

**PHYSICAL COMPUTING EDUCATION:
DESIGNING FOR STUDENT AUTHORSHIP OF VALUES-BASED
LEARNING EXPERIENCES**

A Dissertation
Presented to
The Academic Faculty

by

Kayla DesPortes

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
August 2018

COPYRIGHT © 2018 BY KAYLA DESPORTES

**PHYSICAL COMPUTING EDUCATION:
DESIGNING FOR STUDENT AUTHORSHIP OF VALUES-BASED
LEARNING EXPERIENCES**

Approved by:

Dr. Betsy DiSalvo, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Ben Shapiro
College of Engineering and Applied
Sciences
University of Colorado, Boulder

Dr. Mark Guzdial
School of Interactive Computing
Georgia Institute of Technology

Dr. Paulo Blikstein
Graduate School of Education
Stanford University

Dr. Wendy Newstetter
School of Interactive Computing
Georgia Institute of Technology

Date Approved: July 5, 2018

To my mom and dad, Naomi and David DesPortes who have always believed in me and instilled a mindset to strive to improve myself and the world around me. To my brother and sister in-law, Joshua DesPortes and Ratna Atri, who have always been there for me cheering me on. To the Big Red Love, who are my chosen family, supporting me through the ups and the downs, from near and far. To my partner, Sean McCully, who has grounded me and grown with me, cooked with me and cooked for me, put up with me and threw down for me, entertained me and encouraged me, and most importantly been there for me throughout the whole process.

ACKNOWLEDGEMENTS

Without the help and support of a number of people this dissertation would not be possible. I want to thank the many amazing people within the community organizations I've worked with over the past five years who took the time and effort to work with me and make my investigations possible. The work would not have been possible without the many students who dedicated their time and provided invaluable insights for helping me better understand physical computing education and what it means to create equitable computing environments. All the educators I have worked with deserve special recognition for continuously striving for the best for their students and for opening up pathways for their students to engage with computing. I want to thank all the members of LS&T Alley as well as the HCC community, who provided guidance, feedback and support throughout my time at Georgia Tech. Lauren Margulieux deserves a special thanks for spending several hours patiently going over my studies helping me learn how to conduct more rigorous study design and statistical analyses, all while also helping me recruit participants. Without her I would not have been able to do this work. I'd also like to thank Briana Morrison, Ben Shapiro, Lisa Anthony, and Nathan Holbert who provided support to recruit students at their universities when I was struggling to find participants. Lastly, my dissertation would not be what it is without my amazing committee. Betsy DiSalvo, Mark Guzdial, Ben Shapiro, Wendy Newstetter, and Paulo Blikstein, have lent me their time, expertise and wisdom as they critiqued and improved my work over the years, pushing me to think in novel ways. My adviser, Betsy DiSalvo deserves more thanks than I can give for introducing me to research, providing me with opportunities to explore problem spaces

of interest with the needed the constraints and guidance to succeed, helping me develop perspectives on my work, equity, and the field of computing education, teaching me valuable lessons about research and academic life, and serving as my biggest advocate throughout my journey at Georgia Tech.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
LIST OF TABLES	x
LIST OF FIGURES	xii
LIST OF SYMBOLS AND ABBREVIATIONS	xix
SUMMARY	xx
CHAPTER 1. INTRODUCTION	1
1.1 Why Physical Computing Education?	1
1.2 Justification for Exploring Values in Physical Computing Education	4
1.2.1 Importance of Values for Engagement and Motivation	4
1.2.2 Values-Based Learning—Striving for Equitable Computing Environments	5
1.3 Justification for Exploring Physical Computing Tools	6
1.4 Thesis Statement and Research Questions	6
1.4.1 Thesis Statement	6
1.4.2 Research Questions	7
1.5 Summary of Studies	7
1.5.1 Mapping of Research Questions to Studies	7
1.5.2 Summary of Studies	10
CHAPTER 2. Situating Values-Based Learning	14
2.1 Introduction	14
2.2 Situating the Values-Based Learning	16
2.2.1 Values and Value Domains	16
2.2.2 Integrating Culture into Our Understanding of Values and Action	19
2.3 Differentiating Values-Based Learning	21
2.3.1 Interest in Education	21
2.3.2 Culture in Education	22
2.4 Infrastructure to Deconstruct the Learning Environment	25
2.4.1 Participatory Design as a Method for Understanding Values	26
2.5 Infrastructure in Project-Based Work	27
2.6 Supporting Theoretical Constructs	29
2.6.1 Adolescent Self-Concept	29
2.6.2 Boundary Objects	32
CHAPTER 3. Two Case-Studies of Values-Based Learning	36
3.1 Introduction	36
3.2 The MoveLab Workshop	38
3.2.1 Introduction and Context of Study	38
3.2.2 Data Collection and Analysis	39
3.2.3 Infrastructure in the Learning Environment	41

3.2.4	Findings and Discussion	52
3.3	The Day of the Dead Puppet Summer Camp	67
3.3.1	Introduction and Context of Study	68
3.3.2	Data Collection and Analysis	69
3.3.3	Infrastructure in the Learning Environment	70
3.4	Cross Analysis of Values in MoveLab and Day of the Dead	83
3.4.1	Value Analysis of Infrastructure for Reflection on Values	87
3.4.2	Value Analysis of Infrastructure for Organization	99
3.4.3	Value Analysis of Infrastructure for Building Proficiency and Knowledge	104
3.4.4	Summary of Infrastructure	109
3.5	Conclusion	113
3.6	A Note on Values-Based Learning and its Limitations	116
3.7	Contributions	118
CHAPTER 4. Background: Physical Computing Tools		120
4.1	Introduction	120
4.2	Black-Box and Glass-Box Scaffolding	123
4.3	Previous Analyses of Physical Computing Tools	124
4.4	Analyzing Modularity	126
4.4.1	Segmenting the Hardware	126
4.4.2	Electronic Connectors	128
4.4.3	Programmable Electronics	131
4.4.4	Peripheral Electronics	133
4.5	Spectrum of Modularity	135
4.5.1	Level 1: Plug-and-Play with Static Components	135
4.5.2	Level 2: Plug-and-Play with Static and Variable Components	138
4.5.3	Level 3: Plug-and-Play with Variable Components	141
4.5.4	Level 4&5: Static & Variable Single Component Modules	143
4.5.5	Level 6: Open Circuit Modules	146
4.5.6	Open-Ended Prototyping Tools	147
4.5.7	A Note on the Spectrum of Modularity	151
4.6	Misconceptions	153
4.6.1	Computer Science Misconceptions	153
4.6.2	Electronics Misconceptions	158
4.6.3	Physical Computing Misconceptions	161
CHAPTER 5. BitBlox Comparative Classroom Study		168
5.1	Introduction and Context of Study	168
5.2	Data Collection and Analysis	170
5.3	Findings	172
5.3.1	Circuit Layout	172
5.3.2	Tool Issues	173
5.3.3	Component Errors	174
5.3.4	Terminology	175
5.3.5	Other Circuit Building Issues	175
5.3.6	Collaboration	176
5.4	Discussion	177

5.4.1	Simplicity in Design	177
5.4.2	Affordances for Organization	177
5.4.3	Designing for Appropriation	178
5.4.4	Including Identifiers	178
5.4.5	Size of Tool	179
5.5	Contributions and Limitations	179
CHAPTER 6. Arduino Laboratory Study of Physical Computing Tools		181
6.1	Introduction	181
6.2	Study Design	182
6.3	Study Procedures	186
6.3.1	Study Protocol	186
6.3.2	Protocols for Providing Hints to Participants	192
6.3.3	Protocol for Concurrent Think-Aloud	196
6.3.4	Collecting Video and Audio Data	197
6.4	Study Materials	198
6.4.1	Pre-Study Screening Survey	199
6.4.2	Tests of Self-Efficacy	200
6.4.3	Tests of Knowledge	200
6.4.4	Post-Study Debrief Interviews	201
6.5	Participants	202
6.6	Data Analysis and Findings	204
6.6.1	Tests of Self-Efficacy and Knowledge	205
6.6.2	Completion of Activities Non-Think-Aloud Section	214
6.6.3	Codebook for Qualitative Analysis	216
6.6.4	Overview of the Obstacles, Breakdowns, and Bugs Data	220
6.6.5	Hardware Bugs, Breakdowns and Obstacles	232
6.6.6	Software Bugs, Breakdowns and Obstacles	245
6.6.7	Hardware+Software Bugs, Breakdowns and Obstacles	266
6.6.8	Non-Think-Aloud Section Extra Interview Questions	278
6.6.9	Reflection on The Experience	285
6.7	Discussion	295
6.7.1	Findings Situated with Prior Literature	296
6.7.2	Working Through the Issues – Novices Strategies for Solving Problems	302
6.7.3	Design of Physical Computing Tools	308
6.8	Contributions	318
6.9	Limitations	319
CHAPTER 7. Discussion and Contributions		321
7.1	Introduction	321
7.2	RQ1: Culture, Values, and Physical Computing Learning Environments	322
7.2.1	Summary of Contributions: RQ1	322
7.2.2	Conceptualizing the Design of the Learning Environment	323
7.2.3	Analyzing the Learning Environment from the Perspective of Values	324
7.2.4	Impact of the Learning Environment from the Perspective of Value-Based Learning	326
7.3	RQ2: Novices Experiences with the Arduino	332

7.3.1	Summary of Contributions: RQ2	332
7.3.2	Obstacles, Breakdowns and Bugs of Novices	333
7.3.3	Work Processes of Novices	335
7.3.4	Novice Student Reflections on Their Experience	336
7.4	RQ3: Designing Physical Computing Tools for Novices	338
7.4.1	Summary of Contributions: RQ3	338
7.4.2	Usability of Physical Computing Tools	339
7.4.3	Abstraction Layers Impact on Learning and Self-Efficacy	342
7.5	Conclusion	346
CHAPTER 8. Future Work		349
8.1	Overview	349
8.2	Further Explorations of Infrastructure for Values-Based Learning	349
8.3	Further Exploration of Tools and Learning for Values-Based Learning Experiences	351
8.3.1	Re-Design of Physical Computing Tool Lab Study	351
8.3.2	Moving Forward with the Tool Investigations	352
APPENDIX A. Laboratory Study Materials		356
A.1	Pre-Screening Survey	356
A.2	Pre-/Post-Test of Self-Efficacy and Knowledge	357
A.3	Demographics Survey	364
REFERENCES		365

LIST OF TABLES

Table 1	– Overview of Research Questions and Studies	8
Table 2	– Definitions of Value Domains (replication of table from (Bardi & Schwartz, 2003))	18
Table 3	– Codebook for Understanding Self-Concept	40
Table 4	– Perceived Comfort and Aptitude on the Computer	53
Table 5	– Pre- and Post-Study Survey Data Focused on Participant Interest and Identification	55
Table 6	– High-Level Codes for Values	70
Table 7	– Schwartz et al.’s Value Domains replicated from (Bardi & Schwartz, 2003)	84
Table 8	– Infrastructure in the Learning Environments	86
Table 9	– Dance Themes and Values Coded	87
Table 10	– Puppets and Values Coded	88
Table 11	– Spectrum of Modularity for Peripheral Electronic Components	134
Table 12	– Student demographic information	168
Table 13	– Student prior experience	168
Table 14	– Codebook with descriptions and examples	171
Table 15	– Tools Used in Each Group	183
Table 16	– Arduino Laboratory Study Protocol	188
Table 17	– Study Tasks	191
Table 18	– Participants by College in Each Group for Think-Aloud	203
Table 19	– Participants by College in Each Group for Non-Think-Aloud	203
Table 20	– Demographic Makeup of Groups	204

Table 21	– Descriptive Statistics of Self-Efficacy (SE) and Knowledge (KN) (N=31)	208
Table 22	– Outlier Calculations	
Table 23	– Descriptive Statistics of Self-Efficacy (SE) and Knowledge (KN) (N=30)	
Table 24	– Correlation Tests Between Dependent Variables	
Table 25	– Tests of Between-Subject Effects	
Table 26	– Multivariate Tests	
Table 27	– Tests of Between-Subjects Effects for Post-Test Self-Efficacy	
Table 28	– Tests of Between-Subjects Effects for Post-Test Knowledge	
Table 29	– Task Completion Rates by Group	
Table 30	– Hints Requested by Group	
Table 31	– Original Obstacles/Breakdowns/Bugs coding scheme from Booth et al.’s study of the Arduino in a laboratory setting (Booth et al., 2016)	
Table 32	– Codes for Obstacles: <i>Hurdles that the participant had to overcome</i>	
Table 33	– Codes for Breakdowns: <i>Errors in participant’s thoughts and/or actions</i>	
Table 34	– Codes for Bugs: <i>Faults introduced and fixed</i>	
Table 35	– Bugs Identified by Participants in BitBlox Debug Circuit Question	
Table 36	– Summary of Breakdowns from Arduino Laboratory Study	
Table 37	– Spectrum of Modularity	

LIST OF FIGURES

Figure 1	– Structure of academic self-concept (image remade from Shavelson et al. (Barbara M Byrne & Shavelson, n.d.; Shavelson et al., 1976))	31
Figure 2	– Issue Diagram of <i>Fake Friends</i> completed by one of the participants in the MoveLab. The participant identified this issue as <i>backstabbing, liars, 2-faced, strangers, and shame</i> . This made them feel: <i>disappointed, mad, and irritated to a certain point</i> and then they felt: <i>oh well, their loss, over it</i> . They wanted to express this with representations of: <i>don't talk to me, I'm done, happy I lost a fake friend</i> . Their final stage was a celebration: <i>yay, no fakes!</i>	46
Figure 3	– Dance Planning Worksheet that guided students to organize components. This document was also used to break students into groups with similar themes.	48
Figure 4	– Dance Grid protocol for organizing ideas and planning the dances: (top) template given, and (bottom) example of group use.	50
Figure 5	– Organization Grid protocol to facilitate outline of work: (top) template given, and (bottom) example of group use.	51
Figure 6	– One of the costume tops for the bullying-themed dance, which included: a) two capacitive touch sensors, b) a piezo buzzer, and c) an RGB LED strip	52
Figure 7	– Sketch of a puppet using START, INPUT, OUTPUT and FINAL indicating the computational communication with the microcontroller in the center.	74
Figure 8	– Prompts for Critique Exercise	76
Figure 9	– Puppet of Dog with light sensitive resistor that would trigger NeoPixel strip along the mouth to light up signifying happiness	77
Figure 10	– Puppet of Father that had a NeoPixel strip triggered by a button that would light up red and blue to represent the participant's father's grill	78
Figure 11	– Puppet of Great Aunt in heaven with motorized wheelchair. Input, located at the joystick of the wheelchair, activates motors in the wheels to move the chair back and forth and turns on the motor in the base of the necklace so that it spins.	79

Figure 12	– Puppet of Virus had a button inside the foam body that when pushed with the needle would turn off the lights in the virus (ie LED balls) demonstrating killing the virus.	80
Figure 13	– Puppet of Tiger had a flex sensor attached to a fake tablet that would trigger the rage of the tiger by flashing the LED eyes green and orange and moving the mouth back and forth with a motor when bent signifying the breaking of the tablet	81
Figure 14	– Puppet of Emoji that gave you a pillow to rest on while you charged your phone. It indicates when your phone is charged by buzzing piezo buzzer and moving the mouth of the pillow with a motor. The input was actually a button.	82
Figure 15	– Issue Diagram from a participant who chose the issue of: <i>abandoned homeless people, children, and animals</i> . They indicated this makes them feel: <i>empty, helpless, frustrated</i> . They wanted to express this using <i>Hip Hop</i> and <i>Wild dancing</i> . Their final stage represents a cycle: <i>It's just gonna keep being a cycle until I'm old enough to do something about it</i> .	91
Figure 16	– Planning worksheet from student who wanted to represent the destruction of the environment in her dance. She wanted to represent: <i>The beauty & magic of nature being destroyed by greedy humans whose only cares are money and industry</i> . The movements she wanted to use were: <i>slow, pretty, small movements for nature</i> and <i>fast, morbid, aggressive, big movements for destruction</i> . In the dance she wanted to use: <i>an accelerometer to project images of beautiful nature and then the destruction of it. A light sensor after the nature part it goes dark and morbid music comes on for the destruction</i> .	93
Figure 17	– Segmentation of hardware for design and analysis	127
Figure 18	– Example plug-and-play circuits that provide users with simple input and output pins to gather or provide data: (a) LilyPad Arduino light sensor (“LilyPad Light Sensor,” 2017) and (b) Microphone breakout board (“High Sensitivity Sound Detector,” 2017)	136
Figure 19	– LCD Breakout board that enables the user to control the brightness of the LCD using a potentiometer (“16x2 LCD Keypad Shield for Arduino Version B,” 2017).	139
Figure 20	– Two level 2 modules: (a) two lead sensor circuit with pull-up/pull-down resistor in which the sensor can be changed; (b) 555 timer (astable) break out enabling user to manipulate the resistances and capacitance to change signal frequency	140

