12-2009

An Experimental Comparison of Middle School Students Motivation and Preference Toward Text and Graphic-Based Programming

Stephen E. Williams
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd
Part of the Education Commons

Recommended Citation
Williams, Stephen E., "An Experimental Comparison of Middle School Students Motivation and Preference Toward Text and Graphic-Based Programming" (2009). All Graduate Theses and Dissertations. 487.
https://digitalcommons.usu.edu/etd/487

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact dylan.burns@usu.edu.
ABSTRACT

An Experimental Comparison of Middle School Students’ Motivation and Preference Toward Text- and Graphic-Based Programming

by

Stephen E. Williams, Master of Science
Utah State University, 2009

Major Professor: Gary Stewardson, Ph.D.
Department: Engineering and Technology Education

The purpose of this study was to compare seventh-grade students’ motivation and preference toward text-based programming using Visual Basic, and graphics-based programming using Robolab. Motivation was defined by the My Class Activities questionnaire using the dimensions of interest, challenge, choice, and enjoyment. Preference was determined through team and individual student choice. This study was conducted with 122 students from three 6-week technology education classes. This study examined two hypotheses. First, middle school students will be more motivated when using a graphics-based programming language than text-based as measured by the My Class Activities survey. The second hypothesis for this study was that middle school students preferred using graphic-based programming more than using text-based programming in an introductory experience. Student preference was identified individually and within a team environment.

(96 pages)
I would like to thank my committee for taking the time to work through this study with me. I would especially like to thank Dr. Stewardson for the time and effort he put in helping me, without which this would not have been possible.

I also thank my wife for her support and encouragement in all my educational pursuits.

Stephen E. Williams
# CONTENTS

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Need Statement</td>
<td>3</td>
</tr>
<tr>
<td>Statement of the Problem</td>
<td>4</td>
</tr>
<tr>
<td>Hypothesis</td>
<td>4</td>
</tr>
<tr>
<td>Assumptions</td>
<td>4</td>
</tr>
<tr>
<td>Limitations</td>
<td>5</td>
</tr>
<tr>
<td>Terminology</td>
<td>5</td>
</tr>
<tr>
<td>Acronyms</td>
<td>7</td>
</tr>
<tr>
<td>II. REVIEW OF LITERATURE</td>
<td>8</td>
</tr>
<tr>
<td>Introduction</td>
<td>8</td>
</tr>
<tr>
<td>Evolution of Technology Education</td>
<td>10</td>
</tr>
<tr>
<td>RCX Brick</td>
<td>13</td>
</tr>
<tr>
<td>RoboLAB</td>
<td>15</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>18</td>
</tr>
<tr>
<td>Text Verses Graphics</td>
<td>20</td>
</tr>
<tr>
<td>Survey Instrument</td>
<td>22</td>
</tr>
<tr>
<td>Reliability and Validity</td>
<td>23</td>
</tr>
<tr>
<td>Summary</td>
<td>24</td>
</tr>
<tr>
<td>III. METHODOLOGY</td>
<td>25</td>
</tr>
<tr>
<td>Introduction</td>
<td>25</td>
</tr>
<tr>
<td>Population</td>
<td>25</td>
</tr>
<tr>
<td>Sample</td>
<td>26</td>
</tr>
<tr>
<td>Research Design</td>
<td>26</td>
</tr>
<tr>
<td>Curriculum</td>
<td>27</td>
</tr>
</tbody>
</table>
## IV. FINDINGS

- Introduction ................................................................................................. 34
- Findings Relevant to Student Motivation .................................................... 34
- Findings Relevant to Student Preference ..................................................... 41
- Summary ...................................................................................................... 44

## V. CONCLUSIONS AND RECOMMENDATIONS

- Introduction ................................................................................................. 46
- Conclusions Regarding Motivation .............................................................. 46
- Conclusions Regarding Preference .............................................................. 54
- Recommendations for Teachers ................................................................. 56
- Recommendations for Further Study ......................................................... 57
- Summary ...................................................................................................... 58

### REFERENCES

- References .................................................................................................... 60

### APPENDICES

- Appendix A: Visual Basic Curriculum Example ......................................... 64
- Appendix B: RoboLAB Curriculum Example .............................................. 74
- Appendix C: *My Class Activities Survey* .................................................. 85
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Test-Retest Data <em>My Class Activities</em> Survey</td>
<td>24</td>
</tr>
<tr>
<td>2. Frequency, Mean, and Standard Deviation for the Dimension of Interest Items 1-8</td>
<td>36</td>
</tr>
<tr>
<td>3. Mean Scores and Standard Deviations for the Dimension of Interest Items 1-8 Categorized by Programming Language</td>
<td>37</td>
</tr>
<tr>
<td>4. Frequency, Mean, and Standard Deviation for the Dimension of Challenge Items 9-17</td>
<td>38</td>
</tr>
<tr>
<td>5. Mean Scores and Standard Deviations for the Dimension of Challenge Items 9-17 Categorized by Programming Language</td>
<td>39</td>
</tr>
<tr>
<td>6. Frequency, Mean, and Standard Deviation for the Dimension of Choice Items 18-24</td>
<td>40</td>
</tr>
<tr>
<td>7. Mean Scores and Standard Deviations for the Dimension of Choice Items 18-24 Categorized by Programming Language</td>
<td>40</td>
</tr>
<tr>
<td>8. Frequency, Mean, and Standard Deviation for the Dimension of Enjoyment Items 25-31</td>
<td>42</td>
</tr>
<tr>
<td>9. Mean Scores and Standard Deviations for the Dimension of Enjoyment Items 25-31 Categorized by Programming Language</td>
<td>42</td>
</tr>
<tr>
<td>10. Frequency Table Showing Observed Team Choice</td>
<td>43</td>
</tr>
<tr>
<td>11. Frequency Table Showing Individual Student Preference</td>
<td>44</td>
</tr>
<tr>
<td>12. Mean and Standard Deviations for Interest in Both Programming Languages</td>
<td>48</td>
</tr>
<tr>
<td>13. 2 x 2 x 2 ANOVA Showing the Results for the Factors of Curriculum, Order, and Session with Regard to the Dimension of Interest</td>
<td>49</td>
</tr>
<tr>
<td>14. Mean and Standard Deviations for Challenge in Both Programming Languages</td>
<td>50</td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>15. 2 x 2 x 2 ANOVA Showing the Results for the Factors of Curriculum, Order, and Session with Regard to the Dimension of Challenge</td>
<td>51</td>
</tr>
<tr>
<td>16. Mean and Standard Deviations for Choice in Both Programming Languages</td>
<td>52</td>
</tr>
<tr>
<td>17. 2 x 2 x 2 ANOVA Showing the Results for the Factors of Curriculum, Order, and Session with Regard to the Dimension of Choice</td>
<td>52</td>
</tr>
<tr>
<td>18. Mean and Standard Deviations for Enjoyment in Both Programming Languages</td>
<td>54</td>
</tr>
<tr>
<td>19. 2 X 2 X 2 ANOVA Showing the Results for the Factors of Curriculum, Order, and Session with Regard to the Dimension of Enjoyment</td>
<td>54</td>
</tr>
<tr>
<td>20. Frequency Table Showing Team Preference</td>
<td>55</td>
</tr>
<tr>
<td>21. Frequency Table Showing Individual Student Preference</td>
<td>56</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>1.</td>
<td>A typical RoboLAB screen</td>
</tr>
<tr>
<td>2.</td>
<td>A typical Visual Basic screen</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

From the cable box used to watch your favorite TV show to the microprocessor in your car that performs self-diagnostics and e-mails you the results, controllers and microprocessors are used in almost every part of our daily lives. A controller is a basic system that uses a processor to receive data through inputs, processes that data via a program, and then controls various outputs based on the program logic. An example of this would be a bar code scanner at the supermarket. A laser scans the bar code on the item to be purchased and the information is sent to the microprocessor. The microprocessor processes the data and identifies the item being purchased. This information is sent to the teller, and is used to inventory product levels and cues the reordering of items. Other examples of the use of controllers include using a remote control keypad to unlock a car door, warming up your lunch in a microwave, and controlling a robotic arm used for welding parts in a factory. “Technology is so woven into the fabric of modern life that is has become all but invisible” (Pearson, 2002, p. 48).

The use of systems described in the examples above is referred to as control technology or control systems. A control system uses a computer or microprocessor to monitor inputs and control outputs. A program is at the heart of the control technology, it represents the logic that is used to interpret data from inputs and control outputs. A variety of programming languages are used in control technology. Traditionally, programming interfaces have been text based, meaning that the program uses strings of text and syntax to define the program logic. Some examples of programming interfaces
that are text-based are FORTRAN, Visual Basic (VB), and C++. More recently graphic-based programming interfaces have been introduced, meaning graphics or icons are positioned and linked to define the program logic. Two examples of graphic-based programming interfaces include LabVIEW and RoboLAB. The transition from text-based programming to graphic-based programming is similar to the evolution of the personal computer (PC) operating system.

One of the early operating systems for PCs was Disk Operating System or DOS. When using DOS the operator would receive a prompt on the screen to type text commands then press enter to execute those commands. Text-based commands were used to control every aspect of the computer system from file management to running programs. As technology advanced, graphic applications become common. The next development for the PC was the “user friendly” graphical user interface (GUI) operating system. This was first introduced to the public by Apple with the development of the Macintosh Operating System or Mac OS. This GUI format was quickly accepted by the public and embraced by competitors. These GUI operating systems provided a pick-and-click method of computer operation and now the GUI is second nature to most computer users. Even with the new GUI operating system the older text-based DOS language continued to be used by many people, and taught in schools. One reason DOS continued to be used was that teachers were more familiar with the DOS text operating system and that is what they felt comfortable teaching and using. GUIs quickly have become the standard for new computer applications. Today, popular games, word processing software, and other programs utilize a GUI pick-and-click environment.
One common issue in most university engineering programs is retention. Felder (1998) stated that in a study of 25,000 students at over 300 institutions only 43% of freshmen engineering students went on to graduate (p. 470). Felder went on to identify four main reasons for this issue of retention, (a) students’ attitudes toward engineering, (b) their self-confidence levels, (c) the quality of their interactions with instructors and peers, and (d) their aptitude for engineering. All of these variables can be related to the students’ motivation and classroom experience.

A typical teacher teaching computer programming in a public school would often select a text-based interface because that is what they are most familiar with. However, students that have grown up using GUI based operating systems and computer applications may be more motivated when using a graphics-based programming interface. It is currently unclear if students prefer and are more motivated using a graphics-based programming interface or the more common text-based interface.

Need Statement

Control technology is so commonplace in society that most people are exposed to it in some form on a daily basis. Since control systems impact our lives on a daily basis, individuals need a minimum level of technological literacy regarding the programming and logic that is used to develop and operate control systems. Taylor (1986) suggested that a graphical programming interface may be more “user friendly,” and easier to use. Taylor hypothesized that “intuition seems to suggest that the use of graphic materials makes learning easier” (p. 56). Little research, however, exists to support the assumption
that graphic-based programming interfaces are easier or more motivational for beginners. The preference to a programming interface, text-based verses graphic-based is uncertain. Are students having their introductory experiences learning programming of control systems more motivated using a text-based or graphic-based environment?

Statement of the Problem

The problem of the study was to conduct a experimental comparison of seventh grade students’ motivation and preferences toward text-based programming using VB, and graphics-based programming using RoboLAB. Specifically, measuring the dimensions listed in the My Class Activities instrument of interest, challenge, choice, and enjoyment.

Hypothesis

This study has examined the following two hypotheses.

1. Middle school students will be more motivated when using a graphics-based programming language than text-based as measured by the My Class Activities survey.

2. Middle school students prefer using graphic-based programming more than using text-based programming in an introductory experience.

