____ Session: _____ Date: _____

Goal 4: Teach participant to make Dash go in a full square. Dash will go forward in a straight line, turn left, go in a straight line, turn left, go in a straight line, turn left, and go in a straight line.			
Step	What teacher does	Verbal Direction	Completed
1	Teacher tells student goal of the session	Today we are going to make Dash go forward, turn left, and go forward again. Are you ready?	
2	Teacher asks student how we always start talking to Dash while showing block.	How do we always start our code? Can you point at the block or say its name?	
3	Teacher shows student blocks 'Go' and 'Stop'	Which one is 'Go'? <i>(If student said and pointed to 'Go' in previous direction, mark as correct)</i>	
4	Teacher asks student how we always end talking to Dash and shows block.	How do we always end talking to Dash? Can you point at the block or say its name?	
5	Teacher shows student blocks 'Stop' and 'Forward'	Which one is 'Stop'? <i>(If student said and pointed to 'Stop' in previous direction, mark as correct)</i>	
6	Teacher tells student we need to tell Dash to move forward. Teacher shows student blocks 'Forward' and 'Turn Left'	Do you remember which block says "Forward"?	
7	Teacher tells student we need to tell Dash to turn left.	This block says "Turn Left." See the arrow turning left? Can you point at "Turn Left"?	
	Teacher shows student 'Turn Left' and another block not used (e.g., 'Repeat')	Which block is 'Turn Left'?	
8	Teacher tells student that we will now tell Dash to go forward, turn, go forward, and stop. Shows student all pieces used during session.	Now we are going to tell Dash to go forward in a straight line, turn left, go forward again, and then stop.	
9	Shows student all pieces used during session.	Can you find 'Go'?	
10	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	
11	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Turn Left'?	
12	Shows student all pieces used during	Can you find 'Forward'?	

	session. Student interlocks pieces.		
13	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Turn Left'?	
14	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	
15	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Turn Left'?	
16	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Forward'?	
17	Shows student all pieces used during session. Student interlocks pieces.	Can you find 'Stop'?	
18	<i>If needed:</i> Teacher and student read line of code together.	Let's read the blocks together. Go, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Turn Left, Forward, Stop.	

Generalization Phase C: Fidelity of Implementation
 Fidelity of Implementation
 Programming with KIDS

Inter-rater Initials: _____ *Session:* _____ *Date:* _____

Generalization Goal: Students use iPad application “Blockly”; code Dash to move in a square independently			
Step	What teacher does	Verbal Direction	Completed
1	Teacher tells student goal of the session	Today we are going to tell dash to go forward in a straight line then I’ll let you move him in a square.	
2	Teacher tells student how to start talking to Dash and shows coding block on Blockly application	Look at this block. This block says ‘Start’, which means ‘Go’. We always have to start by telling Dash to ‘Start’ or ‘Go’. Can you point at ‘Start’?	
3	Teacher tells student we need to tell Dash to move forward	This block says “Forward”. Can you point at “Forward”?	
4	Teacher shows student blocks ‘Forward’ and another block (e.g., make a sound)	Which one do you think is “Forward”?	
5	Teacher tells student that we will now tell Dash to go forward and stop. Shows student all pieces used during session.	Now we are going to tell Dash to go forward in a straight line and then stop.	
6	Shows student all pieces used during session.	Can you point at go ‘Go’?	
7	Shows student all pieces used during session. Student interlocks pieces.	Can you find ‘Forward’ and add it to ‘Go’?	
8	Student starts program	Can you press the start button at the bottom of the screen?	
9	Student independently codes Dash to move	Now, without my help, do you think you can make Dash move in a full square?	

APPENDIX B:
SOCIAL VALIDITY SURVEYS (PARTICIPANTS, PARENTS,
TEACHERS/ADMINISTRATORS)

Participant Survey

Participant Name: _____

Please circle the corresponding number that best represents your agreement with the statement using the key below:

Disagree

Neither Agree nor Disagree

Agree



Survey

Statement	Scale		
1. I enjoyed working with the Dash robot			
2. I understood the directions Mr. Matt gave me			
3. I want to continue working with the Dash robot			

Parent Survey

Participant Name: _____

Please complete the five-question survey below regarding the study “Programing with Kindergarten Students with Intellectual Disabilities and Students without Disabilities” your child participated in during the Fall of 2016/Spring 2017. The term “Intervention” refers to the explicit instruction in programing skills. The term “behavior” refers to the computer coding completed by your child.