Figure 21	– Velleman Electronic Lab Kit 130 in 1 (Velleman, 2018)	142
Figure 22	– Snap Circuits (“Snap Circuits,” 2017) are an example of a Level 4 Single Component Static Circuit Board	144
Figure 23	– Thames & Kosmos Science Experiment Kit Electricity Master Lab (Thames & Kosmos, 2018)	145
Figure 24	– Vat19’s Draw Circuits Kit (“Draw Circuits Kit,” 2018)	145
Figure 25	– Level 6 Modules: (a) parallel circuit, and (b) combination parallel/series circuit	147
Figure 26	– Showing the same circuit prototyped with (a) the standard breadboard, and (b) the BitBlox modules. (c) Shows the circuit diagram that represents these circuits– Showing the same circuit prototyped with (a) the standard breadboard, and (b) the BitBlox modules. (c) Shows the circuit diagram that represents these circuits	148
Figure 27	– Most common standard breadboard design with two sections: every hole in each row is interconnected, and every hole within each column is interconnected (a) shows the front or top, with (b) showing the back or bottom of the breadboard	149
Figure 28	– Another common breadboard design in which there are only the columns with interconnected holes.	150
Figure 29	– BitBlox modules (a) and (b) completely connected modules, (c) five separate columns of connected holes, (d) two separate sections of connected holes, (e) seventeen separate columns of connected holes, and (f) front and back of two sections of 10 separate columns of connected holes.	151
Figure 30	– Blinky LED code that is used to continuously turn on and off an LED connected to pin 13.	155
Figure 31	– (a) The schematic for the introductory LED circuit. (b) The schematic for the Simon memory game circuit.	169
Figure 32	– An example Breadboard circuit	172
Figure 33	– (a) Group F’s spread out circuit, (b) Group D’s large organized circuit, and (c) Group A’s small compact circuit– Arduino Open Source Prototyping Platform	173
Figure 34	– Arduino Open Source Prototyping Platform	182

Figure 35	– BitBlox modules with connection scheme conveyed through coloring: (a) and (b) have all the holes connected to one another, (c) has five separate columns of connected holes, and (d) has two separate sections of connected holes	184
Figure 36	– The standard breadboard has two types of connection schemes: every hole in each row is interconnected and every hole within each column is interconnected—(a) shows the top of the breadboard with rectangles around select rows and columns of connected holes and, (b) the bottom of the breadboard illustrating metal connections in the rows and columns	184
Figure 37	– Screenshot of the Modkit Blocks-Based Integrated Development Environment	186
Figure 38	– Blinky LED Arduino circuit (left) and Modkit code (right)	189
Figure 39	-- Blinky LED Circuits for (a) Group 1: Plug-and-Play Module with Static Components, (b) Group 2: BitBlox, and (c) Group 3: Breadboard	190
Figure 40	– Plug-and-Play Circuit Module with Variable Components used by Group 1 for Task #1. The pin headers provide positions for the LED, resistor, power and ground to be connected.	191
Figure 41	– Protocol for hints in Think-Aloud Section	193
Figure 42	– Example of Screen Recording with Webcam Capture	198
Figure 43	– Average Self-Efficacy scores of Think-Aloud participants by group out of 145 points	206
Figure 44	– Average Knowledge test scores of Think-Aloud participants by group in a 15-point test	206
Figure 45	– Average self-efficacy scores of Non-Think-Aloud participants by group out of 147 points	210
Figure 46	– Average knowledge test scores of Non-Think-Aloud participants by group in a 15-point test	210
Figure 47	– The circuit and code created by participants in the Blink LED activity	222
Figure 48	– Task #1 example circuit and code	223
Figure 49	– Task #2 example circuit and code	224

Figure 50 – Obstacles in Think-Aloud and Non-Think-Aloud Section by Activity	225
Figure 51 – Obstacles in Think-Aloud and Non-Think-Aloud Section by Location and Group	226
Figure 52 – Bugs in Think-Aloud and Non-Think-Aloud Section by Activity	227
Figure 53 – Bugs in Think-Aloud Section and Non-Think-Aloud by Location and Group	228
Figure 54 – Breakdowns in Think-Aloud and Non-Think-Aloud Section by Activity	230
Figure 55 – Breakdowns in Think-Aloud and Non-Think-Aloud Section by Group	231
Figure 56 – Hardware Bugs, Breakdowns, and Obstacles by Group and Activity	232
Figure 57 – Hardware Breakdowns from Think-Aloud and Non-Think-Aloud Section	234
Figure 58 – Arduino without labeling on the pin headers (left) and with labeling (right)	236
Figure 59 – Example of open circuit created by participants who did not connect components into their circuits properly	237
Figure 60 – Task #1 correct circuit formation (left) and Task #1 common incorrect circuit formation (right)	240
Figure 61 – Task #2 Correct Circuit Formation	240
Figure 62 – Task #2 Common Incorrect Circuit Formations	241
Figure 63 – Software Bugs, Breakdowns, and Obstacles by Group and Activity	246
Figure 64 – Software Breakdowns from Think-Aloud and Non-Think-Aloud Section	248
Figure 65 – Incorrect code initializing two forever loops	249
Figure 66 – Incorrect code which connects code after a forever loop	250
Figure 67 – Sequentiality Example Step 1	251
Figure 68 – Sequentiality Example Step 2	252

Figure 69	– Sequentiality Example Step 3	253
Figure 70	– Sequentiality Example Step 4	254
Figure 71	– Sequentiality Example Step 5	255
Figure 72	– Participant who had a misconception about the delay block thinking that it caused the LEDs to blink	258
Figure 73	– Example of selecting the wrong Arduino pin from the drop-down menu	260
Figure 74	– Example Pin Signal Breakdown Step 1	261
Figure 75	– Example Pin Signal Breakdown Step 2	262
Figure 76	– Participant attempting to connect blocks (a) unsuccessfully and (b) successfully	263
Figure 77	– Two pieces of code that demonstrate a connection between the pinMode block and the forever block (left) and code that demonstrates unconnected blocks between the pinMode and the forever block (right)	264
Figure 78	– (Left) Code demonstrating the shape differences between the <i>while</i> , <i>break</i> and <i>and</i> code blocks leading participants to integrate or rule out their use. (Right) Similarities between digitalWrite and analogWrite blocks.	266
Figure 79	– Hardware+Software Bugs, Breakdowns, and Obstacles	267
Figure 80	– Hardware+Software Breakdowns from Think-Aloud and Non-Think-Aloud Section	268
Figure 81	– Example Incorrect Assessment of Bug Location Step 1	270
Figure 82	– Example Incorrect Assessment of Bug Location Step 2	272
Figure 83	– Example Incorrect Assessment of Bug Location Step 3	273
Figure 84	– Code from participant who could not tell if her code was blinking one of the LEDs once or twice because it was changing so rapidly	275
Figure 85	– Error participants would receive if they did not plug in their Arduino when programming it	277
Figure 86	– Code in Circuit Debug Question in post-study debrief with non-think-aloud section	282

Figure 87	– BitBlox Arduino Circuit in Circuit Debug Question in post-study debrief with non-think-aloud section	283
Figure 88	– Micro:Bit example of paired down microcontroller (left) and Adafruit Trinket Mini example of a less complex microcontroller (right)	310

LIST OF SYMBOLS AND ABBREVIATIONS

CS Computer Science

HW Hardware

IDE Integrated Development Environment

PD Participatory Design

SW Software

SUMMARY

No one should have to conform or assimilate in order to participate in the design and creation of technology; however, this is the reality facing many students today. Prior research suggests that the issues surrounding lack of participation in computing and engineering education among underrepresented minorities goes beyond simply access or obtaining “critical mass” (Blikstein, 2008; J. Margolis, 2008; J. Margolis & Fisher, 2002). Instead, it points to an exclusive, dominant culture that is often toxic for students outside the stereotypical mold. From being obligated to participate in activities and projects that do not speak to their values, to unspoken and unknown requirements of prior experience; non-dominant students are marginalized in ways that span the socio-technical learning environment.

Creating computing environments that are inclusive, equitable, and supportive for students coming in with different backgrounds, cultures, and values is a difficult and complex problem. Research has shown that the design of these learning environments has several characteristics to attend to in order to be successful. One needs to be cognizant of the values embedded in the content delivered (Buechley & Eisenberg, 2008; Scott & Hood, 2009), the types of tools used (Blikstein, 2015; Deitrick et al., 2015a), the climate fostered within the social interactions (J. Margolis, 2008; J. Margolis & Fisher, 2002), and how the learning environment fits within the socio-cultural context of the students’ lives (Betsy DiSalvo, Guzdial, Bruckman, & McKlin, 2014; K. A. Searle & Kafai, 2015). While researchers have demonstrated the ability to create inclusive learning environments for a targeted set of underrepresented students, we still do not understand how to extrapolate

from these targeted designs meant to fit specific needs, to methods that demonstrate success across various students. The ability to extrapolate is important for being able to more dynamically integrate diverse student values in ways that support differences between students.

The work presented in this dissertation explores learners' values, or their "*conceptions of what is ultimately good, proper, or desirable in human life*"(Graeber, 2001) as a lens to create more dynamic learning environments for building personally meaningful experiences with computing. I posit that by deconstructing and analyzing characteristics within the learning environment for their ability to support students' values, creates an inclusive and equitable learning experience. I explore these concepts within two projects with middle school and high schools students: the MoveLab Workshop and the Day of the Dead Puppet Summer Camp. In the MoveLab Workshop, participants were tasked with creating dance performances that integrated microcontroller technology in order to represent issues that they cared about. In the Day of the Dead Puppet Summer Camp, participants were tasked with creating interactive puppets that were representative of someone or something they had lost in their lives. By having a similar design within both learning environments, we were able to analyze the various values that manifested within these two environments and how they tied to the design decisions in the learning environments.

We found success in the participatory design methods used to guide the participants in reflecting on their values and integrating them into the designs of their computational artifacts. Furthermore, the parts of the learning environments that had low stakes for participation and stimulated discussion around participants' values led to productive

exchanges in ideas and sharing about the students' and leaders' cultures. We found issues when certain design decisions integrated materials or activities that conflicted with the participants' values. Specifically, conflicts around *achievement* and *power* were the most destructive to their learning experience. The tools for physical computing were found to contribute to this, impacting the participants' proficiency with computing and desire to want to engage with it again.

The importance of creating tools that provide access to the hardware and software concepts, lies in how they prepare students with the knowledge needed to integrate computing in ways they value within their lives. If we continue to black-box these concepts for the sake of usability we will “obscure the technology [we] seek to make accessible” (Mellis, Jacoby, Buechley, Perner-Wilson, & Qi, 2013). The final two studies presented within this dissertation examined how we might bring accessibility to the concepts integrated into physical computing environments by examining the design of the hardware and software tools. The first study examined two prototyping tools for electronics: the Breadboard and BitBlox. In a comparative classroom study, novice high school students were observed working and learning with the two different prototyping tools. Through this investigation we were provided with insights into the design decisions within these tools that had implications for how the students were working and collaborating with one another.

A within and between-subjects laboratory study examining different prototyping tools was then conducted in order to more closely understand the work processes, the errors, and the thought processes of novices as they worked with these tools for the first time. Conducting pre-/post-study tests of knowledge and self-efficacy, along with

examining video and audio footage of 46 participants provided a comprehensive understanding of the experiences of novices working with the Arduino. While we did not see statistically significant results in our test data across the groups, the qualitative analysis shed light on how participants learned about physical computing, how they used their prototyping tools, how they approached different problems, and the prevalence of different types of obstacles, breakdowns and bugs. The data revealed the benefits and drawbacks of the different prototyping tools and how the design decisions within the tools contributed to this.

The contributions from this dissertation provide a better understanding of how we should design learning environments and tools for physical computing education in order to support inclusivity. The work provides a way to conceptualize and frame the design and analysis of the learning environment from the perspective of values. Furthermore, the analysis of the learning environments in these studies provided an understanding of what aspects of the learning environment supported and hindered students in shaping the learning environment around their values. The studies captured real-world and laboratory experiences of novice learners working with physical computing tools. The investigations offered insights into the types of obstacles, breakdowns and barriers they are faced with, along with an understanding of how the tools contributed to their experience. By continued exploration of the learning activities and tools to support students' values we will build our understanding for how to support a diversity of students to learn about and use computing in ways that are personally meaningful.

CHAPTER 1. INTRODUCTION

Lack of diversity in computing is a persistent problem, with low numbers of Women, Blacks, Hispanics, American Indians, and Alaska Natives obtaining degrees and working in engineering and computer science fields (NSF, 2017). Broadening participation in computing is important because research has demonstrated that *who* is enabled to participate in the design of technology has implications for how problems are solved and the types of problems technology solves (ex. (Bijker, 1997; Noble, 2018; L. Winner, 1986; Langdon Winner, 1993)). With the growing integration of technology in how we work, play and live, it is essential that we begin to bring equity to who is empowered to design and build technology. While there are a number of complex and multifaceted issues within this problem space, my research investigates a subset of the issues within the context of physical computing education focusing on the design of learning activities and educational technology.

1.1 Why Physical Computing Education?

Physical computing is the domain that lies on the intersection of engineering and computer science. It combines hardware and software aspects of technology to create artifacts that have a physical presence and are programmed to interact with the surrounding world. For example, one could create a gyroscope-controlled RC helicopter that flies based on the user tilting her hand, or an augmented dance floor that responds to dancers' location and movement. The combination of hardware and software provides a powerful tool, enabling more complex designs than could be created with either discipline alone, while providing students with various pathways to get involved with the development of

technology. Students can engage in the programming, the electronics, or the design of the physical embodiment of technology. Furthermore, the physical nature enables the integration of computing into numerous domains while creating an environment for experiential learning (ex. (Buechley & Eisenberg, 2008; Deitrick et al., 2015b; K. A. Searle, Fields, Lui, & Kafai, 2014)). This type of hands-on, exploratory education has been advocated for since the early 1900s (Dewey, 1938). Thus, physical computing activities are uniquely situated as a promising context to study how novices learn about computing.

In addition to the benefits of physical computing as a domain, the landscape of tools and opportunities is growing within our society necessitating a greater understanding. The rise in more accessible platforms, such as the Arduino family, Teensy, BlocklyTalky, Raspberry Pi, the GoGo Board and Micro:Bit, are enabling more opportunities to interact with physical computing. The availability of these tools has served as an impetus to use them as a medium to spread computing education in a novel and engaging way. The BBC demonstrated this when they gave millions of Micro:bits to all students in the UK who were starting secondary school (2015). The hardware trends have been fueled by the popularization of the *Maker Movement* which encapsulates a DIY culture, emphasizing individual *production over consumption* (Dougherty, 2013; Hatch, 2014). The increasing popularity has pushed physical computing and making into educational spaces such as public and private libraries (ex. (Kalish, 2011)), and FabLabs and makerspaces inside of schools (ex. (Blikstein, 2013a; Sheridan et al.; Westervelt, 2016)). The accessibility and prevalence of physical computing are expanding within this movement, emphasizing the need to understand how students are working and learning within this domain.

Alongside the rise of the tools, spaces and opportunities, researchers have begun to explore the potential for physical computing to create inclusive computer science and engineering learning environments. The literature demonstrates the breadth of interdisciplinary experiences that can be created by integrating physical computing into disciplines such as interior design (Camarata, Gross, & Do, 2003), dance (DesPortes, Spells, & DiSalvo, 2016b; Latulipe & Huskey, 2008), robotics (Goldman, Eguchi, & Sklar, 2004), knitting (Rosner & Ryokai, 2008), interactive art (Betsy DiSalvo & DesPortes, 2017), and storytelling (Tanenbaum, Tanenbaum, & Antle, 2010). Investigations of the LilyPad Arduino (a development board for wearable computing) (Buechley & Eisenberg, 2008) have demonstrated the importance of interdisciplinary learning environments for broadening the types of students who become interested in computing and broadening how students who learn in these environments view computing (Buechley, Eisenberg, Catchen, & Crockett, 2008; Kafai et al., 2014; K. A. Searle et al., 2014). The physical computing tools not only have the power to enable new interdisciplinary opportunities, but they can also “be a powerful agent of emancipation” (Blikstein, 2008, p. 2) and improve the capacity of the learning environment to value different types of knowledge (Deitrick et al., 2015b).

In conclusion, the physical computing domain provides a powerful toolset for the design of technology, it is gaining traction as a way to engage more students, and there is an optimistic outlook for its capability to engage a diversity of students. However, when compared to other fields like math and physics, physical computing is still in its infancy as an educational field. While we can use research from both Engineering Education and Computer Science Education to inform the work in Physical Computing, it is important to not equate the educational environment to just being an additive result of these two

domains. There are many things we do not yet understand about designing physical computing learning environments. My dissertation adds to our knowledge of physical computing education in two main ways. First, the work contributes to the discussion of broadening participation in computing by exploring the design and analysis of learning environments through a lens focused on students' values. Second, the work provides a closer look at the learning experiences of students and how they are impacted by the tools they are using.