Assumptions

The following assumptions were made in this study.

1. Students would have little or no knowledge of robots and control systems
prior to the course.

2. The curriculum was the same or similar in difficulty level except for the user interface used to program.

3. Administering the survey instrument twice would have a minimal effect on the student responses due to the counterbalanced design.

4. Both programming methods were taught with minimal teacher bias.

5. Students enrolled in this study are have used and are comfortable with operating computers.

Limitations

The following limitations were identified while conducting this study.

1. This study was limited specifically to middle school students in a suburban school district.

2. This study was limited to the use of VB and RoboLAB as programming languages.

Terminology

Challenge—as defined by the *My Class Activities survey*. “Engages the student and requires extra effort” (Gentry & Gable, 2001, p. 4).

Choice—as defined by the *My Class Activities survey*. “Gives the student the right power to select educational options and direct his or her own learning” (Gentry & Gable, 2001, p. 4).
Enjoyment—as defined by the *My Class Activities survey*. “Provides the student with pleasure and satisfaction” (Gentry & Gable, 2001, p. 4).

Graphic-based programming—A programming interface that uses the manipulation of graphics to create a program. Also known as Visual Programming and Icon Based programming.

GUI—Graphical User Interface, graphics-based user interface that incorporates icons, pull-down menus and a mouse, as in Microsoft Windows or Mac OS. The GUI has become the standard way users interact with a computer.

Interest—as defined by the *My Class Activities survey*. “Reflects positive feelings/preferences for certain topics, subject areas, or activities” (Gentry & Gable (2001, p. 4).

Motivation—as defined by the *My Class Activities survey*. Operationally defined as a combination of the following four dimensions: interest, challenge, choice, and enjoyment.

RCX - Robotics Command eXplorer. This is the name given to the Lego microcontroller used in this study.

RoboLAB—Graphics-based programming software that can be used to program the RCX.

Text-based programming—A programming language that uses the manipulation of text to create a program.

Visual Basic—Text-based programming software that can be used to program the RCX.
Acronyms

RCX—Robotics Command eXplorer

GUI—Graphical User Interface

STEM—science, technology, engineering, and mathematics
CHAPTER II
REVIEW OF LITERATURE

Introduction

A control system uses a computer or a microprocessor to monitor inputs and control outputs. Using control systems in today’s society is so commonplace that many people take them for granted. A program is a key aspect of control technology, it represents the logic that is used to interpret input data and then control outputs. Programming is the “process of transforming a mental plan of desired actions for a computer into a representation that can be understood by the computer” (Myers, 2006, p. 76).

One common issue in most university engineering programs is retention. Felder (1998) stated that in a study of 25,000 students at over 300 institutions only 43% of freshmen engineering students went on to graduate (p. 470). Rugarcia (2000) speculated that part of the reason for the retention issue is that if we look at an engineering classroom we “see the same thing we would have seen in 1970 or 1940” (Rugarcia, p. 1). The argument is that many professors are “unsure of what the alternatives are to the traditional methods, and even those who know about alternatives fear that transforming the way they teach will require a full-time commitment” (Rugarcia, p. 2). Rugarcia also stated the need for “significant changes in engineering education will be required if we are to meet the needs of our graduates in preparing them for the challenges of the coming century” (p. 5). Engineering must adapt to society changes to remain relevant (Pearson,
Traditionally, programming languages have been text-based, however graphic programming languages are appearing and being used for programming control systems. The development of graphical programming languages has followed a similar path as the evolution of the personal computer operating system. After Apple released the first graphical operating systems, other companies followed their lead. Due to this transition to the graphical user interface (GUI) occurring before current middle schools students were born, middle school children have grown up using a GUI for most of their computer applications.

Because students are familiar with the GUI, programming in a graphic language may be more user friendly and intuitive to this new generation. Teachers, on the other hand, may be more comfortable teaching text-based programming because their introductory experiences were with text-based operating systems, applications, and programming languages. This review of literature will examine the evolution and the inclusion of K-12 engineering in its content area; identify programming languages that may be used to create programs for the LEGO RCX microcontroller and explore ways it has been used in education; review the current research regarding text versus graphics-based programming; and, discuss the instrument used in this study. The purpose of this chapter is to evaluate previous research in the area of programming environments and how they relate to the middle school learner.
Evolution of Technology Education

What we know as technology education, as taught in public schools today, originated from the industrial arts (IA) content area. Industrial arts programs have been in existence since the early part of the twentieth century; however, in 1978 the Standards for Industrial Arts Education Project identified the top five curriculum areas that were currently taught in industrial arts programs. These areas were “general woodworking, general metals, general IA, architectural drafting, and mechanical drawing” (Dixon, 1980, p. 33). The rationale for “industrial education was that children needed to learn about technologies of the home and of commercial industry to understand their increasingly technological world” (Foster, 1995, p. 7). The industrial arts curriculum fit the social and cultural needs of the time, but as new advances in technology came, the industrial arts curriculum became inadequate to educate students about the technological world. Wright (1995) noted that “specific skills are being rapidly replaced by new ones” (p. 248). Just 2 years later, in 1980, came the Jackson’s Mill Industrial Arts Curriculum Symposium, which developed a new framework and philosophy for the profession. “The content outline that was created as a result of this meeting focused on the adaptive technological systems of manufacturing, construction, transportation, and communication” (Lewis & Zuga, 2005, p. 10). The efforts of the Jackson’s Mill team laid the foundation for the change from industrial arts to technology education.

In 2000, the International Technology Education Association (ITEA) released The Standards for Technological Literacy: Content for the Study of Technology (STL) as a result of “the growing importance of technology to our society, it has become vital that
students receive an education that emphasizes technological literacy” (ITEA, 2004, p. vii). Gorham (2002) pointed out that the STL’s began a relationship between K-12 engineering and technology education by stating:

As school districts adopt and implement *Standards for Technological Literacy*, increased numbers of pre-college students will be exposed to the breadth of engineering. This exposure is likely to result in more students understanding engineering principles and choosing engineering as a career (p. 32).

He also mentioned the Standards for Technological Literacy will “equip students to be better prepared in pursuing engineering degrees” (p. 34). Salinger (2003), a program officer at the National Science Foundation, stated, “In the last decade or so, technology education has reinvented itself to look more like engineering education” (p. 95). He also suggest that maybe now is the time to “make the move” (p. 95) to engineering education.

The state departments of technology education in Utah, Massachusetts, and Michigan have already made the change by including engineering in the title of their department.

The ITEA conducted a survey in 2006 regarding a potential name change within their organization to include engineering. In this survey, 19% of the people surveyed indicated that the name of their program has changed in the past two years, and an additional 11% of those surveyed indicated their program was planning a name change. In addition, when presented with a list of words that best characterize the content area of technology education, 52% of those surveyed felt engineering best characterized the nature of content in technology education (Starkweather, 2008).

The school district used for this research is taking steps to rename their industrial technology department, including two high schools and middle schools, to include engineering. Coinciding with this name change, the Introduction to Technology course
was modified to include robotics to help teach science, technology, engineering, and mathematics (STEM). The *Introduction to Technology* class was taught to seventh grade students during a 6-week term. The curriculum consisted of three main units; aerodynamics and flight, transportation, and control systems. The control systems unit is a recent addition to the curriculum and lasts 4 of the 6 weeks because multiple concepts can be taught using control systems. For example, Sklar (2004) used robotics to assist teaching math concepts. Concepts addressed by the robotics curriculum are: geometry, electricity, programming, logic, and simple machines.

While developing the robotic curriculum for the *Introduction to Technology* class two graphical programming languages became available for middle school applications, RoboLAB and VB. Both languages are considered appropriate for middle school students. RoboLAB is based on the LabVIEW programming environment and uses graphics to represent commands, while VB uses text to create commands.

It was uncertain which programming language would be best to teach middle school students enrolled in a required Technology and Engineering Education course. Many programming languages are text-based and students could begin to learn the syntax and method of writing a program to control a robotic system. However, if students are more familiar with GUI environments and graphical programming languages are going to be available in the future, students might gain more experience using the pick-and-click environment.
RCX Brick

In 1998, at the world cup of soccer in Paris, France, LEGO introduced their LEGO Mindstorms RCX to the world by presenting a Robot soccer match. This match served as a test of the robotic system and to generate interest in the new microcontroller (Lund, 1999). The RCX brick is “an autonomous microcomputer embedded in a LEGO brick that can be programmed to serve as the “brain” of any LEGO construction” (Potsmore, 1999, p. 26). It consists of a Hitachi H8 microcontroller with 32 KB of RAM. “ALEGO Mindstorms Robot is controlled by the RCX and can be programmed to execute various tasks” (Vento, 2002, p. 73). The students are able to download a program to the RCX from a PC or MAC using an IR transceiver. LEGO connectors are used to connect various LEGO motors and lamps to the RCX through three output ports. The RCX also has three input ports that students can connect LEGO sensors such as push button sensors and light sensors. With the development of the LEGO Mindstorms RCX, “robotics projects provide an opportunity to directly interact with technology, as well as an opportunity to design and implement the various concepts that they embrace” (Weinberg, 2001, p. 1).

Since LEGO released the RCX, many universities have included it in the first or second year of their engineering courses to aid in retention of students in the engineering field (Froyd, 2005; Tester, 2005). For example, the University of Nevada at Reno has been using the LEGO RCX to teach freshman mechanical engineering classes. Since the implementation of the course, Dr. Eric Wang (2001) noted the following benefits of using the LEGOcs in their program.
1. The use of LEGO was found appealing to the students while providing an excellent medium for teaching design, programming, and creativity.

2. The use of ROBOLAB was found to significantly increase the student’s ability to program the robots as compared to the other two languages used.

3. ROBOLAB appears to be an effective method of introducing students to LabVIEW early in the curriculum.

4. Enrollment has reached a point where it is not possible for one instructor to effectively run the course. Multiple sections and/or year-round offering are required.

5. The use of LEGO in the freshmen program has proved to be an excellent recruiting tool. The enrollment has more than doubled in three years, despite the national trend of decreasing enrollment in engineering programs. (p. 15)

There are many programming languages, both graphics-based and text-based, that can be used to program the RCX. Vento (2002) identified the following four languages, “not Quite C (NQC), ROBOLAB, Spirit.ocx with VB, and LeJOS (Lego Java Operating System)” (p. 72). Ten additional programming languages have been identified by Patterson-McNeill (2001) including the following:

1. Programmable Brick FORTH
2. legOS
3. BrainStorm
4. Bot-Kit
5. TclRCX
6. BrickCommand
7. BotCode
8. Gordon's Brick Programmer
9. Mind Control
10. PRO-RCX.

Because so many programming languages exist that can be used to program the RCX, is not difficult to find an environment that will allow significant knowledge to more complex programming applications in the future.

The programming languages used in this study are simplified versions of programming environments that are currently used in industry today. The educational software distributed by LEGO with the Mindstorms kit is called RoboLAB. RoboLAB is based on the graphical programming environment of LabVIEW. The spirit.ocx control enables students to program the RCX using the text-based code in VB. Students can use both languages to write programs on the computer and transmit them to the RCX brick via an infrared transmitter connected through the USB or Serial Port. These programming environments are considered age appropriate for middle school students and have approximately the same level of difficulty.