Please circle the corresponding number that best represents you agreement with the statement using the key below:

Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
1	2	3	4	5

Survey

Statement	Scale
1. The intervention (explicit instruction in programming) focused on an important behavior (computer coding).	1 2 3 4 5
2. Procedures for intervention (explicit instruction in programming) were explained to me.	1 2 3 4 5
3. I feel that my child learned a new skill through this intervention (explicit instruction in programming)	1 2 3 4 5
4. I would recommend this intervention (explicit instruction in programming) to parents of children with or without disabilities.	1 2 3 4 5
5. I would like to continue working on this behavior (computer coding) with my child.	1 2 3 4 5

Teacher/Administrator Survey

Your Title: _____

Please complete the five-question survey below regarding the study “Programing with Kindergarten Students with Intellectual Disabilities and Students without Disabilities” your students(s) participated in during the Fall of 2016/Spring 2017. The term “Intervention” refers to the explicit instruction in programing skills. The term “behavior” refers to the computer coding completed by your child.

Please circle the corresponding number that best represents you agreement with the statement using the key below:

Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
1	2	3	4	5

Survey

Statement	Scale
1. The intervention (explicit instruction in programming) focused on an important behavior (computer coding).	1 2 3 4 5
2. Procedures for intervention (explicit instruction in programming) were explained to me.	1 2 3 4 5
3. I feel that my students learned a new skill through this intervention (explicit instruction in programming)	1 2 3 4 5
4. I would recommend this intervention (explicit instruction in programming) to teachers of children with or without disabilities.	1 2 3 4 5
5. I would like to continue working on this behavior (computer coding) with students or in my school.	1 2 3 4 5

APPENDIX C:
INSTITUTIONAL REVIEW BOARD CONSENT



University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Human Research

From: **UCF Institutional Review Board #1
FWA00000351, IRB00001138**

To: **Matthew S Taylor and Co-PIs Eleazar Vasquez, Lisa A Dieker, and Megan Nickels**

Date: **December 05, 2016**

Dear Researcher:

On 12/05/2016 the IRB approved the following human participant research until 12/04/2017 inclusive:

Type of Review: IRB Continuing Review Application Form
Expedited Review Category

Project Title: Programming with Kindergarten Students with and without
Intellectual Disabilities

Investigator: Matthew S Taylor

IRB Number: SBE-15-11845

Research ID: 4270897

The scientific merit of the research was considered during the IRB review. The Continuing Review Application must be submitted 30 days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form **cannot** be used to extend the approval period of a study. All forms may be completed and submitted online at <https://iris.research.ucf.edu>.

If continuing review approval is not granted before the expiration date of 12/04/2017, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

All data, including signed consent forms if applicable, must be retained and secured per protocol for a minimum of five years (six if HIPAA applies) past the completion of this research. Any links to the identification of participants should be maintained and secured per protocol. Additional requirements may be imposed by your funding agency, your department, or other entities. Access to data is limited to authorized individuals listed as key study personnel.

In the conduct of this research, you are responsible to follow the requirements of the [Investigator Manual](#).

On behalf of Sophia Dziegielewski, Ph.D., L.C.S.W., UCF IRB Chair, this letter is signed by:

IRB Coordinator

REFERENCES

- Abbeduto, L., Warren, S. F., & Conners, F. A. (2007). Language development in Down syndrome : From the prelinguistic period to the acquisition of literacy. *Mental Retardation and Developmental Disabilities Research Reviews*, *13*(3), 247–261.
<https://doi.org/http://dx.doi.org/10.1002/mrdd.20158>
- Adams, K. D., & Cook, A. M. (2013). Programming and controlling robots using scanning on a speech generating communication device: A case study. *Technology & Disability*, *25*(4), 275–286. <https://doi.org/10.3233/TAD-140397>
- Agran, M., & Hughes, C. (2005). Introduction to special issue: Self-determination reexamined are people with severe disabilities any more self-determined? Introduction to the special issue on self-determination: how far have we come? *Research & Practice for Persons with Severe Disabilities*, *30*(3), 105–107.
- Agrawal, J., & Morin, L. L. (2016). Evidence-based practices: Applications of concrete representational abstract framework across math concepts for students with mathematics disabilities. *Learning Disabilities Research & Practice*, *31*(1), 34–44.
<https://doi.org/10.1111/ldrp.12093>
- Aldrich, C. G. (1932). Lessons in child training gleaned from idiots. *Child Development*, *3*, 75–80.
- American Psychiatric Association. (2013). *Diagnostic and statistical manual of mental disorders (5th ed.)*. Arlington, VA: American Psychiatric Publishing.
- Atkinson, B. (1984). Learning disabled students and LOGO. *Journal of Learning Disabilities*, *17*, 500–501. <https://doi.org/10.1177/002221948401700812>