1.2 Justification for Exploring Values in Physical Computing Education

1.2.1 Importance of Values for Engagement and Motivation

Since the late 1800s, researchers have sought to design engaging and motivating learning environments that are personally meaningful for all students (William, 1899). However, designing learning environments that have lasting engagement is difficult. Literature provides us with a number of theories that researchers have explored to address this difficulty. For example, studies investigating ways to create learning environments that are: interest-driven (Edelson & Joseph, 2004a), intrinsically motivating (Malone, 1981), authentic (Shaffer & Resnick, 1999), student-driven (Rahimi, van den Berg, & Veen, 2015), and culturally relevant (Gloria Ladson-Billings, 1995b), seek to help us tap into student motivation in an lasting way. While all of these theories and approaches have demonstrated some success, we still do not have a full understanding of why they succeed when they do, or perhaps more importantly, why they breakdown when they do. I argue that an explicit consideration of values in the design and analysis of learning environments provides additional insight into these important questions. I define values using a definition

from Graeber, as, “*conceptions of what is ultimately good, proper, or desirable in human life*”(Graeber, 2001). Attending to values offers explanatory power across students and incorporates an understanding of the socio-cultural influences both inside and outside of a learning environment. Furthermore, by understanding these values in relation to one’s culture—which is made up of *beliefs, stories, gossip, art forms, and ritual practices* that we gather from the communities that we engage with (A Swidler, 1986)—we can better contextualize the values and students’ participation in the learning environment (explained further in section 2.2).

1.2.2 Values-Based Learning—Striving for Equitable Computing Environments

I present values-based learning as both a lens for the design of educational environments and as an analytical construct to understand how a learning environment supports and inhibits students in creating personally meaningful experiences. The construct understands students’ values as situated in their self-beliefs, their communities, society, and the values within the learning environment. As a design approach, it seeks to engage students in creating a personally meaningful experience by supporting them to leverage their values and culture within the learning environment. As an analytical construct, it enables a multi-faceted understanding of why and how students engaged and disengaged in the learning activities. By attending to socio-cultural characteristics, the construct seeks to understand differences between students’ engagement and motivation, in ways that I argue are essential to creating inclusive and equitable learning environments. I will further situate these ideas within the background literature in CHAPTER 2 and explore them within the research presented in CHAPTER 3.

1.3 Justification for Exploring Physical Computing Tools

While creating personally meaningful experiences with computing is essential to broadening participation, it could prove meaningless if students are not also enabled to gain proficiency and self-efficacy within these experiences. Underrepresented minorities have historically been pushed out of computing environments and typically do not have the same support structure that the dominant majority students possess in these domains (J. Margolis, 2008; J. Margolis & Fisher, 2002). This reality emphasizes the need to target design of these inclusive learning environments for novices. Focusing on how the tools for physical computing affect students' experiences provides insight on how to scaffold student learning in ways that builds their knowledge and their self-confidence with physical computing. By doing this, we not only can support students in the environment to engage with computing in ways that integrate their values but can provide an avenue for further exploration of the domains once they leave the learning environment.

1.4 Thesis Statement and Research Questions

1.4.1 Thesis Statement

From the hardware to the software, physical computing requires a diversity of support for students as they use a variety of tools when designing and building a computational artifact. To understand their experiences in the context of the learning environment I present the following thesis statement:

I theorize that exploring the design and analysis of the learning environment through a lens focused on values can provide us with an understanding of how to

empower students to create personally meaningful experiences with computing. Further, to enable these types of experiences, physical computing tools should be designed to address usability, learning and self-efficacy.

1.4.2 Research Questions

The thesis statement is broken down into the following three research questions:

RQ1. *How does design of physical computing learning environments affect students' ability to integrate their diverse values into their learning experience?*

- 1.1 How can we conceptualize the design of the learning environment from the perspective of students' diverse values?
- 1.2 How can we analyze the learning environment from the perspective of students' diverse values?
- 1.3 What characteristics of the learning environment in physical computing activities impact students' ability to integrate their diverse values into their learning experience?

RQ2. *What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?
- 2.4 What are novice students' reflections on working with the Arduino?

RQ3. *How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability?
- 3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?

1.5 Summary of Studies

1.5.1 Mapping of Research Questions to Studies

The work presented in this dissertation is composed of four research studies. I will layout the studies in relation to how they help shed light on the research questions in the table below. I then give a brief summary of each of the studies in the following section.

Table 1 – Overview of Research Questions and Studies

Research Question	Data Source	Analysis Methods
<p>RQ1.1 How can we conceptualize the design of the learning environment from the perspective of students’ diverse values?</p> <p>RQ1.2 How can we analyze the learning environment from the perspective of students’ diverse values?</p> <p>RQ1.3 What characteristics of the learning environment in physical computing activities impact students’ ability to integrate their diverse values into their learning experience?</p>	<p>MoveLab Workshop Study:</p> <ul style="list-style-type: none"> ● Demographics ● Prior experience ● Pre-/Post-Perspectives of Computing and Dance ● Post-Study Leader and Student Interviews ● Observations ● Daily Leader Debriefs ● Computational Artifacts and Dances <p>Day of The Dead Summer Camp Study:</p> <ul style="list-style-type: none"> ● Demographics ● Prior experience ● Observations ● Post-Study Student Focus Groups ● Post-Study Leader Interviews ● Computational Artifacts 	<p>Demographic information and prior experience used to contextualize qualitative data</p> <p>Pre-/Post-Perspectives, Knowledge Tests and Self-efficacy Tests was used as a supporting data source to compare and triangulate with qualitative data</p> <p>Qualitative data was pattern coded for emergent themes</p>

Table 1 – Continued

<p>RQ2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?</p> <p>RQ2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?</p> <p>RQ2.3 What are the work processes of novices working with the Arduino?</p> <p>RQ2.4 What are novice students’ reflections on working with the Arduino?</p>	<p>MoveLab Workshop Study:</p> <ul style="list-style-type: none"> ● Observations <p>Day of The Dead Summer Camp Study:</p> <ul style="list-style-type: none"> ● Observations ● Post-Study Student Focus Groups <p>BitBlox/Breadboard Comparative Classroom Study:</p> <ul style="list-style-type: none"> ● Demographics ● Prior experience ● Observations <p>Arduino Laboratory Study:</p> <ul style="list-style-type: none"> ● Demographics ● Prior experience ● Pre-/Post-Knowledge Tests ● Pre-/Post-Self-Efficacy Tests ● Audio/Video Data ● Concurrent think-aloud ● Post-Study Interviews 	<p>Demographic information and prior experience used to contextualize data</p> <p>Pre-/Post-knowledge tests and self-efficacy tests were analyzed for statistical differences between groups</p> <p>Video/Audio were coded for obstacles, breakdowns and bugs</p> <p>Observations, post-study interviews, and focus groups were coded for emergent themes</p>
<p>RQ3.1 What characteristics of the hardware and software tools impact their usability?</p> <p>RQ3.2 How do different abstraction layers in the design of physical computing tools impact students’ learning and self-efficacy?</p>	<p>MoveLab Workshop Study:</p> <ul style="list-style-type: none"> ● Observations <p>Day of The Dead Summer Camp Study:</p> <ul style="list-style-type: none"> ● Observations ● Post-Study Student Focus Groups <p>Arduino Laboratory Study:</p> <ul style="list-style-type: none"> ● Demographics ● Prior experience ● Pre-/Post-Knowledge Tests ● Pre-/Post-Self-Efficacy Tests ● Audio/Video Data ● Concurrent think-aloud 	<p>Demographic information and prior experience used to contextualize data</p> <p>Pre-/Post-knowledge tests and self-efficacy tests were analyzed for statistical differences between groups</p> <p>Video/Audio were coded for obstacles, breakdowns and bugs</p> <p>Observations, post-study interviews, and focus groups were coded for emergent themes</p>

1.5.2 *Summary of Studies*

1.5.2.1 The MoveLab Workshop

The MoveLab was an investigation of a physical computing learning experience combining dance and microcontrollers. This was the first of two studies exploring a lens focused on values for design and analysis of computing learning environments. The workshop lasted 5 days as the students learned about dance and the Arduino at a local performing arts venue in Atlanta. The participants consisted of 8 dance and technology leaders working with 13 middle and high schoolers to create dance performance integrating the Arduino. The students were split into groups to create dances that expressed issues that they have encountered in their lives that they wanted to represent using the dance techniques and technology they learned about. The students ended up creating dances about social and environmental issues. Within the workshop we scaffolded the students using participatory design techniques in order to guide the students in reflecting on their values and integrating them into the learning experience. We examined the learning environment through a lens focused on values to understand how we supported and hindered participants in creating personally meaningful experiences with computing. The investigation of the MoveLab contributed to answering RQ1:

***RQ1.** How does design of physical computing learning environments affect students' ability to integrate their diverse values into their learning experience?*

- 1.1 How can we conceptualize the design of the learning environment from the perspective of students' diverse values?
- 1.2 How can we analyze the learning environment from the perspective of students' diverse values?
- 1.3 What characteristics of the learning environment in physical computing activities impact students' ability to integrate their diverse values into their learning experience?

1.5.2.2 The Day of the Dead Puppet Summer Camp

The Day of the Dead or Día de los Muertos Puppet summer camp was the second investigation into the design and analysis of computing learning environments from a lens focused on values. The study consisted of a two-week summer camp with 7 high school students building interactive puppets using microcontrollers. The students were introduced to the Mexican holiday of Day of the Dead and asked to reflect on someone or something they have lost. The students were scaffolded to create a representation of this person or object using the Arduino microcontroller. Similar to the MoveLab we integrated participatory design methods to promote reflection on values. While the learning environments were structured similarly, the differences enabled us to have two case-studies to compare the success and failures of the learning environments. Through this study we expanded our understanding of research questions RQ1:

***RQ1.** How does design of physical computing learning environments affect students' ability to integrate their diverse values into their learning experience?*

- 1.1 How can we conceptualize the design of the learning environment from the perspective of students' diverse values?
- 1.2 How can we analyze the learning environment from the perspective of students' diverse values?
- 1.3 What characteristics of the learning environment in physical computing activities impact students' ability to integrate their diverse values into their learning experience?

1.5.2.3 BitBlox Comparative Classroom Study

The students in the MoveLab and in the Day of the Dead studies struggled with the Arduino microcontroller and electronic components as they worked on their projects. One of the main difficulties and barriers students faced when working on their projects was in using the breadboard (a tool for quickly prototyping circuits). The design hides connection

information that could be useful for novices. I designed another version of the breadboard called BitBlox, which intends to bring visibility to these connections and provide a modular tool that gives the students flexibility in its use. The BitBlox Comparative Classroom Study was a pilot investigation into the use of this tool in a classroom environment comparing two 11th grade classes of 22 novice students. One class was given the original Breadboard tool and another class was given the BitBlox tool. We taught and observed the students as they worked on two introductory physical computing activities using the Arduino. We focused our observations on the students' mistakes, misunderstandings, questions, collaboration patterns, moments when a feature in the tool seemed helpful or problematic, and patterns in tool use. Within this study we began to start answering research questions: RQ2.1, RQ2.2, RQ2.3, and RQ3:

RQ2. *What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?

RQ3. *How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability?
- 3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?

1.5.2.4 Arduino Laboratory Study of Physical Computing Tools

While the BitBlox Comparative Classroom study began to provide an idea about what the students were thinking and what barriers they faced as they worked on introductory Arduino activities, the real-world nature makes it difficult to understand

everything that occurred with the individuals. To gain more insight, I conducted laboratory study in which novice students went through an introduction to the Arduino and common introductory activities using different tools. The participants were given the same learning materials for the Arduino and the same activities to complete, but the tools between the groups will differ. The study was a within and between-subjects study with a group using circuit boards and then BitBlox, a group using BitBlox and then the Breadboard, and a third group that used the Breadboard and then BitBlox. Within the study 15 participants conducted a think-aloud as they traversed the material while 31 participants were just video and audio recorded. Having the students think-aloud gave us insight into their thought processes as they completed the activities, while the non-think-aloud section provided a large data set of the types of obstacles, breakdowns and bugs the participants encountered. Furthermore, a post-study interview with the participants led to greater insight on their interpretation of the experience and the tools they were using. The study offered insight into: RQ2 and RQ3:

***RQ2.** What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?
- 2.4 What are novice students' reflections on working with the Arduino?

***RQ3.** How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability?
- 3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?

CHAPTER 2. SITUATING VALUES-BASED LEARNING

2.1 Introduction

Developing equitable and inclusive learning environments that not only initially engage, but also sustain learners' attention within a particular conceptual domain is challenging. In this dissertation, I explore this problem space within the context of computing education, which constantly struggles with cultures of exclusivity. My research defines and advocates for the use of *values-based learning*, which brings values and culture to the forefront of design and analysis of learning environments. I define values using a definition from Graeber, “*conceptions of what is ultimately good, proper, or desirable in human life*”(Graeber, 2001). Values are framed as a reflection of how one views themselves and who they want to be, as well as a reflection of the cultures and communities in which one participates.

Culture is brought into this discussion to provide a more comprehensive picture of the participants and how the learning environment is situated within their lives. Culture is a conglomeration of the practices, ideas, language, stories and beliefs from the various communities and societies that an individual has participated in (Geertz, 1973; A Swidler, 1986). It is important for understanding values and tied to how one is able to act in relation to their values. Swidler asserts that culture impacts action, “*not by providing the ultimate values toward which action is oriented, but by shaping a repertoire or “tool-kit” of habits, skills, and styles from which people construct ‘strategies of action’*” (A Swidler, 1986). While culture might be just as important as values for understanding how and why students are motivated or engaged in a learning environment, a focus on values provides an

operational framing to center design and analysis of the learning environment in a way that is considerate of cultural influences and competencies. The research presented contends that by integrating values into the discussion of design and analysis of learning environments we can create equitable environments that scaffold students with different cultures to participate and learn with computing.

As a design approach, values-based learning seeks to create inclusivity and equity in a learning experience through providing opportunities for students to reflect and integrate their values into the learning environment. As a lens for analysis, it provides insight into the ways in which values manifest in the environment. By using a framing that takes both culture and values into consideration, we can understand where the learning environment was inclusive and where it was neglectful of different students. However, it is important to note that identification of these instances is still limited to what the participants make visible—i.e. how they participate (or don't), the reasons they give for participating (or not participating), their dialogue and reactions throughout the experience, how they reflect on their experience, etc.

In this chapter, I present the background literature necessary to situate values-based learning. First, I present the related research on values and culture to provide and understanding how values-based learning incorporates ideas from these two bodies of work. Second, I contrast this approach to the literature exploring interest and culture in education in order to differentiate its contributions. Next, I present infrastructure as a way to frame the learning environment in order to explore values and advocate for participatory design as a method to support reflection and integration of values. I cover how the perspective on values and infrastructure can be situated in the context of other theories

surrounding project-based work. I then introduce the literature to describe the constructs of the self-concept and boundary objects, which will be used to situate some of our findings.

2.2 Situating the Values-Based Learning

2.2.1 Values and Value Domains

Graeber provides a comprehensive overview of the various thematic categories in how researchers have discussed values (Graeber, 2001):

1. *'values' in the sociological sense: conceptions of what is ultimately good, proper, or desirable in human life*
2. *'value' in the economic sense: the degree to which objects are desired, particularly, as measured by how much others are willing to give up to get them*
3. *'value' in the linguistic sense, which goes back to the structural linguistics of Ferdinand de Saussure (Saussure, 1966), and might be most simply glossed as 'meaningful difference'*

Values-based learning uses values in alignment with the first definition describing them in terms of the ideals of what people strive for. Graeber states that values are not just what people desire, but *"they are the criteria by which people judge which desires they consider legitimate and worthwhile and which they do not."* Defining values in this way is important for interpreting the work I present for two main reasons. First, it means that values are something that can be identified by the individual who holds them and can be internally

and externally reflected on. Second, it aligns with the perspective that one's culture and self-concept can be instrumental in influencing the values one holds.

In order to have a language to talk about values, values-based learning integrates the idea of value domains (or high-level groupings of values) from Schwartz et al. (Bardi & Schwartz, 2003; S. Schwartz, 1999; S. H. Schwartz, 1992). The value literature has a body of work concerned with the need to identify what is and is not valued by people. Rokeach, for example, claims that values “*serve as standards or criteria to guide not only action but also judgement, choice, attitude, evaluation, argument, exhortation, rationalization, and one might add attribution of causality*” (Rokeach, 1979). Rokeach outlines a list of 36 values which represent desirable end-states (ex. wisdom, equality, happiness, etc.) and behaviors that would presumably help one reach those end-states (ex. self-control, obedience, helpfulness, etc.) (Rokeach, 1979). While the value survey that Rokeach developed has been used in many fields, it has been criticized for being incomplete, disregarding multiple interpretations of the values, and neglecting to consider the context and culture of the values that are included on the list (Gibbins & Walker, 1993; K. C. Lee, 1991).