RoboLAB

The idea of using graphical programming environments has been around since as early as 1965 (Boshernitsan, 2004, p. 1). RoboLAB is a graphic programming environment for use with the LEGO Mindstorms robot kits and is based on LabVIEW. LabVIEW is one of the popular graphical programming languages used today by the science and engineering community. LabVIEW was first released in 1986 by National Instruments and has been reported “to improve product quality, get to market faster, and gain greater engineering and manufacturing efficiency” (Company Overview, 2007, p. 1).
LabVIEW is a powerful programming environment, and like most programming environments is difficult for young students to learn. Potsmore (1999), mentioned that software was needed that could be used by students with a wide range of ages and abilities. “These needs were answered through the formation of a partnership between LEGO DACTA (the division of LEGO that distributes educational materials), Tufts University’s College of Engineering, and National Instruments, makers of LabVIEW programming software” (Potsmore, p. 27). Through this partnership came the creation of RoboLAB. RoboLAB “runs on a modified version of LabVIEW 5” (Irwin, 1999, p. 7). RoboLAB “meets the demands of a sophisticated user, yet at the same time it is simple enough for young students to create meaningful programs for their LEGO designs” (Potsmore, p. 28).

RoboLAB software has three levels, pilot, inventor, and investigator. Pilot is a simple programming method to be used with children in elementary school. “Teachers have found that children can progress through the four levels in Pilot with very little instruction” (Potsmore, 1999, p. 32). Inventor and Investigator are more complex than Pilot and are used at the middle school to high school levels grade 6-12. Investigator is used primarily as a data acquisition segment of RoboLAB while Inventor is used primarily to control the RCX. For the purposes of this study, the students will be using the Inventor level to create input and output control logic. When compared with the Pilot level, “the Inventor section offers a new level of flexibility and power coupled with a slightly different but still graphical interface” (Potsmore, p. 34).
Figure 1 shows the four main parts to the inventor level of the RoboLAB programming language. These four parts are similar to the LabVIEW environment, the block diagram window, the front panel window, functions pallet, and the tools pallet. The main window in the program is called the block diagram window. This is the main area of the program where the students actually place the icons, and digitally wire them together, to create their program. Aside from wiring icons together this window is also used to download the program to the RCX brick. The RoboLAB software compiles the code and is able to transmit the program to a microcontroller through an infrared transmitter.
The front panel window is a feature of RoboLAB that is not commonly used by students when programming. It can be used to place reminders for the programmer regarding the expected outcomes and functions of the program.

The functions palette is the main menu where the programming icons are located. Several icons can be dragged and dropped into the block diagram window of the program. Each icon is a symbol for some action and when they are connected in sequence with the wire tool, the logic for the program is created. The icons range in complexity from simple to complex depending on the abilities of the programmer.

The tools pallet has basic tools that are needed to write a RoboLAB program such as: a wire tool to connect icons together in sequence, a selection tool to allow icons to be selected, moved, and modified, and a text tool to include reminders above complicated parts of the program. These tools are used to manipulate the icons on the block diagram window to assist in the creation of a program.

Compared with LabVIEW the programming environment of RoboLAB has a very similar look and uses the same programming process. One benefit of RoboLAB being similar to LabVIEW is students that learn how to program with RoboLAB can transfer some of the knowledge they gained into programming LabVIEW in future applications.

**Visual Basic**

Visual Basic was first developed in 1991 (Coombs, 2002, p. 10) and is a programming language developed by Microsoft that allows the programmer to use text-based programming to create a GUI. Although VB has a GUI, text is used for writing the
program. For the purposes of this study, students will be writing VB text-based code to read inputs and control outputs on the RCX. Because of the GUI, and the ease of learning the basic language, VB is very user friendly and is currently taught in many middle schools around the country.

To enable the use of VB to program the RCX the LEGO program called SDK needs to be installed in the computer to supply the spirit.ocx control. “Spirit.ocx is a regular ActiveX control, which means its functions are accessible from programming languages like VB and Visual C++” (Vento, 2002, p. 73). By using the spirit.ocx control students can write programs in VB using text and syntax.

Figure 2 shows the two main windows in VB, the first is the form window where students create a button that is used for downloading the program to the RCX. The other window, the code window, contains the text code that will execute after the VB program

![Figure 2. A typical Visual Basic screen.](image-url)
is compiled. This code window is where the students write their program. VB is a useful language in this study because programs can be easily written and transmitted to the LEGO RCX quickly and easily.

RoboLAB and VB with the spirit.ocx control are programming languages that are considered age appropriate for middle school students and are based on programming environments that are used by professionals in industry. The use of the LEGO Mindstorms RCX is also considered age appropriate, it presents the opportunity to conduct this type of study comparing student motivation and preference toward these two types of programming languages.

Text Verses Graphics

Little research exists to assist in determining which method of programming, text versus graphics, is preferred over another. Some researchers (e.g., Fernaeus, Kindborg, & Scholz, 2006; Myers, 1990) made the argument for the use of graphic-based programming applications. However, Petre (1995) argued against graphic-based programming. Neag (2001) argued that “the two paradigms can be used together” (p. 658). More research comparing these two programming methods is needed.

Myers (1990) stated, “Visual programming systems have successfully demonstrated that non-programmers can create fairly complex programs with little training” (p. 1). Myers went on to mention two main benefits of graphical programming environments, “the human visual system and human visual information processing are clearly optimized for multi-dimensional data. Computer programs, however, are
conventionally presented in a one-dimensional textual form, not utilizing the full power of the brain” (Myers, p. 3), and that graphical programming “tends to be a higher-level description of the desired actions (often de-emphasizing issues of syntax and providing a higher level obstruction) and may therefore make the programming task easier even for professional programmers” (Myers, p. 4).

Fernaeus and colleagues (2006) used the ease of reading comic books, also called graphic novels, to argue for the use of graphical programming languages by stating “contextual signs in comics are shown as an effect of what happens to a character, and in programming similar signs could be used to produce an effect” (p. 123). Fernaeus and colleagues also conceded, “A limitation of the approach is that the details of more complex behaviors are difficult to represent in the format of contextual signs” (p. 128). Lourens (2004) even went as far as making the statement that, “Visual programming has been proven to be more efficient than classical textual programming” without presenting any empirical evidence to support his claim (p. 1).

Petre (1995) provided an argument against graphical languages by asking the question, “‘a picture is worth a thousand words’—isn’t it? And hence graphical representation is by its nature universally superior to text—isn’t it? Why then isn’t the anecdote itself expressed graphically?” (p. 33). He argued that graphical programming has a downside of readability, or the programs that are written using graphical expressions are difficult for other programmers to read and understand. He states that when specifically using LabVIEW “I quite often spend an hour or two just moving boxes and wires around, with no change in functionality, to make it that much more
comprehensible when I come back to it” (p. 42).

Neag (2001) concluded, by recommending “the use of both textual and visual languages for test procedure development to support applications with different levels of complexity” (p. 670). Taylor (1986) suggested that a graphical-based programming languages can “aid beginners by eliminating common bugs” (p. 182); however, Taylor also noted the need for more research in this area. It is clear from the literature more research is needed in this area.

Survey Instrument

The survey instrument titled *My Class Activities* will be used for this study to measure the construct of motivation using four sub scales of interest, challenge, choice, and motivation. Amabile and Hannessey (1992) pointed out one of the reasons for determining which programming method is better for motivating students when he states, “People will be most effective when they feel motivated primarily by the interest, enjoyment, satisfaction and challenge of the work itself—not by external pressures” (p. 55). Students who are not motivated “can sometimes be disruptive in class, distracting students who are motivated from their own work” (Gentry & Gable, 2001, p. 1).

The *My Class Activities* survey is a 31 item instrument, used in grades three through eight, to assess students’ perception in four areas; interest, challenge, choice, and enjoyment of their classrooms (Gentry & Gable, 2001). All items are presented on a 5-point Likert scale. Eight items are used to measure interest, nine measure challenge, seven measure choice, and seven measure enjoyment. The survey has four dimensions
that make up the construct of motivation and are defined as follows.

Interest: Reflects positive feelings/preference for certain topics, subject areas, or activities.

Challenge: Engages the student and requires extra effort.

Choice: Gives the student the right or power to select educational options and direct his or her own learning.

Enjoyment: Provides the student with pleasure and satisfaction. (Gentry & Gable, 2001, p. 4)

Reliability and Validity

This instrument has been used by researchers and is commercially available for use in educational research. An internal validity score for middle school grade levels was based on the data obtained from approximately 1523 student respondents from 61 classrooms. Validity data are based on the Tucker-Lewis “goodness of fit” index with a score of .88, a mean root square residual of .09 and suggested that the validity of the score interpretations (i.e., construct validity) of the four dimensions solution was adequately supported by the data (Gentry & Gable, 2001, p. 23). Values range from 0 to 1.0, with 0 indicating a poor fit, and 1.0 indicating a perfect fit. Generally, values over .90 are considered excellent. A table showing the validity numbers for each question on the survey can be found in Appendix C. The reliability coefficients for the survey are broken up into the four dimensions and are shown in Table 1 and range from .66-.74. A common rule of thumb is .80 or higher for adequate reliability and .90 or higher for good reliability. However, for exploratory research, a cutoff as low as .60 is not uncommon therefore these values are adequate for this type of affective survey instrument.
Table 1

*Test-Retest Data My Class Activities Survey*

<table>
<thead>
<tr>
<th>Scale</th>
<th>Reliability estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interest</td>
<td>.70</td>
</tr>
<tr>
<td>Challenge</td>
<td>.66</td>
</tr>
<tr>
<td>Choice</td>
<td>.67</td>
</tr>
<tr>
<td>Enjoyment</td>
<td>.74</td>
</tr>
</tbody>
</table>

Summary

Robotics, automation, and control curriculum have been used in technology education to include engineering concepts into the curriculum. The LEGO platform enables students in elementary and middle school to experience robotics, automation, and control on a level they can understand. Teachers planning to use robotics in their classroom are required to choose the best programming method for their students. These programming languages can be broken into two main categories, text-based using (e.g., C++, VB, and Fortran) or graphic-based (e.g., RoboLAB, and LabVIEW). In order to help teachers and teacher educators make this decision, the instructor should take into account which method will motivate students and which method will students prefer. The *My Class Activities* instrument provided a measurement tool that could be used to determine the level of motivation and preference toward the two methods of programming. Two age appropriate programming languages that can be used to program the same microcontroller the RCX have been identified. The measurement instrument provided the assessment required to identify student motivation toward the two methods of programming identified in this study, graphic-based and text-based.
CHAPTER III
METHODOLOGY

Introduction

The purpose of this study was to compare middle school students’ motivation and preference toward text-based and graphics-based programming. Students used VB for the Text-based programming portion of this study, and students used RoboLAB for the graphic-based portion of the study. Specifically, this study measured motivation as defined by the dimensions of interest, challenge, choice, and enjoyment. Identified in this chapter are the following: target population, sample, research design, outcomes, measurement instrument, and analysis of data. The two curriculums that have been developed for this study and their implementation are also explained in this section.

Population

The target population for this study was 376 middle school students who were enrolled in a technology education course. The school used for this study was a suburban upper middle class school district with a student-teacher ratio of 14.8. The school population was made up of 86.1% White, 6.6% Black, 6.0% Latino, and 3.3% other. The school had 18 technology education sections taught to seventh grade students during the school year.
Sample

One hundred twenty-two students \((n = 122)\) were selected from the 376 students enrolled in the seventh-grade technology education course to serve as the sample for this study. The school had 18 technology education sections taught to seventh grade students during the school year. The school year is divided into six, 6-week sessions with three technology education classes taught per session. All seventh-grade students are required to participate in this course rotation. Seventh grade students are randomly divided into six groups, then into three classes, who rotate through six courses during the course of the school year, music, art, family and consumer science, keyboarding, computers, and technology education. During the first three sessions, nine classes, the curriculum was piloted and revised. After the curriculum was revised six classes were selected from the fifth and sixth sessions that occur during the spring semester involving 122 students.