- Barker, B. S., & Ansorge, J. (2007). Robotics as means to increase achievement scores in an informal learning environment. *Journal of Research on Technology in Education*, 39(3), 229–243. <https://doi.org/10.1080/15391523.2007.10782481>
- Bartolini Bussi, M. G., & Baccaglini-Frank, A. (2015). Geometry in early years: Sowing seeds for a mathematical definition of squares and rectangles. *ZDM: The International Journal on Mathematics Education*, 47(3), 391–405.
- Basham, J. D., Israel, M., & Maynard, K. (2010). An ecological model of STEM education: Operationalizing STEM for all. *Journal of Special Education Technology*, 25(3), 9–19.
- Basham, J. D., & Marino, M. T. (2013). Understanding STEM education and supporting students through universal design for learning. *Teaching Exceptional Children*, 45(4), 8–15.
- Bass, J. E. (1985). The roots of Logo's education theory: An analysis. *Computers in Schools*, 2(2–3), 107–116. https://doi.org/10.1300/J025v02n02_13
- Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3), 978–988.
<https://doi.org/10.1016/j.compedu.2011.10.006>
- Berry, C. A., Remy, S. L., & Rogers, T. E. (2016). Robotics for all ages: A standard robotics curriculum for K-16. *IEEE Robotics & Automation Magazine*, 23(2), 40–46.
<https://doi.org/10.1109/MRA.2016.2534240>
- Berry, R. Q. I., Bull, G., Browning, C., Thomas, C. D., Starkweather, K., & Aylor, J. H. (2010). Preliminary considerations regarding use of digital fabrication to incorporate engineering design principles in elementary mathematics education. *Contemporary Issues in Technology & Teacher Education*, 10(2), 167–172.

- Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12*(2), 1–20.
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & 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.
- Browder, D. M., & Cooper-Duffy, K. (2003). Evidence-based practices for students with severe disabilities and the requirement for accountability in “No Child Left Behind.” *Journal of Special Education, 37*(3), 157–163.
- Browder, D. M., Spooner, F., Ahlgrim-Delzell, L., Harris, A. A., & Wakeman, S. (2008). A meta-analysis on teaching mathematics to students with significant cognitive disabilities. *Exceptional Children, 74*(4), 407–432. <https://doi.org/10.1177/001440290807400401>
- Browder, D. M., Trela, K., Courtade, G. R., Jimenez, B. A., Knight, V., & Flowers, C. (2012). Teaching mathematics and science standards to students with moderate and severe developmental disabilities. *Journal of Special Education, 46*(1), 26–35. <https://doi.org/10.1177/0022466910369942>
- Brown, L., Branston, M. B., Hamre-Nietupski, S., Pumpian, I., Certo, N., & Gruenewald, L. (1979). A strategy for developing chronological-age-appropriate and functional curricular content for severely handicapped adolescents and young adults. *Journal of Special Education, 13*(1), 81–90.
- Brown v. Board of Education, No. 347 U.S. 483 (1954).