Schwartz builds upon this work with the idea of value domains in order to take cross-cultural differences into account and to build an argument for the causal link between values and action. Schwartz conceptualizes value domains as *criteria* for selection rather than *qualities inherent in objects* (S. H. Schwartz & Bilsky, 1987). This would be the difference between analyzing whether or not a student *values* math as a subject, compared to analyzing what socio-cultural influences, such as maintaining a social status amongst their peers, might affect how a student places importance on math. In this work, Schwartz

et al. have examined 88 populations across 38 countries validating the existence and persistence of these value domains across cultures and societies (Bardi & Schwartz, 2003; S. Schwartz, 1999; S. H. Schwartz, 1992; S. H. Schwartz & Bilsky, 1987). They have found that the relations between value domains, or the *compatibilities* and *contradictions* that arise between them, provide an understanding of different cultures (S. H. Schwartz & Bilsky, 1987). Furthermore, the differences between how the underlying values manifest within various environments provides insight into these cultures. Table 2 (originally published by Bardi and Schwartz (Bardi & Schwartz, 2003)) outlines these value domains.

Table 2 – Definitions of Value Domains (replication of table from (Bardi & Schwartz, 2003))

Value Domain: Definition of domain
Power: Social status and prestige, control or dominance over people and resources
Achievement: Personal success through demonstrating competence according to social standards
Hedonism: Pleasure and sensuous gratification for oneself
Stimulation: Excitement, novelty, and challenge in life
Self-direction: Independent thought and action-choosing, creating, exploring
Universalism: Understanding, appreciation, tolerance and protection of the welfare of all people and of nature
Benevolence: Preservation and enhancement of the welfare of people with whom one is in frequent personal contact
Tradition: Respect, commitment and acceptance of the customs and ideas that traditional culture or religion provide the self
Conformity: Restraint of actions, inclinations, and impulses likely to upset or harm others and violate social expectations or norms
Security: Safety, harmony and stability of society, of relationships, and of self

Within the literature, Schwartz’s mapping between values and action has been criticized for the discrepancies between what people say is important to them and what they actually end up doing (ex. (Cancian, 1975; DiMaggio, 1997; A Swidler, 1986; Vaisey, 2008)). However, value domains are useful for talking about values because it has shown that these

categories exist within and across most cultures. I therefore apply this language in my work using it to describe how values manifested within the environment, while using the data from the studies to understand the specifics behind the values implicated. Because of the difficulties with using values as the direct causal link to action, my work acknowledges the role culture plays in how students participate in the learning environment.

2.2.2 *Integrating Culture into Our Understanding of Values and Action*

Researchers have turned to culture as a way to provide a better understanding of why people's actions might not directly align with what they value (*see* (Lizardo & Strand, 2010) for a review of culture and cognition). While previous trends in sociology and anthropology have defined culture as clean and coherent across situations and people (ex. (Geertz, 1973)), the current perspectives diverge from this view. Instead, they understand culture as variable, disjoint, and situationally cued (DiMaggio, 1997). This shift in understanding stems from developments in cognitive psychology that demonstrate that most of human action is automated due to the cognitive strain it would take to deliberately weigh and choose all courses of action (Evans, 2003; Kahneman & Tversky, 1982). Researchers have therefore conceptualized culture's role in action as dependent on the mixed bag of *repertoires*, *schematas*, or *tool-kits* that are encoded and often automatically drawn upon dependent on situational characteristics (DiMaggio, 1997; Lamont, Schmalzbaure, & Waller, 1996; Lizardo & Strand, 2010; A Swidler, 1986; Ann Swidler, 2000, 2013; Vaisey, 2008, 2009). As Lizardo and Strand summarize and cite, "*persons do not (and cognitively cannot) internalize highly structured symbolic systems...These cultural systems are simply too 'cognitively costly'—in anthropologist Harvey*

Whitehouse's sense (Whitehouse, 2004)—*to be capable of being strongly 'internalized' by anybody*” (Lizardo & Strand, 2010).

Swidler aligns with these perspectives, specifically critiquing the values-based explanation of action because it neglects to account for culture's role in making specific goals and thus values attainable in particular contexts. She defines culture as “*symbolic vehicles of meaning, including beliefs, ritual practices, art forms, and ceremonies, as well as informal cultural practices such as language, gossip, stories, and rituals of daily life*” (A Swidler, 1986). Culture thus impacts action, “*not by providing the ultimate values toward which action is oriented, but by shaping a repertoire or “tool-kit” of habits, skills, and styles from which people construct ‘strategies of action’*” (A Swidler, 1986). Depending on how a person is equipped to navigate a particular situation based on the *strategies of action* they can construct, her/his culture could bias her/him towards actions that prioritize certain value domains. Swidler further asserts that while culture is not determinate of the values people hold, one's cultural capacities still influence what those values are. In this research, values-based learning uses values as a way to conceptualize whether or not something was meaningful for a student, while understanding their participation in the learning environment as being influenced by their values and their cultural *tool-kit*. By understanding how students participated and how certain value domains manifested in the learning environment we can understand where the learning environment was equitable and where it could be improved.

2.3 Differentiating Values-Based Learning

In the educational literature, there have been various theoretical frameworks and stances used to help determine how to create personally meaningful educational experiences for students. The research surrounding values presented in this dissertation adds to this conversation. I will situate the distinguishing contributions of values-based learning in the literature involving interest and culture in education.

2.3.1 *Interest in Education*

Within the education literature, interest is often intertwined with the value a student holds for a discipline or activity. Investigating the intricacies of student interest on learning has been explored since the 1800s (William, 1899). While nuances exist between the exact perspectives that researchers take on interest, most refer to it as something that is based on the individual and their disposition towards wanting to engage in a domain or task (Azevedo, 2013; Hidi, 2000; Renninger, Hidi, & Krapp, 2014). The definition of interest is similar to the Value-Expectancy literature's definition of the *value* a task holds (Eccles, 1992; Eccles & Wigfield, 2002). Interest is seen as both *individual*—a person's desire to engage with a subject over time—and *situational*—attention sparked “*in the moment by environmental stimuli*” (Hidi & Renninger, 2006). Hidi and Renninger (Hidi & Renninger, 2006) and Alexander (Alexander, 2004) identify phases to understand how interest begins *situationally*, in the absence of domain knowledge, and changes overtime to an *individual* well-developed interest. While this work has been instrumental in relating both the cognitive and affective components of motivation, the construct of interests does not provide an understanding of why certain things are individually or situationally interesting

for certain students, or why something will sustain one learner while another learner disengages. Interest limits the unit of analysis to the individual, which causes an inadequate understanding of the influences in the socio-cultural context. Edelson and Joseph's (Edelson & Joseph, 2004b) Interest-Driven Design framework suggest five sources of usefulness—*pleasure, concern, identity formation, life goals, and curiosity*—that begin to take socio-cultural characteristics into account. However, even with these suggestions, interest-driven learning struggles from, misalignment between the interest and material, and the students missing the relevance (Edelson & Joseph, 2004b). Values-based learning attempts to provide a lens that takes the socio-cultural context of the students into consideration, while providing a method to help participants create the link aligning their values to the material.

2.3.2 *Culture in Education*

Culture is strongly integrated with our understanding of values and their implications within particular contexts. Similar to Swidler, we view culture having an essential role in the social construction of values and their organization (A Swidler, 1986). Educational researchers throughout several disciplines have demonstrated the importance of integrating culture into education (ex. (Au, 1980; Au & Kawakami, 1994; G. Ladson-Billings, 1995; Gloria Ladson-Billings, 1990)). The work's importance lies in cultures impact on students' values and their ability to act within the learning environment. For example, Au's work integrates the cultural practices of native Hawaiian children to support students to succeed in educational environments where they otherwise were failing (Au, 1980). Ladson-Billings builds on this work describing "*students' culture as a vehicle for*

learning,” highlighting how culture can be incorporated to enable “*students to ‘choose’ academic excellence*” (Gloria Ladson-Billings, 1995a).

While Culturally Relevant Teaching has gained widespread acknowledgement and has been applied to many domains including computing (Eglash, Gilbert, Taylor, & Geier, 2013; Scott, Clark, Hayes, Mruczek, & Sheridan, 2010; K. A. Searle & Kafai, 2015), it is still unclear how to scaffold teachers to become culturally responsive as the successful teachers are often viewed as *heroic isolates* (Gloria Ladson-Billings, 2008). Ladson-Billings challenges teachers to view the curriculum as a malleable cultural artifact which can be *deconstructed* and *reconstructed* in response to its inherent biases. Gay provides similar suggestions stating that we need to teach teachers “*how to do deep cultural analyses of textbooks and other instructional materials, revise them for better representations of cultural diversity*” (Gay, 2010, p. 108). However, neither Ladson-Billings or Gay indicate how to support an untrained teacher to accomplish these difficult and complex tasks. Using a lens focused on values can inform the deconstruction, through identifying points of compatibilities and contradictions between the students’ values, educators’ values, and the values espoused in the curriculum. By drawing attention to these aspects of the learning environment we can better understand how curricula can support students who have differing values. We offer infrastructure as a way to frame the learning environment for design and analysis from the perspective of values, while integrating participatory design into the infrastructure order to guide the students and instructors to co-design learning projects that incorporate a diversity of values.

Within the computing domain, researchers have explored this idea of using a student’s background, interests, and culture to stimulate a greater motivation and alleviate

some of the issues surrounding the underrepresentation of women (ex. (Scott et al., 2010; Scott & Hood, 2009)) and minorities (ex. (Eglash et al., 2013; Franklin, Conrad, Aldana, & Hough, 2011; K. A. Searle & Kafai, 2015)). Eglash, for example, demonstrates success through integrating culture into computing curricula through the lens of *culturally responsive* technologies, which have their cultural relevance embedded within the design (Eglash et al., 2013). However, with the diversity of students and the dynamics of changing culture, these static designs have a narrower appeal and shorter shelf life than a more dynamic approach which would seek to help students integrate their culture and values.

Scott and Hood (Scott & Hood, 2009) create a *culturally responsive* computing environment using digital media and *Teen Second Life*. In accordance with Ladson-Billings view of culturally responsive education, Scott and Hood, facilitated the girls' critique of society through the students' identification of a social/community issue that they research within the first course of their COMPUGIRLS program. Throughout the six-course program the students practice enacting social change through the virtual world of *Teen Second Life*. They participate in, plan for, and execute their proposed projects designed to improve the social/community issue they have chosen in the safety of this virtual world. At the end of the program the girls present their work in a public forum (Scott & Hood, 2009). The program highlights the importance of the authentic problems, the agency the students gain over the technology, and the visibility the students had presenting their projects. While they demonstrate a dynamic culturally responsive program, they do not offer guidance for producing these types of programs.

Blikstein focuses on similar goals within his learning environments, examining technology, "as an emancipatory tool for mobilizing change in schools and empowering

students” (Blikstein, 2008). He draws inspiration for his work from Freire, although Freire is more radical he has similar ideas to Ladson-Billings, in that the educational system should be developed based on the students, their community, and the needs they have. Blikstein points to key factors to enable a *Freirean learning environment*: (1) incorporate a generative theme relevant to the community, (2) use the community’s culture as a basis for introducing technology into the learning environment, (3) use a mixed media approach for expressive tools, and (4) critique standard practices in school-based education (Blikstein, 2008). Similar to Ladson-Billings, the structure of the educational environment is meant to inspire critique of current standards in society. The guidelines of this work provide goals for physical computing learning interventions to be culturally adaptive to the students. We build upon this work by incorporating methods to help educators and students reach these goals.

2.4 Infrastructure to Deconstruct the Learning Environment

Values-based learning emphasizes understanding how values manifest in the learning environment. This makes it important to be able to deconstruct the learning environment to understand the impact of the individual design decisions. I advocate for the use of the concept of infrastructure. *Infrastructure* was initially defined by Star as *something* that emerges within a work environment to facilitate interaction between people and exists in a form that is ready to be appropriated for local practices to service individual needs (Star & Griesemer, 1989b). Infrastructure was chosen because it draws your attention to where the learning environment integrates constraints as well as flexibility for students to navigate through the activities. We appropriated the *infrastructuring strategies* from Ehn as a way to identify the types of infrastructure in the learning environment (Pelle Ehn,

2008). Ehn builds on Star's work identifying four types of *infrastructuring strategies* (Pelle Ehn, 2008) (we substituted *design patterns* with *design cases*):

1. **Formats**—*pre-defined solutions with the important characteristics highlighted. These characteristics can then be flexibly applied to new situations based on the user's knowledge of the process to appropriately modify characteristics.*
2. **Component Strategy**—*LEGO block approach in which the user can build solutions for specific problems they encounter using the components provided*
3. **Design Cases**—*design examples that can serve as cases for the students to reason about and appropriate to new situations (Maher & Gomez de Silva Garza, 1997). They are described in terms of context, problematic situation, and proposed solution (Pelle Ehn, 2008)*
4. **Protocols**—*within a social context are the defined procedural agreements for completing activities and/or communicating*

These four infrastructuring strategies can be seen as a form of *distributed scaffolding* in which the scaffolding is “*distributed across the tools and the context in which the learning occurs*” and spans learning within and between educational activities (Puntambekar & Kolodner, 2004). The infrastructuring strategies can be seen as the structure that enables (or inhibits) students to recognize, share, and integrate their values within an educational context. We use these infrastructuring strategies as a way to draw attention to the design of the scaffolding in the learning environment that impacts students' ability to create experiences driven by their values.

2.4.1 Participatory Design as a Method for Understanding Values

In order to promote and guide students to reflect on their values, we integrated participatory design activities into our studies. To achieve equitable experiences, it is essential to understand what values are embedded within the design processes we use to create learning environments. As Beista asserts: "values are not simply an element of

educational practices, but that they are constitutive of such practices" (pg501). It is imperative that we actively practice design that considers the values of all stakeholders. Participatory design espouses democratic values providing opportunities for all involved stakeholders to have a voice (P Ehn, 2008; Simonsen & Robertson, 2012). Since its inception in Scandinavia in the 1970s it has been applied across several domains including researchers who have bridged these ideas into the learning sciences (Bonsignore et al., 2013; B DiSalvo & DiSalvo, 2014; Betsy DiSalvo, Yip, Bonsignore, & DiSalvo, 2017). Bonsignore et al. advocate for its use as a resource for creating "collaborative, socially constructed learning" experiences (Bonsignore et al., 2013). The research presented demonstrates use of participatory design methods that engage students in design activities whose goals are "not simply for instrumental ends" (B DiSalvo & DiSalvo, 2014) but instead support the students to self-reflect as they express themselves. By acknowledging the student as the expert in her or his own values, participatory design provides legitimate processes for student values to play a supportive role in shaping the learning environment. We contrast these methods with trying to predict student values which would "raise concerns about both individual differences and reinforcement of stereotypes" (Edelson & Joseph, 2004b).

2.5 Infrastructure in Project-Based Work

Designing learning environments using infrastructure and participatory design methods can be used to complement project-centric approaches to learning. We explore how infrastructure can be situated within two central approaches to experiential learning: constructionism and project-based learning. Constructionism involves actively engaging students in exploration of concepts through creation of socially situated artifacts (Seymour

Papert, 1980; Seymour Papert & Harel, 1991). Implementations of constructionism that give the learner full autonomy over choosing their projects allows for self-directed learning; however, it takes a skilled mentor to navigate this open-ended project environment and ensure that the student is deeply engaged with the material. Blikstein, for example, describes how he incorporated semi-structured projects in a fabrication lab to scaffold the learners to become acquainted with the tools. However, once they were introduced to the laser cutter, Blikstein encountered what he called the *keychain syndrome* in which the learners only wanted to create simple, aesthetically pleasing projects instead of diving into something more complex (Blikstein, 2013a). While some might argue that open-ended constructionism is values-based by design since learners can explore what they want, I believe that students need scaffolding throughout the design process to reflect on what they find valuable so they can integrate these values into the designs of their artifacts. Specifically, I posit that participatory design can support students in choosing projects that appeal to their core values beyond a passing fancy, while having the capacity to maintain their engagement through the complex and difficult tasks. Analysis of the infrastructure can highlight the important design characteristics in the learning environment for supporting the students.

While the research behind project-based learning offers more structure for creating and evaluating physical computing learning activities, it too could benefit from integrating participatory design methods. Barron et al. present four main design principles for project-based learning: (a) define learning-appropriate goals, (b) provide scaffolds for educators and students, (c) create situations for formative self-assessment and revision, and (d) promote participation and a sense of agency (Brigid J.S. Barron et al., 1998). However,

even with these guidelines, it can be difficult to choose the appropriate questions or problems in order to incite engagement throughout a difficult project. Barron et al. state, “we have little systemic empirical information about what problems students actually find valuable, interesting, or useful enough to work on for long periods” (Brigid J.S. Barron et al., 1998).