Research Design

In each of the six classes selected for this study, students were taught two units on control systems which differed only by the programming language used. For one method of programming students used a graphic-based program called RoboLAB, and for the other method students used a text-based program called VB. Each programming method was taught in a 2-week unit that used the LEGO RCX brick as the controller for assignments. The activities in each curriculum, although slightly different, were designed to be of equal difficulty level. The specific problem solving activities were changed slightly to avoid the redundancy of completing the same activity using the other
To control for the order effects that the students receive exposure to the two programming languages a counterbalanced design was used. Using this counterbalanced design, three of the six classes first learned programming using a text-based approach with VB, and then they were taught a graphic-based approach using RoboLAB. The other three classes started with the graphic-based approach and then were taught the text-based approach. A three factor ANOVA, 2 x 2 x 2, was conducted with one of the factors specifically checking to determine if the order the students received the curriculum had an impact of their motivation scores.

Curriculum

Two curriculums were developed to teach students about programming robots and control systems. The first curriculum used a text-based programming language called VB. The second curriculum used a graphic-based programming language called RoboLAB. The lab activities in each curriculum were designed to be of equal difficulty level, with only specific variables (e.g., time delays and order of commands) within the lab activities being changed slightly to avoid redundancy.

The text-based curriculum used VB to write the program for the microcontroller, the LEGO RCX. The graphic-based curriculum used RoboLAB to write the program for the LEGO RCX. Both programming languages were piloted during the first three sessions (18 weeks) of the school year to ensure they were age and grade appropriate. In both curriculums, students completed six major programs that run on the LEGO RCX.
The six activities built on each other and increased in complexity until the student had received the instruction to complete a program on their own. Along with verbal instructions, the students were given a workbook to complete as they worked throughout the activities. An example chapter from each workbook is included in Appendix A and B.

The only planned difference between the two curriculums was the programming language used to write the program. After the students completed the two curriculum units, they were assigned a team design challenge as a capstone experience and allowed to choose, as a team, between the two programming languages to solve a control problem. This authentic assessment information was recorded and analyzed to determine the teams preferred method of programming. To ensure this information accurately represented individual student preference, each student responded to a written prompt to determine which method of programming they would individually prefer.

In the study, the experimenter also served as the instructor for all six courses, which can bring up questions regarding experimenter bias. In the assumptions it was assumed that both programming methods were taught with minimal teacher bias, additional steps were taken to control for experimenter bias. The students were taught using a tutorial based approach, which meant they followed instructions from a workbook to learn most of the programming. The experimenter had no personal preference toward teaching either programming language. In addition, the experimenter attempted to ensure their impact of the outcome of the study was minimal so the results could be used to guide the choice of programming language that should be taught at the school used in the study.
Curriculum Implementation

To implement the curriculum, students were divided into cooperative teams of three, and students alternated completing each of the following roles, programmer, constructor, and documenter. The programmer was responsible for writing the program and downloading it to the micro-controller/RCX brick. The constructor was responsible for connecting any wires and LEGO blocks to prepare the device for downloading and execution of the program. The documenter/recorder was responsible for keeping record in a portfolio of all the assignments completed during each unit, also the documenter received instructions from the teacher and then was sent back to direct the other members of their team. Each curriculum unit was divided into three sections and each person changed roles for each section. Each team member had the opportunity to perform in each of the three roles.

Outcomes

This study examined two major hypotheses. First, middle school students will be more motivated when using a graphics-based programming language than text-based as measured by the *My Class Activities* survey. The second hypothesis for this study was; middle school students preferred using graphic-based programming more than using text-based programming in an introductory experience. This was measured using individual and group preference information.

A survey instrument titled *My Class Activities* was used to provide a measure of the student perception of motivation while learning each method of programming. This
construct of motivation was defined by four dimensions of interest, challenge, choice, enjoyment. This survey instrument was given to the students upon completion of each robotics unit. The survey consisted of a 31-item survey and took approximately 15 minutes for the students to complete. An ANOVA was conducted to determine which programming method received a higher score in each dimension area and to identify relationships between the factors of order, session, and the scores on the My Class Activities survey.

Team preference was determined by the programming method student teams chose to use to complete the capstone experience. The programming method selected by each team was recorded. The number of teams that chose each language was compared using a chi-square test to determine if the difference in the number of teams who chose each programming method occurred by chance or if one programming method was significantly preferred by teams to the other. To ensure that team preference accurately represented individual student preference, students were asked individually to respond to the prompt “After learning both RoboLAB and VB, which programming language do you prefer to use?” The ratio of individual students that preferred each language was compared to team choice using a chi-square test to confirm that team choice was representative of individual preference.

Survey Instrument

The My Class Activities survey, developed by Marcia Gentry, Ph.D., and Robert Gable, Ed.D, consisted of 31 items and used for grades three through eight, to assess
students’ perception in four areas: interest, challenge, choice, and enjoyment of their classrooms (Gentry & Gable, 2001). These four areas were identified by Gentry and Gable as the major dimensions of the broader construct of motivation. For the purposes of this study, the four dimensions were considered separately to provide the most information to the user about the nature of the classroom. All items on this survey consisted of a 5-point Likert scale. Eight items measured student interest, nine items addressed student challenge, seven addressed student choice, and seven addressed student enjoyment.

In the *My Class Activities* survey, and for the purposes of this study, the terms interest, challenge, choice, and enjoyment were defined as follows. Interest was defined as reflecting positive feelings/preference for certain topics, subject areas, or activities. Challenge was defined as something that engages the student and requires extra effort. Choice was defined as giving the student the right power to select educational options and direct his or her own learning. Enjoyment was defined as providing students with pleasure and satisfaction (Gentry & Gable, 2001).

The 31 items on the survey were printed on two sides of one sheet of paper, front and back, and can be administered to students in 10 to 15 minutes. ID numbers were used in place of names on the survey to ensure confidentiality. In addition to the 31 items, students were asked to respond to the prompt, “After learning both RoboLAB and VB, which programming language do you prefer to use?”
Data Analysis

The data were analyzed using SPSS version 17.0. A 2 x 2 x 2 ANOVA was conducted for each of the dimensions of motivation including interest, challenge, choice, and enjoyment. The first factor in the ANOVA was the programming language used, graphic-based and text-based, to determine which programming method provided the students with a higher level in each category of motivation as defined above. Gentry and Gable (2001) pointed out the importance of separating the four dimensions when he stated, “although the dimensions of interest, challenge, choice, and enjoyment are moderately related, when they are considered separately they provide the most information to the user about the nature of the classroom and the student perceptions” (p. 23). The second factor in the ANOVA was the order the students received the curriculum. One-half of the students received the graphic-based language first and the other half received the text-based language first. The third factor in the ANOVA was the session of the school year the students participated in the study. The curriculum was taught to six classes that occurred during two sessions that cover an 8-week period. One half of the students were taught during the first 4 weeks and the other half were taught in the second 4 weeks.

After the capstone project was completed, the programming language used by each team was compared, using a chi-squared test, to determine team preference. The ratio of individual students that preferred each language was compared to team choice using a chi-square test to confirm that team choice was representative of individual preference.
The purpose of this study was to compare middle school students’ motivation and preference toward text-based and graphics-based programming. Students used VB for the text-based programming portion of this study, and students used RoboLAB for the graphic-based portion of the study. Specifically this study measured motivation as defined by the dimensions of interest, challenge, choice, and enjoyment. Identified in this chapter were the following: target population, sample, research design, outcomes, measurement instrument, and analysis of data. This study examined two major hypotheses. First, middle school students will be more motivated when using a graphics-based programming language than text-based as measured by the *My Class Activities* survey. The second hypothesis for this study was; middle school students preferred using graphic-based programming more than using text-based programming in an introductory experience. Using the statistical analysis described in this section the two research questions will be answered.
CHAPTER IV
FINDINGS

Introduction

The purpose of this study was to compare middle school students’ motivation and preference toward text-based and graphics-based programming. Students used VB for the text-based programming portion of the study, and students used RoboLAB for the graphic-based portion of the study. Motivation was defined by the *My Class Activities* questionnaire using the dimensions of interest, challenge, choice, and enjoyment. Preference was determined through individual student and team choice. This study was conducted with 122 seventh-grade students from three required technology education sections taught at the middle school level.

This study examined two hypotheses. First, middle school students will be more motivated when using a graphics-based programming language than text-based as measured by the *My Class Activities* survey. The second hypothesis for this study was; middle school students preferred using graphic-based programming more than using text-based programming in an introductory experience. Student preference was identified individually and as a team. The findings are presented in the sections below. A copy of the 31-question *My Class Activities* instrument used can be found in Appendix C.

Findings Relevant to Student Motivation

This section addresses the first hypothesis that middle school students will be
more motivated when using a graphics-based programming language than text-based as measured by the *My Class Activities* survey. The *My Class Activities* survey instrument defined motivation by the four dimensions of interest, challenge, choice, and enjoyment. Gentry and Gable (2001) pointed out that “although the dimensions of interest, challenge, choice, and enjoyment are moderately related, when they are considered separately they provide the most information to the user about the nature of the classroom and the student perceptions” (p. 23). The *My Class Activities* survey instrument was used to evaluate student perception in each of these dimensions after completing each programming method.

**Interest**

Gentry and Gable (2001) defined interest as reflecting “positive feelings/preference for certain topics, subject areas, or activities” (p. 4). The first eight questions of the *My Class Activities* survey dealt with the dimension of interest. These items include the following statements.

1. What I do in my class fits my interests.
2. I have an opportunity to work on things in my class that interest me.
3. What I do in my class gives me interesting and new ideas.
4. I study interesting topics in my class.
5. The teacher involves me in interesting learning activities.
6. What I learn in my class is interesting to me.
7. What I do in my class is interesting to me.
8. My class helped me explore my interests.
Table 2 shows the student responses to these first eight items along with the mean and standard deviation for each item. On this set of items, a higher score indicates a higher level of student interest in the activities in the classroom. Table 3 shows the overall mean and standard deviation for each programming method with regard to interest.

Table 2

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Never (1)</th>
<th>Seldom (2)</th>
<th>Sometimes (3)</th>
<th>Often (4)</th>
<th>Always (5)</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>5</td>
<td>19</td>
<td>78</td>
<td>20</td>
<td>3.93</td>
<td>0.69</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>5</td>
<td>19</td>
<td>76</td>
<td>18</td>
<td>4</td>
<td>2.98</td>
<td>0.78</td>
</tr>
<tr>
<td>2. Stem: I have an opportunity to work on things in my class that interest me.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>2</td>
<td>3</td>
<td>29</td>
<td>48</td>
<td>40</td>
<td>3.99</td>
<td>0.90</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>4</td>
<td>28</td>
<td>55</td>
<td>31</td>
<td>4</td>
<td>3.02</td>
<td>0.87</td>
</tr>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>9</td>
<td>36</td>
<td>57</td>
<td>20</td>
<td>3.72</td>
<td>0.83</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>13</td>
<td>36</td>
<td>65</td>
<td>8</td>
<td>0</td>
<td>2.56</td>
<td>0.77</td>
</tr>
<tr>
<td>4. Stem: I study interesting topics in class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>2</td>
<td>9</td>
<td>43</td>
<td>43</td>
<td>25</td>
<td>3.66</td>
<td>0.94</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>12</td>
<td>30</td>
<td>62</td>
<td>18</td>
<td>0</td>
<td>2.70</td>
<td>0.84</td>
</tr>
<tr>
<td>5. Stem: The teacher involves me in interesting learning activities.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>6</td>
<td>27</td>
<td>42</td>
<td>47</td>
<td>4.07</td>
<td>0.90</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>10</td>
<td>35</td>
<td>43</td>
<td>28</td>
<td>6</td>
<td>2.88</td>
<td>1.02</td>
</tr>
<tr>
<td>6. Stem: What I learn in my class is interesting to me.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>7</td>
<td>24</td>
<td>29</td>
<td>32</td>
<td>3.95</td>
<td>0.83</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>7</td>
<td>40</td>
<td>57</td>
<td>18</td>
<td>0</td>
<td>2.70</td>
<td>0.79</td>
</tr>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>6</td>
<td>16</td>
<td>58</td>
<td>42</td>
<td>4.11</td>
<td>0.82</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>5</td>
<td>40</td>
<td>43</td>
<td>30</td>
<td>4</td>
<td>2.90</td>
<td>0.93</td>
</tr>
<tr>
<td>8. Stem: My class has helped me explore my interests.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>23</td>
<td>35</td>
<td>39</td>
<td>25</td>
<td>3.54</td>
<td>1.02</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>6</td>
<td>43</td>
<td>51</td>
<td>18</td>
<td>4</td>
<td>2.76</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Table 3

Mean Scores and Standard Deviations for the Dimension of Interest Items 1-8

Categorized by Programming Language

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>3.87</td>
<td>0.63</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.81</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Challenge

Questions 9-17 on the My Class Activities survey are related to the dimension of Challenge. Gentry and Gable (2001) pointed out, “children show preference for task that are slightly beyond their abilities and that, therefore, intellectual development requires difficult tasks” (p. 2). These items include the following statements.