- Carr, R. L., Bennett IV, L. D., & Strobel, J. (2012). Engineering in the K-12 STEM standards of the 50 U.S. States: An Analysis of presence and extent. *Journal of Engineering Education, 101*(3), 539–564. <https://doi.org/10.1002/j.2168-9830.2012.tb00061.x>
- Cassidy, E. D., Colmenares, A., Jones, G., Manolovitz, T., Shen, L., & Vieira, S. (2014). Higher education and emerging technologies: Shifting trends in student usage. *The Journal of Academic Librarianship, 40*(2), 124–133. <https://doi.org/10.1016/j.acalib.2014.02.003>
- CAST. (2011). *Universal Design for Learning Guidelines version 2.0*. Wakefield, MA: Author.
- 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.
- Chiang, B., Thorpe, H. W., & Lubke, M. (1984). LD students tackle the LOGO language: Strategies and implications. *Journal of Learning Disabilities, 17*(5), 303–304. <https://doi.org/10.1177/002221948401700513>
- Clements, D. H., & Sarama, J. (2008). Focal points—Pre-K to kindergarten. *Teaching Children Mathematics, 14*(6), 361–365.
- Clements, D. H., & Sarama, J. (2011). Early childhood mathematics intervention. *Science, 333*(6045), 968–970. <https://doi.org/10.1126/science.1204537>
- Cooper, M. M., Caballero, M. D., Ebert-May, D., Fata-Hartley, C. L., Jardeleza, S. E., Krajcik, J. S., ... Underwood, S. M. (2015). Challenge faculty to transform STEM learning. *Science, 350*(6258), 281–282. <https://doi.org/10.1126/science.aab0933>
- Cote, D., Pierce, T., Higgins, K., Miller, S., Tandy, R., & Sparks, S. (2010). Increasing skill performances of problem solving in students with intellectual disabilities. *Education & Training in Autism & Developmental Disabilities, 45*(4), 512–524.

- Devlin, T. J., Feldhaus, C. R., & Bentrem, K. M. (2013). The evolving classroom: A study of traditional and technology-based instruction in a STEM classroom. *Journal of Technology Education, 25*(1), 34–54. <https://doi.org/10.21061/jte.v25i1.a.3>
- DiFrancesca, D., Lee, C., & McIntyre, E. (2014). Where is the “E” in STEM for young children? Engineering design education in an elementary teacher preparation program. *Issues in Teacher Education, 23*(1), 49–64.
- Doabler, C. T., & Fien, H. (2013). Explicit mathematics instruction: What teachers can do for teaching students with mathematics difficulties. *Intervention in School & Clinic, 48*(5), 276–285. <https://doi.org/10.1177/1053451212473151>
- Dunn, L. (1968). Special education for the mildly retarded: Is much of it justifiable? *Exceptional Children, 35*, 5–22.
- Education For All Handicapped Children Act, Pub. L. No. 94–142 (1975).
- Every Student Succeeds Act, Pub. L. No. 114–95, § 1177 (2015). Retrieved from <https://www.gpo.gov/fdsys/pkg/BILLS-114s1177enr/pdf/BILLS-114s1177enr.pdf>
- Faulkner, V. N., Crossland, C. L., & Stiff, L. V. (2013). Predicting eighth-grade algebra students with individualized education programs. *Exceptional Children, 79*(3), 329–345.
- 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 & Education, 63*, 87–97. <https://doi.org/10.1016/j.compedu.2012.11.016>
- Flores, M. M., Hinton, V. M., Strozier, S. D., & Terry, S. L. (2014). Using the concrete-representational-abstract sequence and the strategic instruction model to teach computation to students with autism spectrum disorders and developmental disabilities. *Education and Training in Autism and Developmental Disabilities, 49*(4), 547–554.

- Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences of the United States of America*, *111*(23), 8410–8415. <https://doi.org/10.1073/pnas.1319030111>
- Gardner, H. (1997). Fostering diversity through personalized education: Implications of a new understanding of human intelligence. *Prospects*, *27*(3), 347–363. <https://doi.org/10.1007/BF02736635>
- Geist, E. (2016). Robots, programming and coding, oh my! *Childhood Education*, *92*(4), 298–304.
- Goharpey, N., Crewther, D. P., & Crewther, S. G. (2013). Problem solving ability in children with intellectual disability as measured by the Raven’s Colored Progressive Matrices. *Research in Developmental Disabilities*, *34*(12), 4366–4374. <https://doi.org/10.1016/j.ridd.2013.09.013>
- Goodnough, K., Pelech, S., & Stordy, M. (2014). Effective professional development in STEM education: The perceptions of primary/elementary teachers. *Teacher Education and Practice*, *27*(2–3), 402–423.
- Harlow, D. B., & Leak, A. E. (2014). Mapping students’ ideas to understand learning in a collaborative programming environment. *Computer Science Education*, *24*(2/3), 229–247. <https://doi.org/10.1080/08993408.2014.963360>
- Hartman, E., Houwen, S., Scherder, E., & Visscher, C. (2010). On the relationship between motor performance and executive functioning in children with intellectual disabilities. *Journal of Intellectual Disability Research*, *54*(5), 468–477. <https://doi.org/10.1111/j.1365-2788.2010.01284.x>