While both constructionism and project-based learning have the possibility to engage novices through physical computing activities, neither provides a guiding method or process to make this happen. I propose incorporating the construct of infrastructure to understand the decisions in the learning environment that help support project-based work. Further, I advocate for integration of methods such as participatory design which can scaffold students to reflect on their values such that they can choose themes that speak to their values and provide avenues to integrate and discuss their culture.

2.6 Supporting Theoretical Constructs

Within our analysis we found the constructs of the *self-concept* and *boundary objects* important for exploring our findings from the case studies presented in the next chapter. I will briefly cover the background literature important for understanding them.

2.6.1 Adolescent Self-Concept

I use self-concept as a way to understand how students view themselves in relation to the learning environment. Students within underrepresented populations in computing often perceive the identity of a computer scientist as one that does not align with their culture or values (Martin, 2004; Shashaani, 1993). This misalignment of identities and values often

leads to rejection of opportunities to participate in computing (James DiSalvo et al., 2011; Jane Margolis, 2008). Identity is a complex concept with many theories and methods for exploration. One way to explore the underlying identity misalignment is through the processes of students forming congruence or dissension between their self-concept and computing. The self-concept can be viewed as a set of self-beliefs that can be thought of as an internalization of the student's culture, values and how she views herself.

One's self-concept is composed of abstract self-beliefs that become recognized and more organized throughout adolescence (Steinberg & Morris, 2001). It consists of multiple dimensions of varying hierarchy. Part of the hierarchical structure of the self-concept is the stability of certain self-beliefs (Shavelson, Hubner, & Stanton, 1976). At the bottom are the least stable beliefs, which are the ones that are most easily influenced. These consist of decisions about the self in relation to specific experiences, "not only does the individual develop a description of himself in a particular situation or class of situations, he also forms evaluations of himself in these situations" (Shavelson et al., 1976). As you go up the hierarchy of stability, the beliefs become more abstract and contain more stable beliefs about the self in a general sense. For example, if someone does well on an algebra test they now have an experience in which they proved their knowledge of algebra, which then might support a self-belief that they are *good at math*. Aggregation of the experiential self-beliefs begins to shape the higher order beliefs about the self (Shavelson et al., 1976). Figure 1 below is a recreation of the diagram from Byrne and Shavelson of the hierarchical structure of the adolescent self-concept concerning academics (Barbara M Byrne & Shavelson, n.d.; Shavelson et al., 1976).

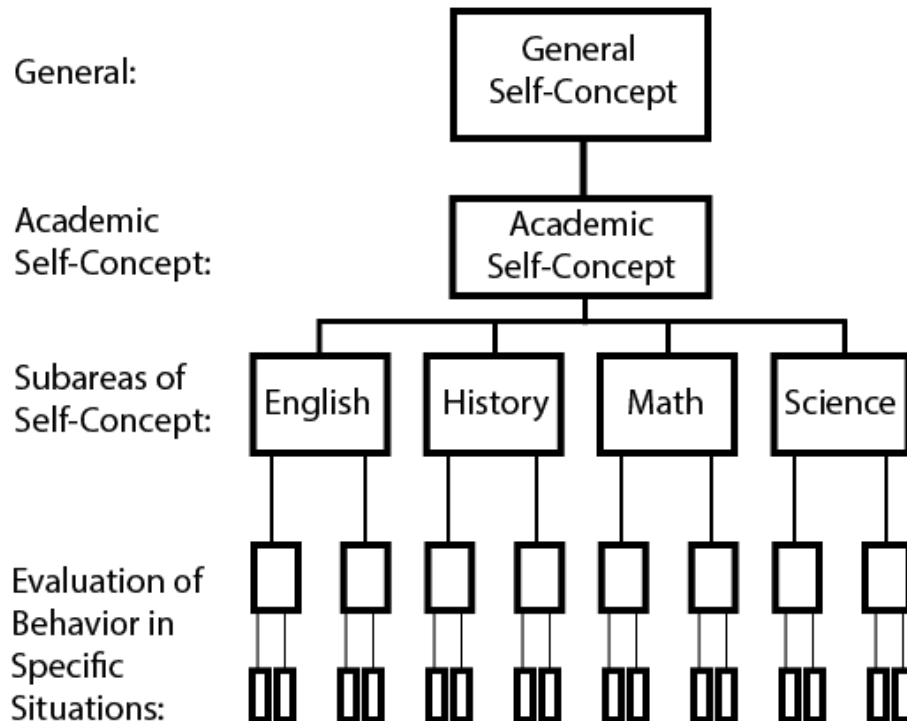


Figure 1 – Structure of academic self-concept (image remade from Shavelson et al. (Barbara M Byrne & Shavelson, n.d.; Shavelson et al., 1976))

In Byrne and Shavelson’s investigation they illustrate a multidimensional organization of self-concept with “perceptions of behavior at the base moving to inferences about the self in sub-areas (e.g., academic—English, mathematics), then to inferences about the self in academic and non-academic areas, and finally to inferences about the self in general” (B. M. Byrne & Shavelson, 1987). Swann brings attention to the importance of the social environment in which these experiences are situated. While the self-concept is internal it is externalized with the actions of the students. Because one’s actions are socially situated in the world there is a reciprocal relationship between how their actions shape and are shaped by the social environment (Swann, 1983). Within these dimensions adolescents begin to develop *self-occupational congruence* which aligns one’s self-concept to the roles one associates with certain occupations (Betz, 1994). Between the ages of 10 to 18, the

exploration of self-concept in terms of occupational roles increases significantly (Montemayor & Eisen, 1977) signifying middle and high school as an important time if we hope to influence how students view themselves in relation to computing occupations.

In the context of our studies, the students have various experiences with computing and the learning environment. These experiences shape their self-concept through either supporting or conflicting with their self-beliefs. We define the process of creating alignment between a student's self-concept and how she views computing as forming *congruence*, and we refer to *dissension* as the process when a student breaks or distances her self-concept from how she views computing. Going back to our definition of culture as, "a *tool kit* of symbols, stories, rituals, and world-views, which people may use in varying configuration to solve different kinds of problems" (Ann Swidler, 1986). A student's self-concept and values affect which problems they choose to solve and how they apply this *tool kit* to their environment. By drawing attention to the students' self-concepts, we are enabled to understand the students' motivation for both applying their culture in various ways in the learning environment and for understanding their decisions for choosing certain problems in the learning environment.

2.6.2 *Boundary Objects*

Within our research we found that values-based learning experiences were facilitated through the artifacts the students and educators created. These artifacts served as boundary objects in which various interpretations and values are embedded. Star and Griesemer (Star & Griesemer, 1989a) present boundary objects as an analytical framework to interpret the collaborative practices surrounding Berkeley's Museum of Vertebrate

Zoology. They used the framework to describe how scientific findings, which often had vastly different meanings and implications depending on the social world, could have coherence (Star & Griesemer, 1989a, p. p132). They further expand on these characteristics:

Boundary objects are objects, which are both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites. They are weakly structured in common use and become strongly structured in individual site use. These objects may be abstract or concrete. They have different meanings in different social worlds but their structure is common enough to more than one world to make them recognizable, as means of translation. (Star & Griesemer, 1989a, p. p132)

Star (Leigh Star, 2010) emphasizes that the term go beyond interpretive flexibility of objects to include understanding how the object influences the “structure of informatic and work process needs and arrangements” within communities as well (Leigh Star, 2010, p. p601). Star clarifies that boundary objects were originally created to describe the phenomena in which opposing forces were able to work together without reaching consensus. She points to the importance of scope and scale of the boundary objects in order to understand the affordances of the object based on its materiality and infrastructural properties (Leigh Star, 2010, p. p601).

This definition has been applied in the educational domain in a few contexts. For example, boundary objects were used as a framework for examining the success of curriculum change that progressed in spite of conflicting powers (Hultén, 2013). The

framework has also been used in the context of individual courses, such as understanding the success of an authentic science course in which students wrote for science journals, bridging them into communities outside of their schools (Polman & Hope, 2014). Wenger also examines learning at boundaries in the context of communities of practice (Wenger, 1998). Outside of these individual explorations, Akkerman and Bakker (Akkerman & Bakker, 2011) completed a literature review to offer insight on conceptualizing the use of boundary objects and boundary crossings in the education domain. Their analysis reveals four ways in which learning occurs at boundaries: *identification*—understanding the diverse perspectives and practices in relations to one another; *coordination*—how the cooperation between the perspectives and practices is regulated; *reflection*—creating opportunities for one to think about and expand their perspectives in relation to others’; and *transformation*—creating new practices based on the intersection of boundaries (Akkerman & Bakker, 2011, p. p1). Throughout the MoveLab we saw how the students and leaders *identified* differences in their perspectives, created a structure for the *coordination* between the disciplines and perspectives within the workshop, and created opportunities for *reflection* through their discussion of differences. Due to the brevity of the workshop described in our case study, we did not witness new practices being developed. However, through our analysis we reveal how the boundary objects served to foster abstract thinking amongst the students and leaders during collaboration periods in which the objects were more weakly structured. These affordances enabled values-based learning even when values might have differed between students in the same group. The abstraction of the objects allowed them to work together while having disparate concrete views of the objects on an individual level.

In the next chapter I present two case-studies on values-based learning: the MoveLab workshop and the Day of the Dead Puppet summer camp. I first present the MoveLab workshop's infrastructure and findings from the perspective of the self-concept and boundary objects. I then present how we iterated on the infrastructure for the Day of the Dead Puppet camp and present the projects of the participants that came out of this experience. I end with conducting a cross-analysis between the two learning environments from the perspective of values-based learning, discussing how the choices within the infrastructure affected the values in the environment.

CHAPTER 3. TWO CASE-STUDIES OF VALUES-BASED LEARNING

3.1 Introduction

Values-based learning was developed through the design and analysis of two physical computing learning experiences: the MoveLab Workshop, and the Day of The Dead Puppet Summer Camp. The original goal focused on understanding:

How can we design a flexible and dynamic physical computing learning environment that accommodates the culture and values of students to create personally meaningful experiences with computing?

The MoveLab study preceded the Day of the Dead Puppet study and was designed to investigate the characteristics of a dance and computing program that impacted how students integrated their values in the learning environment. We used the construct of the self-concept to understand how students' culture and values, that are embedded in their self-beliefs and actions, affected how they interacted within the learning environment. These interactions created either congruence or dissention between their self-concept and computing. Within this intervention we found that one benefit in having students create computational art together is that their creations served as boundary objects that could be interpreted differently by each person within the learning environment. Our post-analysis of the learning environment found that we had supported participants with different infrastructure that was important for enabling the flexibility of the student work and integration of their values into the learning environment.

The Day of the Dead Puppet summer camp was designed using similar infrastructure with an activity that had shown success across several populations. The activity was originally designed for a three-day afterschool program for middle school Latinx students to understand how to promote reflection of both culture and technology (13 participants). Based on the success of the activity, it was expanded to include both the planning and realization of the computational artifact in two semesters of a graduate-level prototyping course in order to motivate their final project (44 participants in total). Students in the graduate level course were a diverse group, representing a number of countries. The activity was then replicated in a summer camp with high school students, which gave us an opportunity create a similar infrastructure in the learning environment as we did for the MoveLab. We were able to compare the two studies to provide a greater understanding of how the infrastructure assisted and detracted from participants ability to integrate their values into the learning environment. The studies contributed to understanding the following research questions:

***RQ1.** How does design of physical computing learning environments affect students' ability to integrate their diverse values into their learning experience?*

- 1.1 How can we conceptualize the design of the learning environment from the perspective of students' diverse values?
- 1.2 How can we analyze the learning environment from the perspective of students' diverse values?
- 1.3 What characteristics of the learning environment in physical computing activities impact students' ability to integrate their diverse values into their learning experience?

I will describe the MoveLab, and examine our findings using the constructs of self-concept and boundary objects. I then cover the infrastructure in the Day of the Dead Puppet summer camp, discuss what the participants created, and end with a cross analysis using a lens focused on values to examine the findings from both the MoveLab workshop and the Day

of the Dead Puppet summer camp. In this analysis of the learning experiences, I examine the design choices in the infrastructure to articulate how the various aspects of the learning environment were supportive, neglectful, and in conflict with student values.

3.2 The MoveLab Workshop

A large portion of the work presented in this section was published in the Proceedings of the ACM Special Interest Group on Computer Science Education (DesPortes, Spells, et al., 2016b) and the ACM Interaction Design and Children (DesPortes, Spells, & DiSalvo, 2016a).

3.2.1 Introduction and Context of Study

The MoveLab was a five-day workshop investigating a community of learners (B. Rogoff, 1994) as they embarked on an endeavor to create dance performances integrating technology. The dances were designed by the students to be representative of issues that the participants cared about. The leaders consisted of two lead researchers (both women, who also served as computing leaders), two additional computing leaders (both men), and four dance leaders (all women). Leaders collaborated with 13 middle school and high school girls (ages 11 to 15). We recruited students from Enchanted Closet, a community organization that supports underserved girls and young women (ages 11 to 19) in the Atlanta-Metro area. They focus on building self-esteem, creating healthy living habits, and broadening the girls' horizons by providing them with new experiences. We used this method for recruitment because we intended the workshop to be an outreach program to engage students who might not typically participate with CS and engineering.

The workshop was held at EyeDrum, a local performing arts center, and took place during the students' Thanksgiving school break. The workshop ran from 10am to 5pm for the first four days and ended with a half day, running from 10am to 2pm, on the fifth day. Student participation fluctuated throughout the workshop due to family commitments and other obligations. Two weeks after the workshop, there was a Saturday rehearsal and a Sunday public recital open to the public, allowing the students to present their dances to a broader audience.

3.2.2 Data Collection and Analysis

We administered pre-study surveys to gather background information on the students and capture a baseline understanding of their interests in dance and technology. Post-study surveys gathered information about summative changes in perspectives about dance and technology. Due to the variability in participation, only 6 students completed both the pre- and post-study surveys. In conjunction with the surveys, we conducted observations, and had the leaders capture their real-time observations in a journal. At the end of each day we audio recorded a debrief session with the leaders to gather their observations. We then conducted semi-structured interviews with 6 students and 4 leaders at the end of the workshop. Interviews asked about learning and teaching opportunities during the workshop, the different ways participants contributed to the performance, the participants' use of technology and dance, and their self-concept in relation to technology and dance. The interviews were audio recorded and transcribed.

The interviews, observations, and leader debriefs were reviewed and analyzed for emergent themes by two researchers. They used pattern coding (Saldaña, 2012) to solidify

themes which identified self-concept as a construct that could be used to organize and understand our data. The analysis led to the codes within the codebook (Table 3) to describe the data. After the codes were agreed upon, the two researchers reviewed the data, identified any missed or controversial codes, and discussed and resolved any disagreements. The researchers identified codes within the codebook, which identified characteristics that led to both congruence and dissension between the students' self-concepts and the disciplines of computing and dance.

Table 3 – Codebook for Understanding Self-Concept

Developing Congruence	
Code: Description	Example
<i>Agency: Taking initiative to get involved</i>	<i>Some of the students wanted to cut the wires and build the circuits while others jumped to program on the computers.</i>
<i>Pride: Pride over success with material</i>	<i>Once I caught onto the computer programming and...hooking up different wires ...I would help the other people in my group.</i>
<i>Narrowing Expertise: Narrowing focus to a particular topic</i>	<i>Students noticed that I'm a little bit more hands-on for the technology side of things and the lead dance instructors are more hands-on for the dancing side of things.</i>
<i>Developing Identity as a Group: Working within the social norms of the group</i>	<i>When it came to coming up with the different sensors and building the sensors, I was more on that end and my other group member was helping more with the dance and costumes.</i>
<i>Expression of Values: Expressing personal values through work</i>	<i>I worked on a project, one that really got me thinking about what I didn't like in the world and what I didn't like within my community.</i>

Table 3 – Continued

Developing Dissent	
Code: Description	Example
<i>Preconceived Ideas:</i> Previous ideas influencing participation with material	<i>I would never have thought to put dance and technology together. Until, of course, this workshop.</i>
<i>Avoidance of Failure:</i> Avoiding activities for fear of failing	<i>She was putting on a facade of “I don’t care” but in reality, she was too nervous to do anything.</i>
<i>Conflicting Values:</i> Previous self-concepts conflicting with work	<i>Because I’m not that big of a dancer...This was definitely a new experience for me, in terms of working with actual dancing. I’ve never done that because I’m pretty...very STEM focused.</i>
<i>Group Resistance:</i> Resisting participation against the social norms of the group	<i>She was having a lot of trust issues in the teamwork because we were trying to build teams and it was about teamwork and especially in the very beginning, she was really anxious letting the other partner even touch [the sensor].</i>

Within analysis of this data the researchers also identified a code for boundary objects that marked instances in the data where there was coordination between the participants facilitated in the presence of their differences through the computational artifacts they created. We discuss the infrastructure in the learning activities and then go into the findings around self-concept and boundary objects.