9. The activities I do in my class are challenging.
10. I have to think to solve problems in my class.
11. I use challenging materials and books in my class.
12. I challenge myself by trying new things.
13. My work can make a difference.
14. I find the work in this class demanding.
15. I am challenged to do my best in class.
16. What we do in class fits my abilities.
17. This class is difficult.

On this set of items, a higher score indicates a higher level of challenge. Table 4 shows the student responses to these nine statements along with the mean and standard deviation for each item. Table 5 shows the overall mean and standard deviation for each programming method with regard to the dimension of challenge.
Table 4

*Frequency, Mean, and Standard Deviation for the Dimension of Challenge Items 9-17*

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Never</th>
<th>Seldom</th>
<th>Sometimes</th>
<th>Often</th>
<th>Always</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
<td>(5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Stem: The activities I do in class are challenging.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>1</td>
<td>14</td>
<td>49</td>
<td>34</td>
<td>24</td>
<td>3.54</td>
<td>0.96</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>5</td>
<td>17</td>
<td>48</td>
<td>33</td>
<td>19</td>
<td>3.36</td>
<td>1.04</td>
</tr>
<tr>
<td>10. Stem: I have to think to solve problems in my class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>3</td>
<td>4</td>
<td>43</td>
<td>48</td>
<td>24</td>
<td>3.7</td>
<td>0.91</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>6</td>
<td>11</td>
<td>53</td>
<td>32</td>
<td>20</td>
<td>3.4</td>
<td>1.03</td>
</tr>
<tr>
<td>11. Stem: I use challenging materials and books in my class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>9</td>
<td>31</td>
<td>37</td>
<td>35</td>
<td>10</td>
<td>3.05</td>
<td>1.08</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>16</td>
<td>33</td>
<td>60</td>
<td>13</td>
<td>0</td>
<td>2.57</td>
<td>0.85</td>
</tr>
<tr>
<td>RoboLAB</td>
<td>6</td>
<td>8</td>
<td>29</td>
<td>43</td>
<td>36</td>
<td>3.78</td>
<td>1.09</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>9</td>
<td>34</td>
<td>55</td>
<td>19</td>
<td>5</td>
<td>2.81</td>
<td>0.93</td>
</tr>
<tr>
<td>13. Stem: My work can make a difference.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>12</td>
<td>50</td>
<td>42</td>
<td>18</td>
<td>3.54</td>
<td>0.86</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>9</td>
<td>44</td>
<td>35</td>
<td>23</td>
<td>11</td>
<td>2.86</td>
<td>1.09</td>
</tr>
<tr>
<td>14. Stem: I find the work in this class demanding.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>9</td>
<td>21</td>
<td>35</td>
<td>43</td>
<td>14</td>
<td>3.26</td>
<td>1.10</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>4</td>
<td>27</td>
<td>49</td>
<td>33</td>
<td>9</td>
<td>3.13</td>
<td>0.95</td>
</tr>
<tr>
<td>15. Stem: I am challenged to do my best in class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>5</td>
<td>2</td>
<td>25</td>
<td>55</td>
<td>35</td>
<td>3.93</td>
<td>0.96</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>4</td>
<td>30</td>
<td>53</td>
<td>33</td>
<td>2</td>
<td>2.99</td>
<td>0.85</td>
</tr>
<tr>
<td>RoboLAB</td>
<td>1</td>
<td>10</td>
<td>44</td>
<td>36</td>
<td>31</td>
<td>3.70</td>
<td>0.97</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>10</td>
<td>30</td>
<td>59</td>
<td>17</td>
<td>6</td>
<td>2.82</td>
<td>0.94</td>
</tr>
<tr>
<td>17. Stem: This class is difficult.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoboLAB</td>
<td>28</td>
<td>28</td>
<td>30</td>
<td>18</td>
<td>18</td>
<td>2.75</td>
<td>1.36</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>14</td>
<td>35</td>
<td>33</td>
<td>28</td>
<td>12</td>
<td>2.91</td>
<td>1.17</td>
</tr>
</tbody>
</table>
Table 5

Mean Scores and Standard Deviations for the Dimension of Challenge Items 9-17

Categorized by Programming Language

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>3.47</td>
<td>0.53</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.98</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Choice

The next seven questions, 18-24, address the dimension of Choice. Gentry and Gable (2001) stated, “Students are engaged in meaningful learning when they are involved in projects about which they care deeply and that they choose to pursue” (p. 3). These items include the following statements:

18. I can choose to work in a group.

19. I can choose to work alone.

20. When we work together, I can choose my partners.

21. I can choose my own projects.

22. When there are many jobs, I can choose the ones that suit me.

23. I can choose materials to work with in the class.

24. I can choose an audience for my product.

On this set of items, a higher score indicates a higher level of choice. Table 6 shows the student responses to these nine statements along with the mean and standard deviation for each item. Table 7 shows the overall mean and standard deviation for each programming method with regard to the dimension of choice.
Table 6

*Frequency, Mean, and Standard Deviation for the Dimension of Choice Items 18-24*

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Never (1)</th>
<th>Seldom (2)</th>
<th>Sometimes (3)</th>
<th>Often (4)</th>
<th>Always (5)</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>8</td>
<td>10</td>
<td>16</td>
<td>27</td>
<td>61</td>
<td>4.01</td>
<td>1.25</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>30</td>
<td>32</td>
<td>42</td>
<td>11</td>
<td>7</td>
<td>2.45</td>
<td>1.13</td>
</tr>
</tbody>
</table>

18. Stem: I can choose to work in a team.

19. Stem: I can choose to work alone.

20. Stem: When we work together, I can choose my partners.

21. Stem: I can choose my own projects.

22. Stem: When there are many jobs, I can choose the ones that suit me.

23. Stem: I can choose materials to work with in class.

24. Stem: I can choose an audience for my product.

Table 7

*Mean Scores and Standard Deviations for the Dimension of Choice Items 18-24*

*Categorized by Programming Language*

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>3.25</td>
<td>0.80</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.32</td>
<td>0.67</td>
</tr>
</tbody>
</table>
Enjoyment

The last seven questions, 25-31, measure the dimension of Enjoyment. This dimension is especially important because “the best learning occurs when children enjoy what they are doing” (Gentry & Gable, 2001, p. 4). These items include the following statements.

25. I look forward to my class.
26. I have fun in my class.
27. The teacher makes learning fun.
28. I like what I do in my class.
29. I like working in a class.
30. The activities I do in my class are enjoyable.
31. I like the projects I work on in my class.

On this set of items, a higher score indicates a higher level of enjoyment. Table 8 shows the student responses to these nine statements along with the mean and standard deviation for each item. Table 9 shows the overall mean and standard deviation for each programming method with regard to the dimension of enjoyment.

Findings Relevant to Student Preference

The second hypothesis of the study was that middle school students prefer using graphic-based programming more than using text-based programming in an introductory experience when given the choice. The students had two opportunities to demonstrate a preference toward one particular method of programming the RCX. To identify team
Table 8

Frequency, Mean, and Standard Deviation for the Dimension of Enjoyment Items 25-31

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Never (1)</th>
<th>Seldom (2)</th>
<th>Sometimes (3)</th>
<th>Often (4)</th>
<th>Always (5)</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>0</td>
<td>5</td>
<td>13</td>
<td>47</td>
<td>57</td>
<td>4.28</td>
<td>0.82</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>7</td>
<td>31</td>
<td>56</td>
<td>28</td>
<td>0</td>
<td>2.86</td>
<td>0.84</td>
</tr>
</tbody>
</table>

25. Stem: I look forward to my class.

26. Stem: I have fun in my class.

27. Stem: The teacher makes learning fun.


29. Stem: I like working in a class.

30. Stem: The activities I do in my class are enjoyable.

31. Stem: I like the projects I work on in my class.

Table 9

Mean Scores and Standard Deviations for the Dimension of Enjoyment Items 25-31

Categorized by Programming Language

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>4.36</td>
<td>0.71</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.84</td>
<td>0.78</td>
</tr>
</tbody>
</table>
preference, teams were assigned a capstone project during which team members were allowed to choose which method they would use to program the RCX. To identify individual preference students were given an opportunity, after both units had been taught, to record a written response to the prompt, “After learning both RoboLAB and VB, which programming language do you prefer to use?” Individual choice was compared with group choice to ensure group dynamics did not have an effect on individual choice.

The first method used to identify student preference was to observe team choice of which programming method they used on a capstone project at the end of the robotics units. Table 10 shows 35 of the 40 teams (87.5%) chose to use RoboLAB as their programming language when programming their capstone project and only 5 of 40 teams (12.5%) chose to use VB.

The second method used to identify student preference was to identify individual preference through the use of student response to the prompt, “After learning both RoboLAB and VB, which programming language do you prefer to use?” While student were completing the second survey instrument sheet they were asked to write a response to the prompt and hand it in with the second survey. Table 11 shows 105 of 122 (86.1%)

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Frequency</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>35</td>
<td>87.5</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>5</td>
<td>12.5</td>
</tr>
<tr>
<td>Total</td>
<td>40</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Table 11

*Frequency Table Showing Individual Student Preference*

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Frequency</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>105</td>
<td>86.1</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>17</td>
<td>13.9</td>
</tr>
<tr>
<td>Total</td>
<td>122</td>
<td>100.0</td>
</tr>
</tbody>
</table>

students indicated they prefer to use RoboLAB to program the RCX with only 17 of 122 (13.9%) indicating they would rather use VB.

Summary

The purpose of this study was to compare seventh-grade students’ motivation and preference toward text-based and graphics-based programming. Students used VB for the text-based programming portion of this study, and students used RoboLAB for the graphic-based portion of the study. Motivation was defined by the *My Class Activities* questionnaire using the dimensions of interest, challenge, choice, and enjoyment. Preference was determined through individual student response and team choice.

This study has examined two hypotheses. First, middle school students will be more motivated when using a graphics-based programming language than text-based as measured by the *My Class Activities* survey. The *My Class Activities* survey uses for dimensions of interest, challenge, choice, and enjoyment that contribute to student motivation. The second hypothesis for this study was; middle school students prefer using graphic-based programming more than using text-based programming in an introductory experience. Student preference was identified individually and as a team.
The most important finding to note is the mean scores in each dimension of motivation are higher for the graphic-based programming interface. Also, by examining both methods of measuring preference, students chose the graphic-based programming interface more often than the text-based programming interface.
CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Introduction

The purpose of this study was to compare seventh grade students’ motivation and preference toward text-based programming using VB, and graphics-based programming using RoboLAB. Motivation was defined by the My Class Activities questionnaire using the dimensions of interest, challenge, choice, and enjoyment. Preference was determined through team and individual student choice. This study was conducted with a total of 122 students from three 6-week middle school technology education classes.