- Hefty, L. J. (2015). STEM gives meaning to mathematics. *Teaching Children Mathematics*, 21(7), 422–429.
- Huey, E. B. (1913). The education of defectives and the training of teachers for special classes. *The Journal of Educational Psychology*, 4(9), 545–550.
- Individuals with Disabilities Education Act, Pub. L. No. 101–476, § 104 Stat. 1142 (1990).
- Individuals with Disabilities Education Act, Pub. L. No. 108–446, § 118, Stat. 2647 (1997).
- Individuals with Disabilities Education Act (IDEA), Pub. L. No. 108–446, § 118, Stat. 2647 (2004).
- International Society for Technology in Education. (2007). *ISTE standards for students*. Retrieved from <http://www.iste.org/standards/standards/standards-for-students>
- International Society for Technology in Education. (2016). *ISTE standards: Students*. Retrieved from <http://www.iste.org/standards/standards/for-students-2016>
- Israel, M., Wherfel, Q. M., Pearson, J., Shehab, S., & Tapia, T. (2015). Empowering K–12 students with disabilities to learn computational thinking and computer programming. *Teaching Exceptional Children*, 48(1), 45–53. <https://doi.org/10.1177/0040059915594790>
- Jimenez, B. A., Browder, D. M., & Courtade, G. R. (2008). Teaching an algebraic equation to high school students with moderate developmental disabilities. *Education and Training in Developmental Disabilities*, 43(2), 266–274.
- Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, 52, 200–210. <https://doi.org/10.1016/j.chb.2015.05.047>

- Kalelioğlu, F., & Gülbahar, Y. (2014). The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective. *Programavimo Mokymo Naudojant Scratch Efektyvumas: Mokinio Perspektyva.*, 13(1), 33–50.
- Kärnä-Lin, E., Pihlainen-Bednarik, K., Sutinen, E., & Virnes, M. (2006). Can robots teach? Preliminary results on educational robotics in special education (pp. 319–321). Presented at the Advanced Learning Technologies, 2006. Sixth International Conference, Kerkade, Netherlands: IEEE.
- Karp, T., Gale, R., Lowe, L. A., Medina, V., & Beutlich, E. (2010). Generation NXT: Building young engineers with LEGOs. *IEEE Transactions on Education*, 53(1), 80–87.
- Kzakoff, E., & Bers, M. U. (2012). Programming in a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia*, 21(4), 371–391.
- Kemp, C., & Carter, M. (2006). The contribution of academic skills to the successful inclusion of children with disabilities. *Journal of Developmental and Physical Disabilities*, 18(2), 123–147.
- Kratochwill, T. R., Hithcock, J., Horner, R. H., Levin, J. R., Odom, S. L., Rindskopf, D. M., & Shadish, W. R. (2010). Single-case designs technical documentation. Retrieved from What Works Clearinghouse website: http://ies.ed.gov/ncee/wwc/pdf/wwc_scd.pdf.
- Kwon, D.-Y., Kim, H.-S., Shim, J.-K., & Lee, W.-G. (2012). Algorithmic bricks: A tangible robot programming tool for elementary school students. *IEEE Transactions on Education*, 55(4), 474–479.
- Lange, M. (1985). Using the turtle tot robot to enhance logo for the hearing impaired. *American Annals of the Deaf*, 130(5), 377–382. <https://doi.org/10.1353/aad.2012.0935>