3.2.3 *Infrastructure in the Learning Environment*

The MoveLab and the Day of the Dead Puppet activities were structured in similar ways in order to support the participants. The overall goal of the infrastructure was to provide enough structure and guidance so they could successfully create with computing in an open-ended environment, while providing enough flexibility for participants to create something that was personally meaningful. The infrastructure integrated *components*,

design cases, formats and protocols (as discussed in section 2.4) and served four main purposes: first, to help participants gain proficiency and knowledge in the domains they were working with; second, to provide guidance and opportunities for participants to self-reflect on their values; third, to facilitate a shared dialogue around participants' values; and fourth, to organize the work processes of the participants. While each piece of the infrastructure usually served multiple of purposes (ex. an organizational strategy that also stimulated discussion of values), they will be presented with their main purpose.

3.2.3.1 Infrastructure for Building Proficiency and Knowledge

The learning environments integrated infrastructure designed to provide a set of resources and to develop a base of knowledge for the participants to dynamically draw on within the environment as needed. Introducing the participants to specific components and design cases was intended scaffold participants into making informed decisions when it came to applying the components in their projects. This differs from a completely open environment where components exist but students do not have the base knowledge to know what components make sense for their ideas.

The workshop was seeded with a set of physical and conceptual components that were intended for the participants to build from. They had a set of electronic tools at their disposal to use throughout the workshop including Arduinos, various types of sensors and actuators (ex. accelerometers, capacitive touch sensors, RGB LED strips), breadboards, soldering irons, etc. The participants were also introduced to various dance dynamics (ex. size, speed, location, etc.) and choreographic techniques (ex. call-and-response, ripples, cannons, etc.).

The researchers guided participants through constructing a set of design cases with both the electronics and the dance concepts to help the participants gain proficiency in using them. The intention was to provide the students with examples of how they might apply the various electronic pieces and dance techniques. For the electronics, students built a set of predetermined design cases that each used different electronic components. Each pair of participants was randomly assigned to build one of the design cases. The idea was to create disparities in what the students worked on so they could learn from one another. We embedded peer-teaching exercises, such that once the participants had built their Arduino circuits, they would present to their peers. This teaching exercise served as a protocol for communication between the students and was intended to set a precedent for the students to teach and learn from one another throughout the workshop. The constructed design cases served as a starting point for the students to remix the circuit and code as they appropriated the components into their dances.

Within the dance activities a similar approach was used, but there was an added flexibility in the exercises. The dance leaders encouraged the students to apply the dance components themselves; for example, tasking students with coming up with a phrase (a group of dance moves) that represented their names. The students then had to think about the dance dynamics—i.e. the size, speed and shape of their body—as they constructed poses and transitioned between them. The phrases that the students developed served as design cases which they could then bridge into their dances. Throughout the workshop during the lunch breaks, we also exposed students to professional design cases of artists who couple dance and technology, and music and technology. This gave the students

examples of how professionals are integrating technology into their work and served as a tool to analyze what these artists were expressing and how they were doing it.

3.2.3.2 Infrastructure for Reflecting on Values and Creating a Shared Dialogue

The environment encompassed infrastructure designed to promote self-reflection on the participants' values in order to create meaningful dialogue to seed the designs of their dances. In both the MoveLab workshop and the Day of the Dead Puppet camp, we set the participants up with driving questions that we felt were constrained enough to provide guidance but open-ended enough to enable participants to integrate their values into their reflections of the question. We engaged the participants in participatory design activities surrounding these questions in order to guide their self-reflection as they designed diagrams and translated their ideas across different mediums for discussion.

The driving question of the MoveLab centered the designs of the dances on an issue that the participants cared about. By having them focus on an issue it, provided some constraints to exactly what they might choose, but still gave them room to make it personally meaningful as they could bring in a range of topics. The participants engaged in a participatory design activity of creating issue-expression story board. The story board served as a format that directed the students to identify an issue, describe their feelings surrounding that issue, identify how they might express their feelings, and create a final stage that could serve as a conclusion of a dance. The leader of the exercise began by sketching out an issue that was important to her, a recent controversy on social media, where females who write about video games were attacked and threatened by individuals who identified themselves as part of Gamergate. This introduction to the exercise served

as a format in which the students were given a fleshed-out example of the exercise, as the instructor walked them through each section demonstrating how they could flexibly apply many ideas into their storyboard. Having the educator walk through an issue that she cared about within a format, enabled her to serve as a cognitive model (Collins, Brown, & Newman, 1989) as she described her depth of thought within the example. After the students completed their storyboard, they presented their ideas to the group. The participants' ideas ranged from inequality between wealthy and poor schools, bullies, fights at school, environmental issues, animal welfare, and the homeless. Figure 2 is an example of one of the participants who created her diagram about fake friends.

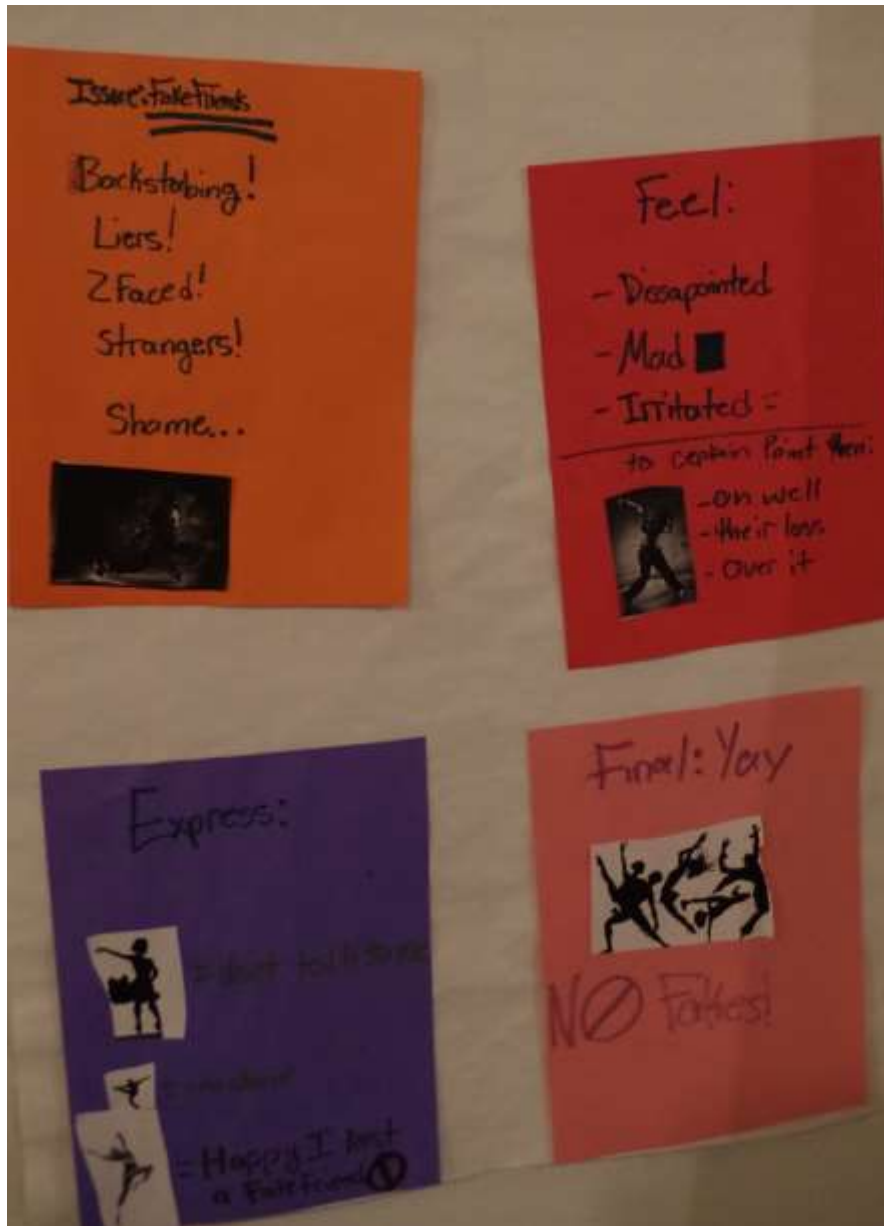


Figure 2 – Issue Diagram of *Fake Friends* completed by one of the participants in the MoveLab. The participant identified this issue as *backstabbing, liars, 2-faced, strangers, and shame*. This made them feel: *disappointed, mad, and irritated to a certain point* and then they felt: *oh well, their loss, over it*. They wanted to express this with representations of: *don't talk to me, I'm done, happy I lost a fake friend*. Their final stage was a celebration: *yay, no fakes!*

Without prompting, participants began to make connections between their ideas and several emerged as the issues they wanted to address. After the participants learned about the electronics, dance and choreographic techniques, we prompted them to translate their ideas from their formats into components they could incorporate into a dance. We provided a dance planning worksheet that served as a protocol for communicating with the leaders and a way to organize the participants' thoughts and designs by requiring them to identify what they wanted to represent within their dance and the various ways they might attempt to do this (Figure 3). From this we were able to find similarities between the participants' ideas and grouped them appropriately.

Name: _____

Think back to the activities and things you learned yesterday to answer these questions. Your answers to these questions will help us understand your vision for the dance performance!

1. What do you want to represent in the dance?

What emotions, senses or ideas do you want to convey in the dance?

2. What movements can you use to represent it?

What dance moves, phrases or styles do you want to use in the dance?

3. What do you want to use in the dance?

What sensors, circuits, or outputs do you want to use in the dance?

Figure 3 – Dance Planning Worksheet that guided students to organize components. This document was also used to break students into groups with similar themes.

At the end of the workshop we wanted a platform for the participants to be able to continue this dialogue around their values with the surrounding community. We therefore setup the final meeting to be an open house dance performance at EyeDrum, which would provide an authentic venue for the participants to share their ideas, reflections, and hard work with a broader audience. After the dances, the participants engaged in a talk-back

with the audience where they were able to give an overview of the ideas behind their dances and answer any of the audience's questions.

3.2.3.3 Infrastructure for Organizing Work

Within the MoveLab there was infrastructure specifically designed for facilitating group work and communicating progress with the leaders. When participants were initially placed in groups they needed to formulate their ideas together as a group and reconcile differences between them. The Dance Grid activity prompted the participants to decide on the aspects of their design and reach consensus on the visual, auditory, and technical details (see Figure 4). The grid served as a protocol for the participants to communicate with one another and the leaders about what was going to be in their dances.

Dance Grid—Theme:	
<i>Sensors:</i>	<i>Dance Dynamics:</i>
<i>Choreographic Tools:</i>	<i>Music Choices:</i>

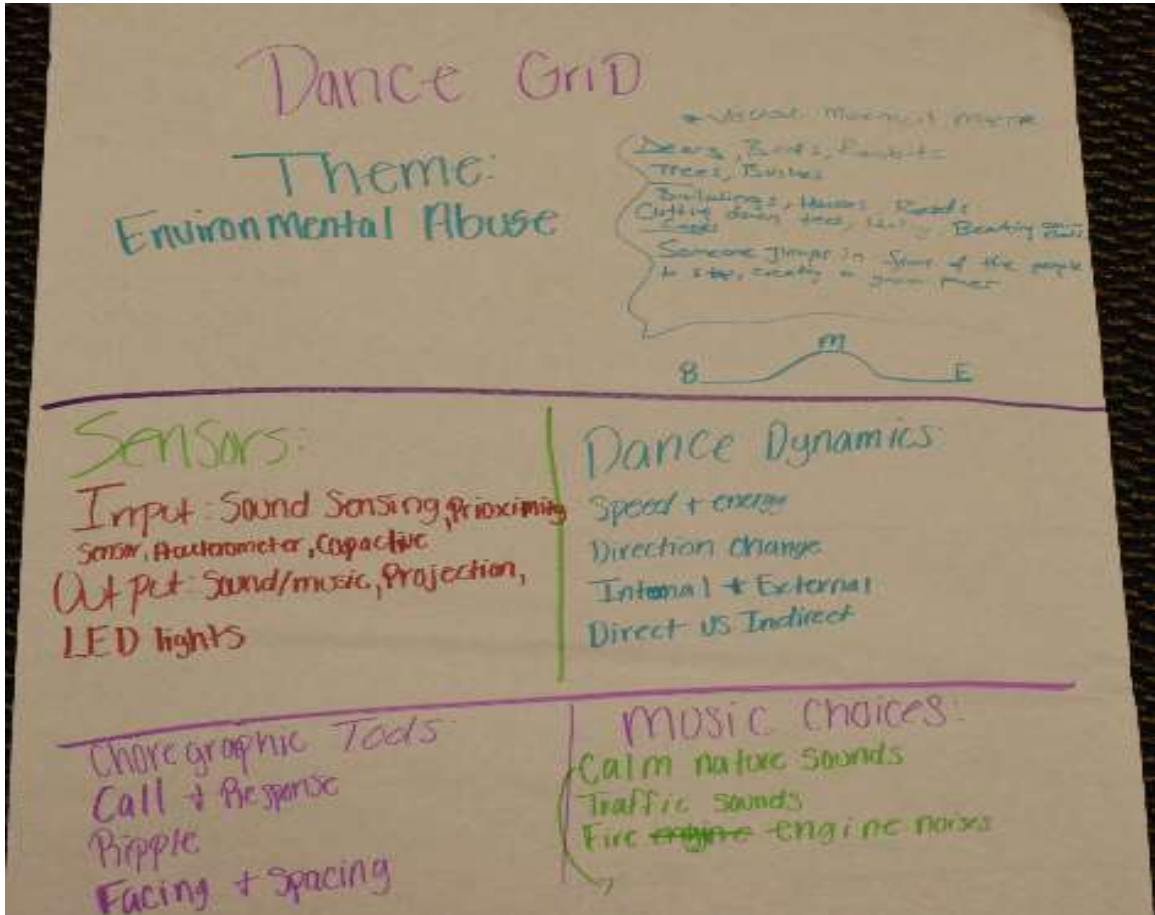


Figure 4 – Dance Grid protocol for organizing ideas and planning the dances: (top) template given, and (bottom) example of group use.

We also, had the students complete a *To Do Grid* where students wrote down what needed to be completed for the circuitry, code, dance/choreography, music, costumes, and practice. The themes the participants decided upon, guided what they needed to learn and work with to develop the technology, dance moves and choreography for their dances. Articulating this information helped the students organize what needed to be completed as well as facilitated identification of where they needed help. Figure 5 shows the template used and an example from the workshop.

To Do:	
<i>Circuits:</i>	<i>Code:</i>
<i>Dance/Choreography:</i>	<i>Music:</i>
<i>Costumes:</i>	<i>Practice:</i>

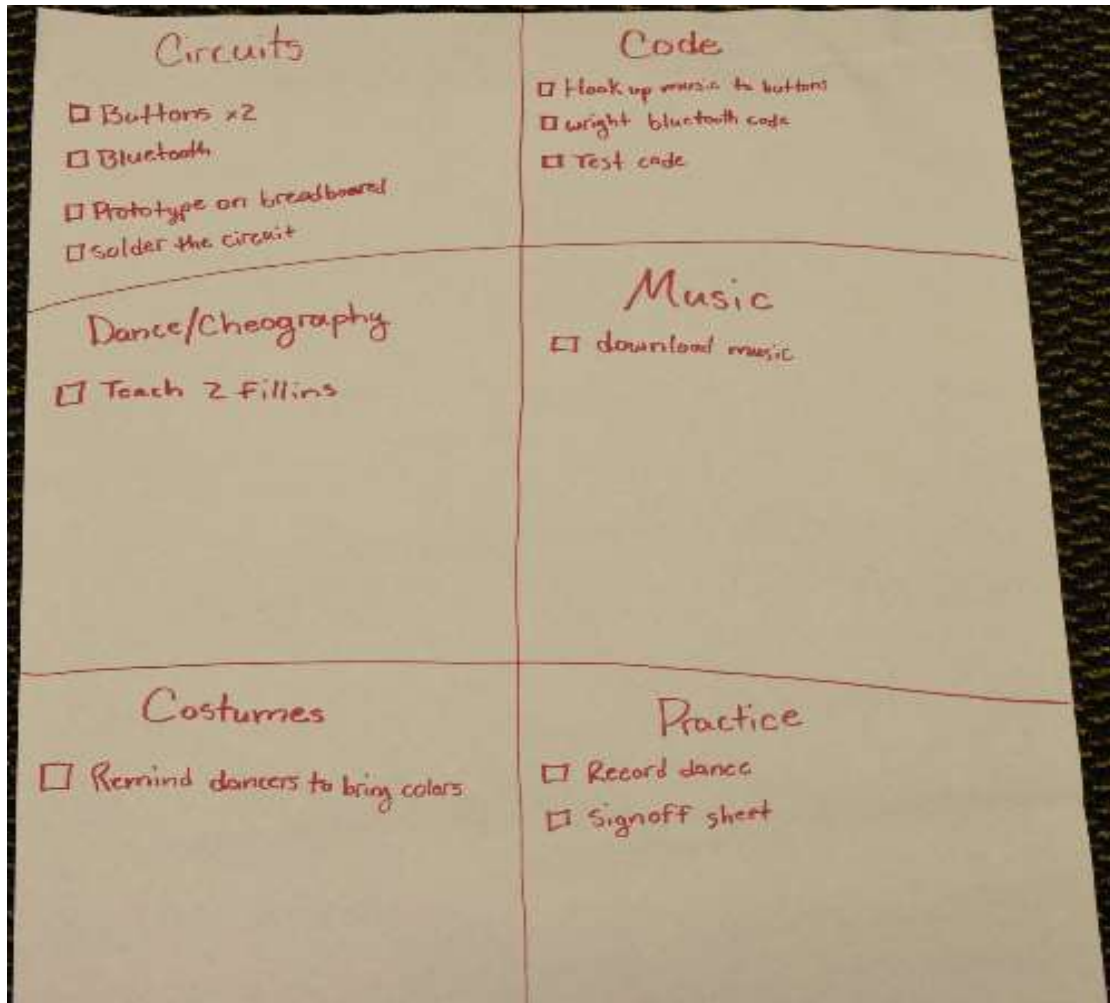


Figure 5 – Organization Grid protocol to facilitate outline of work: (top) template given, and (bottom) example of group use.