This study examined two hypotheses. First, middle school students will be more motivated when using a graphics-based programming language than text-based as measured by the My Class Activities survey. The second hypothesis for this study was, middle school students prefer using graphic-based programming more than using text-based programming in an introductory experience. Student preference was identified through both a team and individual selection process. Identified in this chapter are the following conclusions with regard to motivation, conclusions with regard to preference, and recommendations for teachers and further study.

Conclusions Regarding Motivation

The first hypothesis was that students will be more motivated when using a graphics-based programming language than text-based as measured by the My Class
Activities survey. The My Class Activities survey instrument defines motivation by using the four dimensions of interest, challenge, choice, and enjoyment. To determine whether students preferred one method of programming over the other four separate 2 x 2 x 2 ANOVAs were conducted. The first factor determined differences in student perception for each of the four dimensions of motivation. The second factor in the ANOVA determined differences in the order the students received the curriculum; half of the students learned RoboLAB first and the others learned VB first in this experimental counter balanced research design. The third factor in the ANOVA determined differences between the sessions the students were in, as the study was conducted during two sessions in the school year each occurring within six weeks of the other.

Interest

A 2 x 2 x 2 ANOVA was conducted to evaluate student perception of interest on the effects of two curriculum units using two programming interfaces, the order the curriculums were received, and the session the students received the curriculums. The first eight questions of the My Class Activities survey dealt with the dimension of interest. These items include the following statements.

1. What I do in my class fits my interests.
2. I have an opportunity to work on things in my class that interest me.
3. What I do in my class gives me interesting and new ideas.
4. I study interesting topics in my class.
5. The teacher involves me in interesting learning activities.
6. What I learn in my class in interesting to me.
7. What I do in my class is interesting to me.
8. My class helped me explore my interests.

The means and standard deviations for interest as a function of curriculum received are presented in Table 12. The results of the ANOVA (see Table 13), show the students had a significantly higher interest score on the RoboLAB unit with regard to the interest dimension of the My Class Activities survey $F(1, 122) = 15.15, p < .001$, partial $\eta^2 = .113$. According to Cohen (2008), and effect size of .8 or greater is considered large, .2 or lower is considered small, and .5 is considered a medium effect size. The curriculum main effect indicated that students tend to have significantly higher interest in graphics-based programming than text-based. Also, results of the ANOVA (Table 13) show that the order the students received the curriculum did not have a significant effect of the level of interest toward the programming languages $F(1, 119) = .009, p = .923$, partial $\eta^2 < .001$. Additionally, results of the ANOVA, see table 14, show the session the students were involved in the study did not have a significant effect on the level of interest $F(1, 119) = .097, p = .756$, partial $\eta^2 = .001$. From the results of the ANOVA the students indicated a higher interest level when using a graphic programming interface than a text-based interface regardless of order and session the students received the curriculum.

Table 12

Mean and Standard Deviations for Interest in Both Programming Languages

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>3.92</td>
<td>0.55</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.81</td>
<td>0.67</td>
</tr>
</tbody>
</table>
Table 13

2 x 2 x 2 ANOVA Showing the Results for the Factors of Curriculum, Order, and Session with Regard to the Dimension of Interest

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III sum of squares</th>
<th>df</th>
<th>Mean square</th>
<th>F</th>
<th>Sig.</th>
<th>Partial Eta squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curriculum</td>
<td>5.369</td>
<td>1</td>
<td>5.37</td>
<td>15.15</td>
<td>&lt;.001</td>
<td>.113</td>
</tr>
<tr>
<td>Curriculum * Session</td>
<td>0.034</td>
<td>1</td>
<td>0.03</td>
<td>0.097</td>
<td>0.756</td>
<td>.001</td>
</tr>
<tr>
<td>Curriculum * Order</td>
<td>0.003</td>
<td>1</td>
<td>0.00</td>
<td>0.009</td>
<td>0.923</td>
<td>.000</td>
</tr>
<tr>
<td>Error(Language)</td>
<td>42.155</td>
<td>119</td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Challenge

A 2 x 2 x 2 ANOVA was conducted to evaluate student perception of challenge on the effects of two curriculum unit using two programming interfaces, the order the curriculums were received, and the session the students received the curriculums. Questions 9-17 of the My Class Activities survey dealt with the dimension of challenge. These items include the following statements.

9. The activities I do in my class are challenging.
10. I have to think to solve problems in my class.
11. I use challenging materials and books in my class.
12. I challenge myself by trying new things.
13. My work can make a difference.
14. I find the work in this class demanding.
15. I am challenged to do my best in class.
16. What we do in class fits my abilities.
17. This class is difficult.
The means and standard deviations for challenge as a function of curriculum are presented in Table 14. The results of the ANOVA (Table 15) show the students had a significantly higher challenge score on the RoboLAB unit with regard to the challenge dimension of the *My Class Activities* survey $F(1, 119) = 58.66, p < .001$, partial $\eta^2 = .332$. The curriculum main effect indicated that students tend to have significantly higher challenge in graphics-based programming than text-based. Results of the ANOVA (see Table 14) show that the order the students received the curriculum did not have a significant effect of the level of challenge toward the programming languages $F(1, 119) = .1.72, p = .192$, partial $\eta^2 < .014$. Additionally, results of the ANOVA, see table 15, show the session the students were involved in the study did not have a significant effect on the level of challenge $F(1, 119) = .1.036, p = .311$, partial $\eta^2 = .009$. From the results of the ANOVA the students indicated a higher challenge level when using a Graphic programming interface than a text-based interface regardless of order and session the students received the curriculum.

*Choice*

A 2 x 2 x 2 ANOVA was conducted to evaluate student perception of choice on the effects of two curriculum unit using two programming interfaces, the order the

Table 14

*Mean and Standard Deviations for Challenge in Both Programming Languages*

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>3.47</td>
<td>0.53</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.98</td>
<td>0.45</td>
</tr>
</tbody>
</table>
curriculums were received, and the session the students received the curriculums.

Questions 18-24 of the *My Class Activities* survey dealt with the dimension of choice. These items include the following statements:

18. I can choose to work in a group.
19. I can choose to work alone.
20. When we work together, I can choose my partners.
21. I can choose my own projects.
22. When there are many jobs, I can choose the ones that suit me.
23. I can choose materials to work with in the class.
24. I can choose an audience for my product.

The means and standard deviations for choice as a function of curriculum are presented in Table 16. The results of the ANOVA (see Table 17) show the students had a significantly higher choice score on the RoboLAB unit with regard to the choice dimension of the *My Class Activities* survey $F(1, 119) = 95.154, p < .001$, partial $\eta^2 = .446$. The curriculum main effect indicated that students tend to have significantly higher
Table 16

*Mean and Standard Deviations for Choice in Both Programming Languages*

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>3.25</td>
<td>0.80</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.32</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Table 17

*2 x 2 x 2 ANOVA Showing the Results for the Factors of Curriculum, Order, and Session with Regard to the Dimension of Choice*

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III sum of squares</th>
<th>df</th>
<th>Mean square</th>
<th>F</th>
<th>Sig.</th>
<th>Partial Eta squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curriculum</td>
<td>47.952</td>
<td>1</td>
<td>47.95</td>
<td>95.154</td>
<td>&lt; .001</td>
<td>.446</td>
</tr>
<tr>
<td>Curriculum * Session</td>
<td>1.1</td>
<td>1</td>
<td>1.10</td>
<td>2.182</td>
<td>.142</td>
<td>.018</td>
</tr>
<tr>
<td>Curriculum * Order</td>
<td>.553</td>
<td>1</td>
<td>0.55</td>
<td>1.098</td>
<td>.297</td>
<td>.009</td>
</tr>
<tr>
<td>Error(Language)</td>
<td>59.465</td>
<td>119</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

choice in graphics-based programming than text-based. Results of the ANOVA, see table 17, show that the order the students received the curriculum did not have a significant effect of the level of choice toward the programming languages $F(1, 119) = 2.182, p = .142$, partial $\eta^2 <.018$. Additionally, results of the ANOVA, see table 14, show the session the students were involved in the study did not have a significant effect on the level of choice $F(1, 119) = 1.098, p = .297$, partial $\eta^2 = .009$. From the results of the ANOVA the students indicated a higher choice level when using a graphic programming interface than a text-based interface regardless of order and session the students received the curriculum.
Enjoyment

A 2 x 2 x 2 ANOVA was conducted to evaluate student perception of enjoyment on the effects of two curriculum unit using two programming interfaces, the order the curriculums were received, and the session the students received the curriculums. Questions 18-24 of the *My Class Activities* survey dealt with the dimension of enjoyment. These items include the following statements:

25. I look forward to my class.
26. I have fun in my class.
27. The teacher makes learning fun.
28. I like what I do in my class.
29. I like working in a class.
30. The activities I do in my class are enjoyable.
31. I like the projects I work on in my class.

The means and standard deviations for enjoyment as a function of curriculum is presented in Table 18. The results of the ANOVA (see Table 19) show the students had a significantly higher enjoyment score on the RoboLAB unit with regard to the enjoyment dimension of the *My Class Activities* survey $F(1, 119) = 324.392, p < .001$, partial $\eta^2 = .733$. The curriculum main effect indicated that students tend to have significantly higher enjoyment in graphics-based programming than text-based. In addition, results of the ANOVA (Table 19) show that the order the students received the curriculum did not have a significant effect of the level of enjoyment toward the programming languages $F(1, 119) = .002, p = .967$, partial $\eta^2 < .001$. Results of the ANOVA show the session the
Table 18

**Mean and Standard Deviations for Enjoyment in Both Programming Languages**

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>3.25</td>
<td>0.80</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>2.32</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Table 19

**2 X 2 X 2 ANOVA Showing the Results for the Factors of Curriculum, Order, and Session with Regard to the Dimension of Enjoyment**

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III sum of squares</th>
<th>df</th>
<th>Mean square</th>
<th>F</th>
<th>Sig.</th>
<th>Partial Eta squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curriculum</td>
<td>126.754</td>
<td>1</td>
<td>126.75</td>
<td>324.392</td>
<td>&lt;.001</td>
<td>.733</td>
</tr>
<tr>
<td>Curriculum * Session</td>
<td>0.001</td>
<td>1</td>
<td>0.00</td>
<td>.002</td>
<td>.967</td>
<td>.000</td>
</tr>
<tr>
<td>Curriculum * Order</td>
<td>0.064</td>
<td>1</td>
<td>0.06</td>
<td>.163</td>
<td>.687</td>
<td>.001</td>
</tr>
<tr>
<td>Error(Language)</td>
<td>49.108</td>
<td>119</td>
<td>0.39</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

students were involved in the study did not have a significant effect on the level of enjoyment $F(1, 119) = .163, p = .687$, partial $\eta^2 = .001$. From the results of the ANOVA the students indicated a higher enjoyment level when using a graphic programming interface than a text-based interface regardless of order and session the students received the curriculum.

Conclusions Regarding Preference

The second hypothesis for this study was; middle school students prefer using graphic-based programming more than using text-based programming in an introductory experience. The method used to identify team preference was to observe each teams
choice as to which programming method they used on a capstone project at the end of the robotics units. Table 20 shows 35 of the 40 teams (87.5%) chose to use RoboLAB as their programming language when programming their capstone project and only 5 of the 40 teams, (12.5%) chose to use VB.