- Lee, J., Lee, J. O., & Collins, D. (2009). Enhancing children's spatial sense using tangrams. *Childhood Education, 86*(2), 92–94. <https://doi.org/10.1080/00094056.2010.10523120>
- Lott, K., Wallin, M., Roghaar, D., & Price, T. (2013). Engineering encounters: Catch me if you can! *Science and Children, 51*(4), 65–69.
- Lottero-Perdue, P. S., Lovelidge, S., & Bowling, E. (2010). Engineering for all. *Science and Children, 47*(7), 24–27.
- Louca, L. T., Zacharia, Z. C., & Constantinou, C. P. (2011). In Quest of productive modeling-based learning discourse in elementary school science. *Journal of Research in Science Teaching, 48*(8), 919–951.
- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61. <https://doi.org/10.1016/j.chb.2014.09.012>
- Martin, G. E., Klusek, J., Estigarribia, B., & Roberts, J. E. (2009). Language characteristics of individuals with Down syndrome. *Topics in Language Disorders, 29*(2), 112–132.
- Miller, B., Doughty, T., & Krockover, G. (2015). Using science inquiry methods to promote self-determination and problem-solving skills for students with moderate intellectual disability. *Education & Training in Autism & Developmental Disabilities, 50*(3), 356–368.
- Miller, P. (2009). Learning with a missing sense: What can we learn from the interaction of a deaf child with a turtle? *American Annals of the Deaf, 154*(1), 71–82. <https://doi.org/10.1353/aad.0.0075>
- Mills v. Board of Education of District of Columbia, No. 343 F. Supp. 866 (D. C. 1972).
- Mitts, C. R. (2016). Why STEM? *Technology & Engineering Teacher, 75*(6), 30–35.

- Monari Martinez, E., & Benedetti, N. (2011). Learning mathematics in mainstream secondary schools: Experiences of students with Down's syndrome. *European Journal of Special Needs Education, 26*(4), 531–540. <https://doi.org/10.1080/08856257.2011.597179>
- Moomaw, S. (2012). STEM begins in the early years. *School Science & Mathematics, 112*(2), 57–58. <https://doi.org/10.1111/j.1949-8594.2011.00119.x>
- Nadelson, L. S., Callahan, J., Pyke, P., Hay, A., Dance, M., & Pfiester, J. (2013). Teacher STEM perception and preparation: Inquiry-based stem professional development for elementary teachers. *Journal of Educational Research, 106*(2), 157–168.
- National Council of Teachers of Mathematics. (2000). *Principles and standards of school mathematics*. Reston, VA: Author. Retrieved from <http://www.nctm.org/standards/mathematics>.
- National Education Association. (2011). *P21 common core toolkit: A guide to aligning the common core state standards with the framework for 21st century skills* (Partnership for 21st Century Skills) (pp. 1–44). Washington, DC. Retrieved from <http://www.p21.org/storage/documents/P21CommonCoreToolkit.pdf>
- National Governors Association Center for Best Practices & Council of Chief State School Officers. (2010). *Common Core State Standards for Mathematics*. Washington, DC: Authors.
- National Human Genome Research Institute. (2016, September). Learning about Down syndrome. Retrieved from <https://www.genome.gov/19517824/#3>
- National Media Consortium. (2017). *NMC/CoSN Horizon report preview: 2017 K-12 edition* (pp. 1–11). Retrieved from <https://cdn.nmc.org/media/2017-nmc-cosn-horizon-report-k12-preview.pdf>

- Newman, L., Wagner, M., Knokey, A.-M., Marder, C., Nagle, K., Shaver, D., ... Swarting, M. (2011). *The Post-High School Outcomes of Young Adults With Disabilities up to 8 Years After High School. A Report From the National Longitudinal Transition Study-2 (NLTS2)* (No. NCSER 2011-3005). Menlo Park, CA: SRI International.
- NGSS Lead States. (2013). *Next generation science standards: For States, by States*. Washington, DC: The National Academies Press.
- Nickels, M. (2014). Meaningful measurement: Addressing equity through STEM. *Wisconsin Teacher of Mathematics*, 66(1), 8–12.
- Pantoya, M. L., Aguirre-Munoz, Z., & Hunt, E. M. (2015). Developing an engineering identity in early childhood. *American Journal of Engineering Education*, 6(2), 61–68.
<https://doi.org/10.19030/ajee.v6i2.9502>
- Papert, S. (1972). Teaching children thinking. *Innovations in Education & Training International*, 9(5), 245–255. <https://doi.org/10.1080/1355800720090503>
- Papert, S. (1980). *Mindstorms: Childen, computers, and powerful ideas*. New York: Basic Books.
- Papert, S. (1985). Different visions of logo. *Computers in the Schools*, 2(2/3), 3–8.
- Parker, R. I., Vannest, K. J., & Davis, J. L. (2011). Effect size in single-case research: A review of nine nonoverlap techniques. *Behavior Modification*, 35(4), 303–322.
- Pennsylvania Association for Retarded Children v. Commonwealth of Pennsylvania, No. 334 F. Supp 1258 (E. D. P.A. 1972).
- Perlman, R. (1974). *TORTIS (Toddler's own recursive turtle interpreter system)* (No. LOGO-9) (pp. 1–10). Cambridge, MA: Massachusetts Institute of Technology. Retrieved from <http://files.eric.ed.gov/fulltext/ED118366.pdf>