3.2.3.4 Dances Resulting from Infrastructure

The MoveLab activities led to the emergence of three groups. The first group created a dance about bullying and a girl learning to stand up for herself. The group used LED lights, capacitive touch sensors, and a piezo buzzer to express the emotion and

conflict around bullying (see Figure 6). The second group created their dance about the harms of deforestation, expressing interactions between humans and nature. The group used green and red LED strips to represent differences between nature and humans, and capacitive sensors triggered a wood-chopping sound. The third group created a dance about resisting peer pressure and standing up for oneself. The group attached LED strips to a cup to symbolize negative peer-pressure to drink. All three groups used a push button to control their music during the dances.



Figure 6 – One of the costume tops for the bullying-themed dance, which included: a) two capacitive touch sensors, b) a piezo buzzer, and c) an RGB LED strip

3.2.4 Findings and Discussion

Through our analysis we identified characteristics within the learning environment that influenced the students' transformation of their self-concepts in relation to computing and dance and identified how the students' dances were serving as boundary objects throughout the workshop. We discuss both of these findings in the following sections after

a short analysis of the quantitative data we collected, which will give some context to the participants in the study.

3.2.4.1 Perspectives of Computing and Dance

The analysis of the survey data has a small sample size so we are using it simply as another data source to triangulate with the qualitative data we collected to improve the *credibility* of our study (Lincoln & Guba, 1985). The survey asked about computer comfort, skill, and support in the pre-study survey to understand the girls’ technical familiarity and identify a correlation between technical familiarity and aptitude for computer science and engineering concepts. Overall, the majority of girls identified themselves as being comfortable, skilled, and supported with the computer. Of the 10 collected responses, 6 reported that feeling very skilled at using the computer and comfortable using the computer by themselves; 2 responded that the statement was somewhat untrue and only 1 responded they did not feel skilled at using the computer. Seven responded that they identified with feeling supported when running into a problem on the computer. However, these technical competencies did not translate to identifying with computer science.

Table 4 – Perceived Comfort and Aptitude on the Computer

<i>Statement</i>	<i>Very Untrue</i>	<i>Untrue</i>	<i>Somewhat Untrue</i>	<i>Somewhat True</i>	<i>True</i>	<i>Very True</i>
<i>I feel comfortable using the computer by myself</i>	0	0	2	1	1	6
<i>I feel like I have support when I run into a problem on the computer</i>	0	1	1	1	3	4
<i>I feel like I am skilled at using the computer</i>	0	1	2	0	1	6

We asked the girls to identify themselves with respect to various roles and interests surrounding the dancer, artist, engineer, and computer science spectra. In the pre-study surveys, the highest concentration of negative or “very untrue” responses occurred in questions about technical identifiers (I am an engineer, 4 responses; I am a computer scientist or programmer, 6 responses) and technical interests (I enjoy programming, 3 responses). The highest concentration of positive “very true” responses occurred in questions about interests that could be viewed as hobbies (I enjoy making things, 7 responses; I enjoy designing things, 6 responses; I enjoy building things, 4 responses). Pre-study data shows a negatively skewed response for self-identifying questions related to technology occupations, and a positively skewed responses for interest questions. While the girls didn’t know or identify with the technology centric occupations or hobbies, they did express strong connections to interests that lend themselves to engineering and computer science concepts (e.g., making, designing, and building).

We only were able to collect paired pre- and post-surveys from 6 participants because of the fluctuation in participation. When we look at the trends in the deltas between the six pairs of pre- and post-surveys (see Table 5), it shows that there is an overall trend of the data points moving up the scale towards a more positive rating of the occupations and hobbies. The workshop seemed to solidify what students already enjoyed (ie. building things and making things) and showed increased perceived enjoyment in programming and dance. However, it is interesting to note that while students did enjoy programming, they still were far from seeing themselves as a computer scientist, computer programmer, or engineer.

Table 5 – Pre- and Post-Study Survey Data Focused on Participant Interest and Identification

<i>Statement</i>		Very Untrue	Untrue	Somewhat Untrue	Somewhat True	True	Very True
<i>I am a dancer</i>	Pre	1	2	0	2	0	1
	Post	0	1	1	2	0	2
<i>I am a choreographer</i>	Pre	1	2	1	2	0	0
	Post	1	1	0	3	0	1
<i>I am an engineer</i>	Pre	3	0	2	0	1	0
	Post	1	0	3	1	0	1
<i>I am an artist</i>	Pre	0	0	1	1	0	4
	Post	0	0	2	0	1	3
<i>I am a computer scientists or computer programmer</i>	Pre	5	0	0	0	0	1
	Post	1	3	0	1	0	1
<i>I enjoy dancing</i>	Pre	1	0	2	0	0	3
	Post	0	0	1	2	1	2
<i>I enjoy programming</i>	Pre	2	1	1	1	0	1
	Post	0	0	0	0	3	3
<i>I enjoy building things</i>	Pre	0	0	0	2	1	3
	Post	0	0	0	0	2	4
<i>I enjoy making things</i>	Pre	0	0	0	2	1	3
	Post	0	0	0	0	2	4

3.2.4.2 Experiences Impacting Students’ Self-Concept

Throughout the workshop we saw transformations of the students’ self-concepts. The transformation was not a straightforward evolution, but more of a negotiation between the students’ changing views of dance and technology, in conjunction with their understanding of themselves. As their perspectives evolved over the course of the project, they navigated the incorporation of new aspects of these disciplines into their self-concepts. We found instances in which there was congruence formed between their self-beliefs and the disciplines, as well as instances in which there was dissension formed between their self-beliefs and the disciplines.

Developing Congruence

We identified three characteristics within the MoveLab that enabled students to form congruence between their self-concepts and the computing and dance material: (1) creating multiple roles for participation, (2) fostering a socially supportive community, and (3) integrating student values within the themes.

Multiple Roles for Participation. The open-ended project nature of the MoveLab created many roles for the students to get involved with both dance and computing. Many students took on specific roles in their groups such as the role of circuit builder or choreographer. These roles were important for creating agency and facilitating the leaders in guiding the students. Bandura showed that agency could be viewed as an artifact of perceived self-efficacy, which in turn often positively aligned one's perception of themselves within a discipline (Bandura, Barbaranelli, Caprara, & Pastorelli, 2001). The agency the students assumed within the MoveLab enabled them to take on more tasks and engage more deeply with the content in ways that were beneficial to the students and others in the community: *"I was actually kind of surprised because I'm not usually that great...with technology or any of [the] elements surrounding technology. Once I caught onto the computer programming and hooking up different wires...I would help the other people in my group to...catch on with it as well and hook up different sensors."* — Janelle (student).

Like Janelle, we found that by encouraging students to form a narrow expertise, we increased their ability to develop a deeper understanding of a topic rather than a breadth of superficial knowledge. These roles served as a guide for the leaders to understand what

piqued the interest of each of the students and to understand where each of them was gaining expertise, so the leaders could then push the students in the directions necessary to engage more deeply with the topics. This deeper engagement created knowledge disparities between the students, allowing them to serve as technical resources for others. This is a key characteristic within communities of learners, which allows for integration and thus appreciation of various types of knowledge in the community (B. Rogoff, 1994; Barbara Rogoff, 2001). Even the programming aspects had tangible results once uploaded to the microcontrollers. This externalization provided a way for students to gain recognition as others acknowledged and appreciated their contributions.

Supportive Community. The supportive community developed in the environment allowed students to take more risks and not be as afraid of failure. For example, one of the students reflected on how the environment enabled her to try something new: “*Since I’m so used to doing something strict and technical like ballet, it was a bit out of my comfort zone because I’m not used to just doing freeform whatever...But, um, since it was just the girls I guess it was fine.*” — *Janelle (student)*. Creating a place where students could interact in new ways with material they originally felt uncomfortable with led many of the students to re-evaluate their initial rejection of the material. One student originally only wanted to participate with the technology but found herself also contributing to the choreography: “*I was actually proud of myself for this. I made up a dance move. I was like ‘yay!’ because I never make up dance moves.*” — *April (student)*. The importance of these experiences ties back to the hierarchical nature of one’s self-concept, in which the student’s evaluation of herself during specific experiences shapes her higher order beliefs of how she views herself in terms of certain disciplines (Shavelson et al., 1976).

Values-Based Learning. We created an initial design activity to enable the students to have control over their performance themes to express values relevant to them. This structured design activity enabled students to talk about issues they were grappling with and helped them represent those issues through dance and technology. Interviews demonstrated that this was an important aspect of the educational experience that differentiated it from other experiences they have had: *“You’re still learning but it’s more fun and it’s not all about learning, it’s about helping with life and stuff like that.”* — Malia (student). By allowing the students to choose their themes, they began to see how the concepts they learned could be integrated into topics they cared about. When April was asked what her favorite thing about the dance was, she stated: *“My favorite thing is the message behind it...Because we’re basically saying that you should not put someone down and that the person that is being put down can, you know, stand up and not take the bullying or the hurt.”* — April (student). The students thought of their work as relevant outside of just the workshop, placing it in the context of the *real world* and other things they cared about. Furthermore, it was clear this characteristic was important to the participants and differentiated the MoveLab from their other learning experiences.

Developing Dissention

Throughout the workshop, students also developed dissension between their perspective of dance and technology and their self-concept. We identified three characteristics that cultivated this dissension: (1) their previous perspectives of the disciplines (2) fear of failure, and (3) working in a group environment.

Student Perspectives of Disciplines. The students entered the workshop having already built some ideas around dance and technology. As the leaders worked with the students many of these conceptions were revealed, and they often came out in ways to show how students previously disengaged with the material. Some of their perspectives of the material centered on how they saw the disciplines situated in society. This led to views about who should be participating in certain forms of dance: “*one girl said that...poor kids don't get to go to good schools. And she had drawn a picture of a girl doing ballet that was happy and [at] the rich school...and then a picture of the sad poor girl doing hip-hop*” — Whitney (dance leader). Many contrasted their perspectives of the computing and engineering material in this workshop to their previous experiences in school: “*I learned [in the MoveLab] that science doesn't always have to be in the book. It can also be fun. Or you could like use it in life, in real world experiences.*” — Maila (student).

Some of the students already had strong identities around dance and/or technology: “*I'm not really much of a dancer, but I'm definitely going to be doing more technology.*” — April (student). Our workshop was not the first exposure these students had with either dance or technology. Through the media, school, and informal educational programs, the students had many beliefs and ideas about the topics we presented to them. We observed how these previous experiences seemed to cause them to disengage with the material, making them reluctant to participate. Bringing these beliefs to light enabled our leaders to work with and around perspectives that were creating dissension between the students' self-concepts and the material. When a student's values were conflicted, it was important to be aware and sensitive to them, as it was imperative that activities are done with a respect for students. Research supports the usefulness of revealing students misconceptions about

material at the beginning of the learning process (Hestenes, Wells, & Swackhamer, 1992). Our study suggests it might also be useful to understand what students' perspectives are of a discipline and how it relates with their values. By doing this we can get a better understanding of how students view themselves in terms of what participation in these disciplines represents.

Fear of Failure. Throughout the workshop, we saw students attempting to do things with either dance or technology and struggling with their self-esteem. If they got to a point in which they were unhappy with themselves, it would often lead to the students disengaging with a particular activity: “*She tried really hard and I think she was doing well, but I think the self-judgment...or whatever her expectation, was not meeting with whatever we were doing. So that put her down and she kept pulling herself down each time*” — *Aiza (dance leader)*. When this happened, it would make it more difficult to get the participants to work with the material in future instances because they did not want to be put in that situation again.

Participants also avoided the material even before they experienced failure. For example, Harmony was outgoing and vocal throughout the workshop. She was consistently demonstrating her knowledge in the community through showing off things she had previously made and talking about other experiences she had with computing. However, during the daily leader debriefs, many expressed how Harmony would shut down whenever she was asked something she was unsure about or had to think through. At one point, when one of the leaders asked if she wanted to work on the code for her dance, she physically ran away from the computer toward the costuming table saying, “*I’m not a technology person.*” Harmony participated in ways that guided her away from engaging with the

technology. While an interest-driven perspective might presume that Harmony is *just not interested* in technology, the socio-cultural aspects of the situation paint a different picture. Her prior experiences with creating stop-animation on a tablet and designing a computer game in a workshop indicated that she did in fact have interests involving technology. However, these interests were seated in the value she placed on being able to show off her creations and gain social recognition. By presenting her with a difficult task where she was unsure of her ability to succeed, her values became at odds with participating with the technology. Harmony navigated this contradiction through participating in other activities, like the costume design.

The idea of failure also impacted how participants viewed the final performance. The workshop was five days and the students were novices in both dance and computing leading to a performance that could still be improved in many ways. Some participants expressed fear that the final performance would be a failure, and some had parents who publicly expressed concern that the performance was not of a high enough quality for their child to participate in. A few of these participants did not show up to the final performance and we believe it was because they were navigating around this fear of failure. This type of concern for failure is of particular concern with authentic artistic engagement. Because this was a real performance, at a real art venue, it set an expectation that the quality of the performance needed to be high. However, because we were working with novices and we had limited time, we sacrificed production quality for the participants' leadership that allowed their ideas to drive the project. By placing dance as central to our endeavor, rather than just a "*hook*", we unwittingly created a higher bar to reach for authenticity. This raised

expectations, and therefore created a greater opportunity for public failure that worked as a barrier to participation.

As we create these experiences for students we need to understand how to help them deal with failure in a way that is healthy and productive, because failure is an important aspect of the creation process for both dance and computing. At the same time, we need to understand how to prepare students such that they can feel proud of their final artistic expression. Extending the time to work on the dance and creating a space for them to show off their work in ways they are comfortable, are important considerations.

Group Collaboration. While there were many benefits to working in a group, we also found that the participants rejected working with or in front of other participants. One of the girls was having difficulties at first working with others: *“So the first day, I thought she is not going to work, like teamwork, so I tried to just talk to her and help her understand that it’s a team [activity] and we could make mistakes and we could also correct them.”* — *Aiza (dance leader)*. This type of group resistance was much more common at the beginning of the workshop and subsided as the girls became accustomed to the expectations, each other, and the dance and technology material.

Within the learning environment, not all participants felt comfortable at all times with sharing their work with one another. In one of the initial dance activities, where the participants were creating dance sequences for their name, a couple participants did not want to share the sequence they came up with. At the end of the activity, the participants and leaders sat in a circle on the floor, and the dancers led a discussion about working with your body. Several participants discussed the concerns they had with their body image

which made it more difficult for them to want to move and dance. The leaders were often able to understand and work with the difficulties the participants were having, such as working with them 1:1 when the other participants were doing other tasks. By taking it out of the social context leaders avoided publicly embarrassing the students or making them feel any additional discomfort. There are a variety of reasons that students might be resistant to participation in a group setting, so it is essential that these moments be handled in ways that do not marginalize the student or their choice to not participate.

Other instances where participants ran into difficulties with group work, were in instances where they were not able to navigate between the different values they wanted to integrate into their dances. In the next section, we examine these types of navigations through the construct of boundary objects.

3.2.4.3 Boundary Objects: Collaboration in the Face of Differences

In conjunction with the codes around self-concept our data also revealed the presence of boundary objects. As the students worked on their dances throughout the workshop we realized how the dances were serving as boundary objects that enabled the students to integrate their values into the experience even in the context of group work where participants' values and ideas may differ. The boundary objects supported collaboration, enabled students and leaders to learn about each other and from one another, and created opportunities for the participants to take on different roles. Within this section, I report on various instances which are indicative of the themes we saw in the data where the dances and computational objects arose as boundary objects.