Team Preference

A one sample chi-squared test was conducted to assess whether student teams prefer to use RoboLAB over VB. The results of the test were significant, $\chi^2 (1, N = 122) = 69.377, p < .001$. The portion of student teams who chose graphic-based programming ($P = .87$) was much greater than the expected proportion of .50, while the proportion of student teams who chose text-based programming ($P = .12$) was much lower than the expected proportion of .50. Overall, these results suggest that students significantly prefer to use a graphic-based programming method to a text-based method.

Individual Preference

The individual student preference was identified using student response to the prompt, “After learning both RoboLAB and VB, which programming language do you prefer to use?” Table 21, shows 105 of 122 (86.1%) of the students indicated they prefer

Table 20

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Frequency</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>35</td>
<td>87.5</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>5</td>
<td>12.5</td>
</tr>
<tr>
<td>Total</td>
<td>40</td>
<td>100</td>
</tr>
</tbody>
</table>
Table 21

*Frequency Table Showing Individual Student Preference*

<table>
<thead>
<tr>
<th>Programming language</th>
<th>Frequency</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoboLAB</td>
<td>105</td>
<td>86.1</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>17</td>
<td>13.9</td>
</tr>
<tr>
<td>Total</td>
<td>122</td>
<td>100</td>
</tr>
</tbody>
</table>

to use RoboLAB to program the RCX with only 17 (13.9%) indicating they would rather use VB.

An additional chi-squared test was conducted to determine whether the proportion of individual students who preferred to use each programming interface differed significantly from the proportion of teams who preferred to use each interface. The results of the tests were not significant, \( \chi^2 (1, N = 122) = .304, p = .518 \). The proportion of students who indicated a preference for graphic based programming \( P = .87 \) was not significantly different than the proportion reported in the group preference of .87 while the proportion of student teams who chose text-based programming \( P = .12 \) was not significantly different than the proportion reported in the group preference of .12. Overall, these results suggest that individual preference and group preference are not significantly different.

**Recommendations for Teachers**

The results of this study indicate that students prefer, and are more motivated when using, graphic-based programming interfaces over text-based. This is important for
teachers who teach control systems. Teacher should consider the following recommendations. Using the information from a study like this could help educators in selecting a programming language to teach in their classrooms. As educators select, adapt and implement their curriculum, they need to determine which programming method is best for what they teach. Graphic-based programming should be used for teaching control systems and robotics at the middle school level. Programming logic should be taught regardless of the programming language so students are taught concepts that can transfer between many programming languages. Technology and Engineering educators at the middle school level should focus their control systems programming on methods that motivate students to increase retention.

Recommendations for Further Study

The following are recommendations for further study. A similar study could be conducted to replicate this study with additional schools over a larger geographical area. This could assist in determining whether the results of this study are more generalized to the broader population. A similar study could be conducted at a high school or collegiate level to determine if student preference is the same throughout grade levels. Two programming languages that could be used are C++ for text-based and LabVIEW for graphic-based programming. Both of these are languages that are age appropriate for the high school and college level students and are used to program control systems. A study could be conducted to determine which programming method provides students with a better understanding and knowledge of programming control systems. A study could be
conducted to determine which methods teachers and professors prefer to use to determine if age and experience may play a role in programming language used. This would confirm whether teachers taught text-based programming because that is what they learned or because that is what they prefer to use. Gender could be added as an additional factor in the ANOVA to investigate any difference between male/female responses. With the issues of attracting females to engineering it may be beneficial to see if a gender difference exists.

Summary

The purpose of this study was to compare seventh-grade students’ motivation and preference toward text-based programming using VB, and graphics-based programming using RoboLAB. Motivation was defined by the My Class Activities questionnaire using the dimensions of interest, challenge, choice, and enjoyment. Preference was determined through team and individual student choice. This study was conducted with a total of 122 students from three 6-week technology education classes taught at a middle school.

The results from each dimension of the My Class Activities survey indicate that students are significantly more motivated ($p < .001$) for all four dimensions of motivation, when using a graphic-based programming interface than a text-based interface. Additionally, a significant majority of students 87% indicated they preferred to use a graphic based programming interface when given a choice. The results in this chapter lend support to both hypotheses of this study. Therefore, middle school students are more motivated when using a graphics-based programming language than text-based
as measured by the *My Class Activities* survey, and middle school students prefer using graphic-based programming more than using text-based programming in an introductory experience.
REFERENCES


APPENDICES
Appendix A

Visual Basic Curriculum Example
Unit 2: Control Technology

Lesson 2.1: Introduction to RCX and Visual Basic

Objectives:
- Work effectively in a cooperative learning environment.
- Explain the characteristics and functions of the On-Off and Run buttons on the RCX.
- Explain how an output works.
- Connect the Lego USB Infrared Transmitter to the computer.
- Download and run a program on the RCX.
- Write a simple program in Visual Basic to control the RCX using the following commands:
  - RCX.On A
  - RCX.Off A
  - RCX.Wait 2, 500
  - RCX.BeginOfTask 1
  - RCX.EndOfTask 1

Robotics Command eXplorer (RCX)

The RCX is a microcontroller that will be used to run the programs written in Visual Basic. The most essential feature to be familiar with is the red On-Off button. This button turns the RCX on. When it is on, numbers appear on the LCD display in the center of the RCX. It should look similar to the picture below.

The next feature to be familiar with is the green Run button. Use this button to run the program after it has been downloaded to the RCX. When you are running a program the Lego man next to the numbers with appear to walk, that is how you know your program is running. There are two other buttons on the RCX, a black View button and a grey Prgm button. You will learn more about these buttons in later activities.

Outputs

There are three outputs available on the RCX. On the RCX they are colored black, and labeled A, B, and C as pictured below. The outputs on the RCX simply supply the device connected with a voltage. Use a standard Lego wire connector to connect any output device to the RCX brick. Multiple Lego wire connectors may be connected to
one output. It is important to be sure that the output your wire is connected to is the output you specified in the program.

**Programming in Visual Basic**

Visual Basic is a text-based programming environment, and can be used to write programs for the RCX. The software uses an syntax-based environment, which makes it easy and fun to use. The commands will be typed in sequence and sent by an infrared transmitter to the RCX brick microcontroller.

**Step-By-Step Programming Example**

In the example that follows, Visual Basic will be used to write a simple program to be sent to the RCX. Specifically, the program will perform the following functions in the sequence outlined below:

- Turn output A on
- Wait for 5 seconds
- Turn output A off

The following syntax commands will be necessary to complete the program:

- RCX.BeginOfTask
- RCX.On Motor_A
- RCX.Wait 2, 500
- RCX.Off Motor_A
- RCX.EndOfTask
Step 1—From the Windows desktop click on the icon on the desktop titled “Visual Basic” to open a preexisting file that will be used as a starter file for programming the RCX. The program should open and should look like the graphic below:

This is the program you will need to start all of the programs for the RCX using Visual Basic.

Step 2—Double click on the grey “Send Program to the RCX” button on the screen. A new window will appear showing the syntax section of the program. This programming window should look like the graphic below. Highlight and erase the words program here, this will be where you will type your program.
Visual Basic is made up of two main windows. The one we are looking at is the **Code** window, and the other window with the buttons on it is the **Form** window. You will be using the Code window to create your program and the Form window to send it to the RCX.

**Step 3**—To use Visual Basic to program the RCX you will use an Active X control that contains all of the commands the RCX can understand. To see these commands you will need to begin typing in “RCX.” Once you type the period after the letters RCX a menu will appear showing you the commands you may use. Scroll down and look at all of the commands that you can use.

The first command you will use in every program is the “BeginOfTask” command. Also the last command you will use in each program is the “EndOfTask” command.

- RCX.BeginOfTask 1
- RCX.EndOfTask 1

Type these commands in and press enter on your keyboard to create a space between the commands. If Visual Basic gives you an error message, check to ensure your command was typed correctly. Any commands you add to your program must be between these two commands. Your program should look like the graphic below.
Step 4—Now let’s look back to what we want the RCX to do. Our program needs to:
- Turn output A on
- Wait for 5 seconds
- Turn output A off

Type in the letters RCX and then a period and scroll down to find a command you think will allow you to turn output A on. The command you will use is the “On” command. Now you cannot just say on and have the RCX know which output you want so you need to explain it and specify which output you want to turn on. Your command to turn output A on is:

RCX.On A

Your program should now look like the graphic below.

Step 5—Now just for fun we are going to see how your program is doing so far by sending it to the RCX and running the program. Find the Lego USB tower in your kit. The tower is already connected to the computer using the USB port. Now we can work on attaching the lamp to the output.

Earlier you learned that the output simply sends a designated amount of voltage out a designated port. From our program we see we are using output A. Connect a white Lego lamp to output A as shown below. Be sure it is only connected to output A and not A and B.
Step 6—Place the RCX approximately six to twelve inches from the Lego tower with the tower facing the infrared receiver on the RCX as shown below. Now you are ready to send the program. Be sure the RCX is turned on.

In Visual Basic find the Run menu in the main toolbar across the top of the screen and select “Start.” A slightly different Form window will appear on your screen. Click on the “Connect” button to ensure you are able to connect to the RCX. If you get an error, be sure the RCX is on and try again. When the program has been sent and received you will hear a conformation sound from the RCX, then click Disconnect before you run your program.

Step 7—To test your program simply press the green run button on the RCX and see if the Lego lamp you connected to output A turns on. If this happens your program is correct so far, however remember we need to only have the light on for 2 seconds. When you see your light is on you will need to press the red “On-Off” button on the RCX to turn it off.

Step 6—We can now finish our program. As a reminder the first program we will write will perform the following functions in the sequence outlined below.

- Turn output A on
- Wait for 5 seconds
- Turn output A off
We have the first step, turning output A on. Now we will finish our program. Close the Form window on your screen and the Code window should appear. Because computers cannot assume what we want we need to program in every step. If you were told to “turn the light off for two seconds” you would probably go turn the switch wait for two seconds and then turn it back on. The RCX needs to be told every step.

The next step is to find a command that will cause the RCX to “Wait,” do nothing, for two seconds. Type in the letters RCX and then a period and scroll down until you see the wait command as shown below.

After selecting the “Wait” command you will need to tell the RCX how long you want it to wait for. To do this type a space and then “2, 500.” The two tells the RCX it will count, and the 500 is how high the RCX will count. It takes about 1 second for the RCX to count to 100 so for 5 seconds we use 500. So your program should look like the one in the graphic below.

Step 7—It would not do any good to test to see if your program told the RCX to wait for 5 seconds because it has not been told to do anything after it waits. We need to give it another command. If you look back you will see the next step is to turn off output A. it
will be much like turning it on except you will use the command “Off.” Type the following command:
RCX.Off A

**Step 9**—Now your program should be complete. Find the Lego USB tower in your kit. Connect the RCX to the computer and be sure the light is still connected and place the RCX approximately six to twelve inches from the Lego tower with the tower facing the infrared receiver on the RCX as shown below. Now you are ready to send the final program. Be sure the RCX is turned on.

![Image of RCX and Lego tower](image.jpg)

In Visual Basic find the *Run* menu in the main toolbar across the top of the screen and select “Start.” A slightly different *Form* window will appear on your screen. Click on the “Connect” button to ensure you are able to connect to the RCX. If you get an error, be sure the RCX is on and try again. After the program has been sent and received, you will hear a conformation sound from the speaker on the RCX. After hearing the sound click the “Disconnect” button before you run your program.

**Step 10**—To test your program simply press the green run button on the RCX and see if the Lego lamp you connected to output A turns on for two seconds and then off. If this happens you have completed the program.