- Perlman, R. (1976). *Using computer technology to provide a creative learning environment for preschool children. ai memo 360* (No. LOGO-24) (pp. 1–31). Cambridge, MA: Massachusetts Institute of Technology. Retrieved from <http://files.eric.ed.gov/fulltext/ED207576.pdf>
- Polloway, E. A., Patton, J. R., & Marvalin, N. A. (2011). Intellectual and developmental disabilities. In J. M. Kaufman & D. P. Hallahan (Eds.), *Handbook of Special Education* (pp. 175–186). New York, NY: Routledge, Taylor, & Francis Group.
- Potter, A. (1853). *Education of idiots: An appeal to the citizens of Philadelphia*. Philadelphia, PA: H. Evans.
- Ratcliff, C. C., & Anderson, S. E. (2011). Reviving the turtle: Exploring the use of logo with students with mild disabilities. *Computers in the Schools*, 28(3), 241–255. <https://doi.org/10.1080/07380569.2011.594987>
- Rehabilitation Act of 1973, Pub. L. No. 93–112, § 504 (1973).
- Reynolds, M. C. (1962). A framework for considering some issues in special education. *Exceptional Children*, 28(7), 367–370.
- Rücker, M. T., & Pinkwart, N. (2016). Review and discussion of children’s conceptions of computers. *Journal of Science Education and Technology*, 25(2), 274–283. <https://doi.org/10.1007/s10956-015-9592-2>
- Ruppar, A. L., Neeper, L. S., & Dalsen, J. (2016). Special education teachers’ perceptions of preparedness to teach students with severe disabilities. *Research & Practice for Persons with Severe Disabilities*, 41(4), 273–286. <https://doi.org/10.1177/1540796916672843>
- Scruggs, T. E., & Mastropieri, M. A. (1997). Can computers teach problem-solving strategies to students with mild mental retardation? *Remedial & Special Education*, 18(3), 157–164.

- Sermier Dessemontet, R., & Bless, G. (2013). The impact of including children with intellectual disability in general education classrooms on the academic achievement of their low-, average-, and high-achieving peers. *Journal of Intellectual & Developmental Disability*, 38(1), 23–30. <https://doi.org/10.3109/13668250.2012.757589>
- Spooner, F., & Browder, D. M. (2015). Raising the bar: Significant advances and future needs for promoting learning for students with severe disabilities. *Remedial and Special Education*, 36(1), 28–32.
- Spooner, F., & Brown, F. (2011). Educating students with severe cognitive disabilities. In J. M. Kaufman & D. P. Hallahan (Eds.), *Handbook of Special Education* (pp. 503–515). New York, NY: Routledge, Taylor, & Francis Group.
- Spooner, F., Knight, V., Browder, D., Jimenez, B., & DiBiase, W. (2011). Evaluating evidence-based practice in teaching science content to students with severe developmental disabilities. *Research and Practice for Persons with Severe Disabilities (RPSD)*, 36(1–2), 62–75. <https://doi.org/10.2511/rpsd.36.1-2.62>
- State of the Union Address*. (2016). Retrieved from <https://www.whitehouse.gov/the-press-office/2016/01/12/remarks-president-barack-obama-%E2%80%93-prepared-delivery-state-union-address>
- Strawhacker, A., & Bers, M. (2015). “I want my robot to look for food”: Comparing kindergarteners’ programming comprehension using tangible, graphic, and hybrid interfaces. *International Journal of Technology & Design Education*, 25(3), 293–319.
- Sullivan, A., & Bers, M. U. (2013). Gender differences in kindergarteners’ robotics and programming achievement. *International Journal of Technology & Design Education*, 23(3), 691–702. <https://doi.org/10.1007/s10798-012-9210-z>

- Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, 26(1), 3–20. <https://doi.org/10.1007/s10798-015-9304-5>
- Sullivan, A., Kazakoff, E. R., & Bers, M. U. (2013). The wheels on the bot go round and round: Robotics curriculum in pre-kindergarten. *Journal of Information Technology Education: Innovations in Practice*, 12, 203–219.
- Sullivan, F. R., & Heffernan, J. (2016). Robotic construction kits as computational manipulatives for learning in the STEM disciplines. *Journal of Research on Technology in Education*, 48(2), 105–128. <https://doi.org/10.1080/15391523.2016.1146563>
- Taylor, M. S., Vasquez, E., & Donehower, C. (2017). Computer programming with early elementary students with Down syndrome. *Journal of Special Education Technology*, 1–11. Retrieved from <http://journals.sagepub.com/eprint/VaXPBktvkMbAwFRSHf8Q/full>
- U.S. Department of Education. (2016). *Future ready learning: Reimagining the role of technology in education* (National Education Technology Plan No. ED-04-CO-0040/0010). Washington, DC. Retrieved from <http://tech.ed.gov>
- Vannest, K. J., Parker, R. I., & Gonen, O. (2016). Single Case Research: Web based calculators for SCR analysis (Version 1.0). College Station, TX: Texas A&M University. Retrieved from singlecaseresearch.org
- Varney, M. W., Janoudi, A., Aslam, D. M., & Graham, D. (2012). Building young engineers: TASEM for third graders in woodcreek magnet elementary school. *IEEE Transactions on Education*, 55(1), 78–82. <https://doi.org/10.1109/TE.2011.2131143>

- Vilorio, D. (2014). *STEM 101: Intro to tomorrow's jobs* (Occupational Outlook Quarterly) (p. 12). Washington, DC: Bureau of Labor Statistics. Retrieved from www.bls.gov/ooq
- Wakeman, S., Karvonen, M., & Ahumada, A. (2013). Changing instruction to increase achievement for students with moderate to severe intellectual disabilities. *Teaching Exceptional Children, 46*(2), 6–13.
- What Works Clearinghouse. (2014). What Works Clearinghouse: procedures and standards handbook (version 3.0). Retrieved from http://ies.ed.gov/ncee/wwc/pdf/reference_resources/wwc_procedures_v2_1_standards_handbook.pdf
- Whitehouse Office of Science and Technology Policy. (2015). Preparing Americans with 21st century skills: Science, technology, engineering, and mathematics (STEM) education in the 2015 budget. Retrieved from www.whitehouse.gov/ostp
- Witzel, B. S., Mercer and, C. D., & Miller, M. D. (2003). Teaching Algebra to Students with Learning Difficulties: An Investigation of an Explicit Instruction Model. *Learning Disabilities Research & Practice (Wiley-Blackwell), 18*(2), 121–131. <https://doi.org/10.1111/1540-5826.00068>
- Wolf, M. M. (1978). Social validity: The case for subjective measurement or how applied behavior analysis is finding its heart. *Journal of Applied Behavior Analysis, 11*, 203–214. <https://doi.org/10.1901/jaba.1978.11-203>
- Wyatt v. Stickney, No. 325 F. Supp. 781 (M.D. Ala. 1972).
- Xie, Y., Fang, M., & Shauman, K. (2015). STEM education. *Annual Review of Sociology, 41*, 331–357. <https://doi.org/10.1146/annurev-soc-071312-145659>

- Yelland, N. (1995). Mindstorms or a storm in a teacup? A review of research with Logo. *International Journal of Mathematical Education in Science & Technology*, 26(6), 853–869. <https://doi.org/10.1080/0020739950260607>
- Yudin, M., & Musgrove, M. (2015). *Guidance on FAPE: Dear Colleague Letter*. Washington, DC: U.S. Department Of Education. Retrieved from <https://www2.ed.gov/policy/speced/guid/idea/memosdcltrs/guidance-on-fape-11-17-2015.pdf>