**Team Assignment Challenge**

As a team, write a program to control the outputs on the RCX. First, turn output A on for 2 seconds, then output B on for 1 second, then output C on for 5 seconds. Save the program and exit Visual Basic when completed. Remember you will need to connect a motor or lamp to outputs A, B, and C.
Review Questions

1. An output sends information to the microcontroller. True or False
2. How many icons can be used to send electricity from an output? ______
3. What does RCX stand for? __________________  __________________
   ____________________.
4. How many outputs are available on the RCX? ______
5. How can you tell if your RCX is turned on? _______________________
6. What happens if the program you have written has errors?
   _______________________________________________________________
7. Can you send a program with errors to the RCX? ______
8. What happens after the RCX finishes receiving the program?
   _______________________________________________________________
9. How can you tell whether the program on your RCX is running?
   _______________________________________________________________
10. How far away from the tower should your RCX be when sending a program?
    ____ to ____ Inches
Appendix B

RoboLAB Curriculum Example
Lesson 2.1: Introduction to RCX and RoboLAB

Objectives:
- Work effectively in a cooperative learning environment.
- Explain the characteristics and functions of the On-Off and Run buttons on the RCX.
- Explain how an output works.
- Connect the Lego USB Infrared Transmitter to the computer.
- Download and run a program on the RCX.
- Write a simple program in RoboLAB to control the RCX using the following commands:

```
Begin
Motor A Forward
Wait for 2 sec.
Stop A
End
```

Robotics Command eXplorer (RCX)

The RCX is a microcontroller that will be used to run the programs written in RoboLAB. The most essential feature to be familiar with is the red On-Off button. This button turns the RCX on. When it is on, numbers appear on the LCD display in the center of the RCX. It should look similar to the picture below.

The next feature to be familiar with is the green Run button. Use this button to run the program after it has been downloaded to the RCX. When you are running a program the Lego man next to the numbers with appear to walk, that is how you know your program is running. There are two other buttons on the RCX, a black View button and a grey
Prgm button. You will learn more about these buttons in later activities.

**Outputs**
There are three outputs available on the RCX. On the RCX they are colored black, and labeled A, B, and C as pictured below. The outputs on the RCX simply supply the device connected with a voltage. Use a standard Lego wire connector to connect any output device to the RCX brick. Multiple Lego wire connectors may be connected to one output. It is important to be sure that the output your wire is connected to is the output you specified in the program.

**Programming in RoboLAB**

RoboLAB is a graphic programming environment based on LabVIEW, a powerful programming graphic language developed by National Instruments, and is used to write programs for the RCX. The software uses an icon-based, diagram building environment, which makes it easy and fun to use. The icons will be strung together in sequence and sent by an infrared transmitter to the RCX brick microcontroller.

**Step-By-Step Programming Example**

In the example that follows, RoboLAB will be used to write a simple program to be sent to the RCX. Specifically, the program will perform the following functions in the sequence outlined below:

- Turn output A on
- Wait for 2 seconds
- Turn output A off

The following graphical commands will be necessary to complete the program:

- Begin
- Lamp A
- Wait for 2 sec.
- Stop A
- End
Step 1—From the Windows desktop select Start, select All Programs, select ROBOLAB 2.5.4, and select ROBOLAB 2.5.4 from the list of programs. The following screen will appear. Select the programmer button.

Step 2—After selecting the Programmer button the following screen will appear. Double click on the “Inventor 4” line.
Step 3—After you double click the “Inventor 4” line, the following screen will appear. Three main windows will appear. The Front Panel, the Block Diagram, and the Functions Palette. From this point you will set up your programming environment for optimal use. Click on the Front Panel window and minimize it. Then click and drag the Functions Palette window to the side, and maximize the Block Diagram window.

After you complete the last task your screen will look similar to the one below. Now we have two more windows to open before our environment is set up.
Step 4—From the main menu that extends across the top of the programming environment of the Block Diagram select Window and scroll down and select Show Tools Palette.

Next from the main menu that extends across the top of the programming environment of the Front Panel select Help and scroll down and select Show Context Help. After you click Show Tools Palette and Show Context Help you now have four visible windows on your screen, the Functions Palette, the Tools Palette, the Show Context Help, and the Block Diagram which is the main window and takes up most of the screen.
Step 5—Arrange the four windows so that you can work with each one. Below is an example of a screen set up to work with all four windows.

The Block Diagram is the base window where icons may be placed to write the program. When it appears it already has two icons on it a Begin icon and an End icon, all other icons in the program must be placed between these two icons.

The Functions Palette is where all the functions you will use to write your program are located. Each icon represents a different function. The top seven rows of icons are the most common used functions, where the bottom five rows of icons lead you to other sub-palettes where you can find more complex functions.

The Tools Palette is used to change the type of function your mouse performs. The most common tools are the Position/Size/Select tool (the arrow) and the Connect Wire tool (spool of wire). When working on the block diagram you can switch between these two tools by pressing the space bar. The other tools on the palette will be use in later activities.

The Context Help window can be very helpful. As you select various programming icons from the Functions Palette an example graphic of that icon will appear in the Context Help window showing wire connections and explaining the icon’s function. At the bottom of the window you will also see a link you can click for more information about using the selected icon.
Step 6—We can now begin programming. As a reminder the first program we will write will perform the following functions in the sequence outlined below.

- Turn output A on
- Wait for 2 seconds
- Turn output A off

First, every program must start with a **Begin** icon. This signifies where the beginning of your program is. When you opened RoboLAB the **Block Diagram** already contains a **Begin** icon. The icon looks like a green traffic light.

Now that you know where your program will begin you need to add the next step which is to turn output A on. There are three icons on the **Functions Palette** that can send electricity out from output A.

- **Motor A Forward**
- **Motor A Reverse**
- **Lamp A**

The icon you use will depend on the output you are looking for. For this activity we will use the **Lamp A** icon. Click on the **Lamp A** icon and move your mouse into the **Block Diagram** window. You now have the icon and you can place it on the **Block Diagram** screen just to the right of the **Begin** icon leaving a little white space between the icons. A pink line should appear connecting the **Lamp A** icon and the **Begin** icon. Your program should look similar to the graphic below.

If you do not see the pink wire connecting the two icons you will need to use the **Connect Wire** tool from the **Tools Palette** to connect (by clicking) the top right side of the **Begin** icon to the top left side of the **Lamp A** icon.

Step 7—Next we need to find the **Wait For** sub-palette from the **Functions Palette** so we can program the RCX to wait for two seconds. As you move you mouse over the different icons on the **Function Palette** you will notice the name of each icon you move over displayed at the top of the **Functions Palette** window. The **Wait For** icon looks like the graphic displayed below.
Find the *Wait For* icon and click it. This will change your *Functions Palette* into a *Wait For Palette* and give you more icons to work with. The icon that will give your program a two second delay is the one that looks like the graphic below.

Place the *Wait for 2 sec.* icon in your program just like you did with last icon but place it directly to the right of the *Lamp A* icon leaving a little white space between the icons. Again the pink wire should appear and connect to the icon directly to the *Lamp A* icon, if the wire does not appear you will need to connect it using the *Connect Wire* tool from the *Tools Palette*. Your program should look like the example below.

**Step 8**—Because computers cannot assume what we want we need to program in every step. If you were told to “turn the light off for two seconds” you would probably go turn the switch wait for two seconds and then turn it back on. The computer has to be told every step. So next we need to tell our program to turn off the output after the *Wait for 2 sec.* icon. The icon we will use is the *Stop A* icon, it looks like the graphic below.

Place it in your program to the right of the *Wait for 2 sec.* icon, again leaving a little white space between the icons. Be sure the pink wire attached, if not you will need to connect the icons.

Now use the *Connect Wire* tool from the *Tools Palette* to connect the top left of the *End* icon already on your *Block Diagram* window to the top right of the *Stop A* icon. Your program should look like the graphic below.

Your program should now be complete. If your program has been written correctly you will see a white arrow directly under where the *Edit* menu is on the main menu that extends across the top of the programming environment. If your program has errors there will be a broken grey arrow.
If you have a broken arrow check to be sure your wires are connected correctly and that you have no extra icons on your *Block Diagram* programming area. If you have a white arrow you are ready to send your program to the RCX for testing, but first you need to get the RCX set up to receive and run the program.

**Step 9**—Find the Lego USB tower in your kit. The tower is already connected to the computer using the USB port. Now we can work on attaching the lamp to the output.

Earlier you learned that the output simply sends a designated amount of voltage out a designated port. From our program we see we are using output A. Connect a white Lego lamp to output A as shown below. Be sure it is only connected to output A and not A and B.

![Connect Lamp](image1.png)

**Step 10**—Place the RCX approximately six to twelve inches from the Lego tower with the tower facing the infrared receiver on the RCX as shown below. Now you are ready to send the program. Be sure the RCX is turned on.

![6-12 Inches](image2.png)

In RoboLAB, find the white arrow mentioned before under where the edit menu is on the screen and click it. You may need to click the green check mark on the screen and the RCX will make a sound to say it has received the program. If it did not work try again and be sure the RCX is turned on.

**Step 11**—To test your program simply press the green run button and see if the Lego
lamp you connected to output A turns on for two seconds and then off. If this happens you have completed the program.

**Team Assignment Challenge**

As a team, write a program to control the outputs on the RCX. First, turn output A on for 5 seconds, then output B on for 1 second, then output C on for 5 seconds. Save the program and exit RoboLAB when completed. Remember you will need to connect a motor or lamp to outputs A, B, and C.

**Review Questions**

1. An output sends information to the microcontroller. True or False
2. How many icons can be used to send electricity from an output? ______
3. What does RCX stand for? __________________ _________________

   __________________

4. How many outputs are available on the RCX? ______
5. How can you tell if your RCX is turned on? _________________________
6. What happens if the program you have written has errors?

   ______________________________________________________________

7. Can you send a program with errors to the RCX? ______
8. What happens after the RCX finishes receiving the program?

   ______________________________________________________________

9. How can you tell whether the program on your RCX is running?

   ______________________________________________________________

10. How far away from the tower should your RCX be when sending a program?

    ___to___ Inches
Appendix C

My Class Activities Survey
**My Class Activities**

Marcia Gentry Ph.D. and Robert K. Gable Ed.D.

We would like to know how you feel about your class activities. Read each sentence and indicate how often this happens for you in your class by coloring in the doughnut. There are no right or wrong answers. Your answers will be kept secret. Remember to color in a doughnut for each sentence.

<table>
<thead>
<tr>
<th>Student ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

**Grade**

<table>
<thead>
<tr>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

**I am a...**

- Female
- Male

In the example below, the person indicated that his/her class is often enjoyable.

**Example: My class is enjoyable.**

<table>
<thead>
<tr>
<th>Never</th>
<th>Seldom</th>
<th>Sometimes</th>
<th>Often</th>
<th>Always</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. What I do in my class fits my interests.
2. I have an opportunity to work on things in my class that interest me.
3. What I do in my class gives me interesting and new ideas.
4. I study interesting topics in my class.
5. The teacher involves me in interesting learning activities.
6. What I learn in my class is interesting to me.
7. What I do in my class is interesting.
8. My class has helped me explore my interests.
9. The activities I do in my class are challenging.
10. I have to think to solve problems in my class.

Please continue on the back
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11. I use challenging materials and books in my class.</td>
<td>Never</td>
<td>Seldom</td>
<td>Sometimes</td>
<td>Often</td>
</tr>
<tr>
<td>12. I challenge myself by trying new things.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. My work can make a difference.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14. I find the work in this class demanding.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. I am challenged to do my best in class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16. What we do in class fits my abilities.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17. This class is difficult.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18. I can choose to work in a group.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19. I can choose to work alone.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20. When we work together, I can choose my partners.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21. I can choose my own projects.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22. When there are many jobs, I can choose the ones that suit me.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23. I can choose materials to work with in the class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24. I can choose an audience for my product.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25. I look forward to my class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26. I have fun in my class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27. The teacher makes learning fun.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28. I like what I do in my class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29. I like working in my class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30. The activities I do in my class are enjoyable.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31. I like the projects I work on in my class.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>