When the groups were first formed they worked in their groups to brainstorm. The bullying group was comprised of two girls who wanted to work on bullying and one girl who wanted to represent a dictatorship. After each of the girls talked about their ideas, the girl who wanted to represent a dictatorship was able to find common ground. She remarked on how their ideas were not that different because each of their scenarios went from a bad situation to a good situation, and the expression would be similar. This realization enabled them to get over the first bump in their collaboration and move forward discussing how they would use the dance and technology to express the bad situation and the good situation in their dance. Even before the dances and technology were realized, we witnessed how the dances acted as a boundary objects through the students' negotiation of themes, helping them bridge their differences. While each of the girls had a robust definition of the meaning of their dance, they were able to talk about it in an abstract way when they were working together across their perspectives.

Having the students express their differences created an environment where the students could recognize that art and technology can be interpreted in many different ways. During the post-interviews one student remarked, "*I enjoyed that we might like different things about what we do*". They also recognized these differences throughout the workshop as they caused tensions as the varying opinions emerged:

At certain points in the dance, um, there were certain things that not everybody agreed on, so we had to change a few things in the dance. It ended up working out but it was just, um, a bit hard to agree on certain movements in the dance at certain points.

The need to collaborate on producing the dance brought contrasting views to light giving students opportunities to negotiate between their differences. However, they did not necessarily reach a consensus in order to move forward with the decisions they had to make. The students had to *coordinate* between the various differing perspectives. A student in the deforestation group reflected on how her group members thought about the dance:

One of my group members, her original theme was about bullying, and so [to] incorporate that into the nature theme she said that humans would bully nature so she took it that way. And then, so that was a difference. And the other group member, she thought a lot about the animals in nature and protecting the animals so that was her main concern.

It was through these differences in perspectives of the dance itself that students were able to *coordinate* despite their differing interests. These particular differences between students' views had an effect on what they wanted to incorporate and accomplish with the work they put into the dances. When one student was asked about how the other students contributed to her dance, she stated:

Some of our ideas were different so we had to figure out how to come together...Like, two of them were similar, but not really. So we had to figure out where to put it all into the dance...everybody did like one part inside another part.

The dances became a conglomeration of the various aspects that each student contributed, but by working together they were able to achieve cohesion. They were able to understand each other's perspectives, and in this instance, weave their own additions to the dance into the greater structure.

The dances were also a boundary object for the leaders who viewed the dances from their own perspective. One leader remarked on her interpretation,

They brought up...these global issues [that] also impact girls on a micro level and they impact them in their everyday life. And it's something that they do and want to build solutions for. They just need to have the know-how and education and resources to do it.

The leaders often saw the societal context in which the issues the students raised were situated. One dance leader saw the dance as a tool for empowering these students to talk about the issues they want solved in the world. Throughout the workshop we saw how the leaders' different backgrounds led them to approach the various aspects within the dance or technology differently. One of the dance leaders remarked on her experience understanding the technology through the students' interpretation of the technology:

I basically just sat back when you guys were doing your, uh, your lecture about the Arduinos and the breadboards and putting it together. And then I just circled around the kids when I felt like they had, like, kind of figured out what they were creating and then asked them. What did you create? What are you making? What does that do? So, I feel like it also helped me to know if they knew what they were talking about and it also taught me what they're talking about.

Here electronics portion of the dance was perceived differently: from the technology leaders' perspective, it was a teaching tool that uncovered how the students were thinking about the microcontroller; while from the dance leaders' perspective, it was a learning tool,

which the students could use to explain the concepts to them. The technology artifacts offered those coming in with different backgrounds different affordances.

Boundary objects enabled us to understand how the participants, with varying backgrounds, interests and expertise, could still create dance performances representative of their values. The collaborations within the MoveLab workshop were facilitated through the dances and computational artifacts that the students and leaders created together. These boundary objects were a form of artistic expression that enabled the participants to collaborate using abstractions while embedding different individual values. From learning about one another's backgrounds to learning about one another's expertise, the boundary objects served as learning tools for the entire community of learners. The boundary objects highlighted the various values and interests of the students and educators and helped to identify the learning opportunities between the diversity of participants. With further research, boundary objects could prove to be an important analytical framework to understand how fostering abstract thinking about boundary objects affects collaboration and integration of values in these interdisciplinary interventions.

3.3 The Day of the Dead Puppet Summer Camp

The Day of the Dead Puppet summer camp was an iterative study of values-based learning, which gave us an opportunity to understand if we could improve upon the infrastructure that was in the MoveLab from the perspective of values. The Day of the Dead Puppet camp was designed around creating interactive visual art rather than performance art, in which we found strong conflicts in the MoveLab. Furthermore, we incorporated individual projects to alleviate from social dynamics that could cause repression of values and

included protocols such as critiques and SCRUMs (stand up meetings) to ensure we could still support dialogue around values and culture in the absence of group projects. I will discuss the context of this study, the types of infrastructure that was integrated, and then do a cross analysis of both the summer camp and the MoveLab from the viewpoint of how values manifested in the environment in relation to the context and infrastructure.

3.3.1 Introduction and Context of Study

The Día de los Muertos or Day of the Dead Puppet activity involves participants creating an interactive art piece, which is representative of someone that has passed away in their lives. The participants are tasked with reflecting on people who have been important in their lives, cultural practices around death and mourning, and ways that they can express these complicated concepts with computing. The participants can appropriate this theme of loss in their own way; for example, to include loss of an object. The participants work individually to integrate microcontrollers to animate the puppets they have designed. The Día de los Muertos Puppet activity has demonstrated success across several populations of makers with varying expertise and experience levels (i.e. middle school students, high school students, graduate students, and adults). For the purposes of this dissertation I am going to describe the study of the activity being used in a summer camp run with high school students, since this was the workshop I was most involved with in terms of the planning, data collection, and analysis.

The summer camp was a total of six weeks, two of which were focused on the Day of the Dead Activity, the other four consisted of an Earsketch (online python based program for creating music with code (Freeman et al., 2014)) module and an aerospace engineering

module. The camp was run by a high school designed to help students who have difficulty in traditional public schools. They have both an avenue for students to receive their high school diploma and a path for students to get their GED. The summer camp consisted of 8 high school students, 7 of which participated in the study. Five of the students were in the school's diploma track, while three of the students came from outside of the school. The students ranged from advanced home school students (12yrs old), to students who were a grade or two behind (19yrs old). There were four instructors, inclusive of myself, another researcher, a PhD student, and a master's student who served as the consistent instructor across the course of the workshop. The camp lasted two weeks Mon.-Thurs. 10am-3pm. Students were consistent with their attendance throughout the entire workshop. On the final day of the camp, students presented their work to parents and other community members who were invited to come into the camp to see what they had made.

3.3.2 Data Collection and Analysis

Data collection consisted of pre-study surveys, observations on some of the days, two focus groups with four students in each, and post-study interviews with the other two student instructors. Pre-study surveys were administered to understand how students aligned themselves to engineering, CS, design, and art. Observation notes were recorded on the days the researchers were there and design artifacts were kept or documented for review. Focus groups were conducted involving 8 of the participants in a final summer camp. The focus groups involved talking about the camp generally, asking students about their experience, their puppets, their design processes, how they felt about their designs, and if it has changed how they view engineering, computing and design. The goal was to understand how values were integrated into both what the participants were building and

the learning environment as a whole. The initial codes used on the data were coded using the *Expression of Values* code developed by the researchers working on the MoveLab. I then created a reverse code to capture the instances where there was an observable or reported conflict, rejection of values in the environment, or a participant was inhibited in doing something they found valuable. The codes that I used are in Table 6. The data was then analyzed to understand what value domains might have been involved. We capture the findings from this analysis of values and value domains in section 3.4 .

Table 6 – High-Level Codes for Values

<i>Code: Description</i>	<i>Examples</i>
<i>Expressing and Integrating Values:</i> Student is able to articulate and integrate their values while participating in the learning environment	<i>“I was being considerate in thinking of the minds of blind people because blind people have problems too but they can’t see and vice versa for deaf people. They feel vibration but they can’t hear.”</i>
<i>Rejection of Environment or Inhibited Expression of Values:</i> Student is restricted in discussing or integrating their values or rejects participating in the environment	<i>One of the students wanted to broaden the scope of the activity, “I couldn’t really think of anything else to do. I didn’t really want to do that...I wanted to see and think if I could do anything else but that’s what I ended up having to do...I feel confined with the Day of the Dead”</i>

3.3.3 *Infrastructure in the Learning Environment*

Similar to the MoveLab Workshop the Day of the Dead camp incorporated infrastructure consisting of *components, design cases, formats and protocols*. The infrastructure again focused on: helping participants gain proficiency and knowledge; providing the structure for participants to self-reflect on their values; facilitating conversation around participants’ values; and organizing the participants’ work processes. I will discuss the infrastructure and then provide an overview of all of the projects.

3.3.3.1 Infrastructure for Building Proficiency and Knowledge

Similar to the MoveLab, the infrastructure in the Day of the Dead camp integrated a set of components and design cases for the participants to build knowledge with. The components and design cases were focused on learning about the Day of the Dead, reflecting on speculative design, and developing skills with the physical prototyping materials and electronic components. One of the instructors began the camp by introducing the cultural significance of Día de los Muertos, including the act of celebrating the dead, the meaning of symbolic colors and images, the purpose of puppets, altars, and other artifacts produced as part of the celebration. These various examples and representations served as *design cases* for participants to understand how the cultural symbols and designs could be applied to express emotions. To get the students reflecting on expressive design, one of the instructors also led a discussion on speculative design sharing *design cases* from professional artists who use design in a way that critiques or exemplifies certain issues or standards within our society (Dunne & Raby, 2013). For example, a video illustrating a speculative technology that monitors the calories of everything you consume and actually inhibiting you from eating over your allowable intake.

A set of electronic components and prototyping materials seeded the environment with resources for the participants to use. The SparkFun Arduino kit provided a set of electronic components and the researchers brought in a limited set of materials for prototyping (ex. foam core, foam block, clay, markers, 3D printer, etc.). Similar to in the MoveLab students were introduced to the technology through building out pre-determined *design cases* that could serve as applications for the students to reason about. The students went through three of the cases as a group: building a simple blinking LED circuit to

become familiar with the Arduino, building circuits with a button and potentiometer to control blinking and brightness of an LED to learn about digital and analog controls, and constructing a project called *rainbow fruit* which used fruit (strawberry, grape, and blueberry) as capacitive touch sensors to change the color of an RGB LED giving them a more complex design case. The students were then tasked with choosing a sensor as an input (ex. light sensitive resistor) to control an actuator as the output (ex. addressable LED strip) using code from the previous exercises.

Participants also constructed a slightly more open-ended prototyping design case in which participants brainstormed and prototyped an interactive emotional lamp using the foam core board and foam block prototyping materials. Everyone was able to experiment with both the foam core and foam block as they worked on the design activity. The participants worked in pairs and ended up creating lamps that represented emotions in two ways: some created lamps that were reactive and mimicked the mood or emotions of the user; for example, if the user was happy it would turn yellow or if they were sad it would turn blue; other pairs created lamps that were actively trying to change the mood of the user; for example, one pair design a book light that would understand the mood in the story and try and immerse the user through changing the light to an appropriate color.

3.3.3.2 Infrastructure for Reflecting on Values and Creating a Shared Dialogue

The Day of the Dead Puppet Camp had a driving question that was intended to integrate discussions around participants' values, and similar to the MoveLab, we engaged the participants in participatory design activities to engage them in reflecting on the driving question. The driving question used in this experience was a bit narrower than the MoveLab

and more explicitly called for a reflection on one's culture and how they respond to loss, death and mourning. The participants engaged in participatory design activities where they reflected on the loss of a person or object of their choosing first on a worksheet and then in the design of an interaction storyboard for their puppet.

The brainstorm worksheet placed the participants as the experts in understanding a death or loss they experienced and their own cultural response to death and loss. The worksheet was designed to support participants to be thoughtful in identifying components to use for the aesthetics and interactions that their puppet would support. The handout guided the participants in thinking about expression with 5 sections that gave them room to write and draw responses:

- 1. Who is someone important to you that has died?*
- 2. Why were they important to you? How did their death impact you?*
- 3. What is the story you want to tell about the person you are celebrating?*
- 4. Symbolic Representation - What is something the person you are celebrating did in the world that mattered to you? How can you represent that physically, visually, or with sound?*
- 5. What are the things that the person you are celebrating related to from the world around them? How can you use physical computing to symbolize this reaction with sound, touch, light, motion, data?*

The questions were structured such that they could be transferred to their puppet interaction storyboard. The storyboarding was a format that guided the design process, helping participants move from the conceptual ideas, to design of a puppet that incorporates programming and hardware to animate it. The storyboard involved four states: START, INPUT, OUTPUT, and FINAL. As the instructor walked them through the sections, students could grasp the significance of the various sections of the storyboard. Participants then sketched out their concept on craft paper using drawing material and collage materials,

such as images from Día de los Muertos celebrations, microcontrollers, sensors, and output devices. The use of the four states (START, INPUT, OUTPUT, and FINAL) encouraged students to think about their puppets as interactive objects and to better tie their conceptual ideas about symbolic triggers and representations into hardware, such as light, motion sensors, speakers, or lights. An example can be seen in Figure 7.

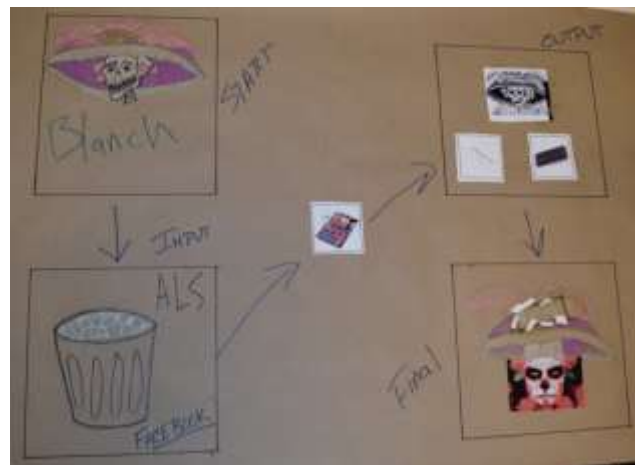


Figure 7 – Sketch of a puppet using START, INPUT, OUTPUT and FINAL indicating the computational communication with the microcontroller in the center.

We wanted to offer the participants a chance to share what they had created and have a chance to explain the reason for the specific design choices they incorporated. On the final day of the camp, the participants presented their puppets in an open house in which students were able to share their work with others in the community. The majority of people who came to the open house were people that the students knew (ie. other people involved with the school, their family members, friends, etc.). The open-house began with each of the students presenting their project and demonstrating how it worked. For the rest of the time students stood next to their project and explained to the community members who came in and out of the open house.

3.3.3.3 Infrastructure for Organizing Work

In the camp, we provided the participants with slightly different infrastructure than the MoveLab workshop in order to organize the work processes of the participants. At the end of each day in the camp they would have a stand-up SCRUM meeting which was used as a *protocol* for the students to talk about what they have done, what they are doing, and what their plans were for the next day. This gave them an organizational structure to ensure they all continued to have something to work on and were making progress. Different days had different goals, for example, in one of the SCRUM sessions the participants needed to talk about the materials they were using and what they still needed to figure out. The participants talked about the circuit elements they were using, what prototyping materials they were using, how they were planning to decorate each part, and out of that, what materials they still needed to get. On that day, the SCRUM session was moved up to the beginning of the workshop because the participants were given a budget of \$10 went on a field trip to a craft store to buy any remaining materials they needed.

Critiques were integrated throughout the workflow as a *protocol* to set a student-to-educator and student-to-student standard for communication. Throughout the design process, the instructors visited individuals as they worked, offering critiques. The idea was to create an open environment for constructive criticism while enabling the students to share and defend their design choices. During the process of working on the final designs, we organized an official critique session where the participants presented their work to the rest of the group. We scaffolded these interactions during this official critique sessions using a template (Figure 8) which gave students an idea of the type of feedback that could be useful in a critique session. These group critiques offered a chance for students to learn

and be inspired by each other, gain vocabulary for talking about technology and art, and reflect on their own ideas.

Name _____ Ask a question?
Name _____ Point out one thing that can be improved in the design.
Name _____ Point out one strength of the design.

Figure 8 – Prompts for Critique Exercise

3.3.3.4 Puppets Created from Infrastructure

In the summer camp, each participant designed and built their own puppet. By the time the participants reached the final day of the study they each had a working project and were able to successfully present their narrative to the community in the open house (two participants were absent for the final presentation because of another obligation). I will briefly present the seven projects to provide an understanding of the different types of puppets the participants created during the camp. The puppets had a variety of representations within them. Most of the puppets had an input (ex. button, flex sensor, etc.) and an output (ex. motor, LEDs, etc.), and some of the puppets were designed to have

multiple. On top of having different designs for the electronics, participants used different prototyping materials, which resulted in puppets that looked significantly different from one another. One participant molded a puppet of his dog out of clay (Figure 9). When the puppet was *pet*, the light sensitive resistors on the top of the puppet's head would trigger the NeoPixel (addressable LED) strip around the mouth and LEDs in the eyes to light up. The interaction was intended to demonstrate the happiness of the dog.

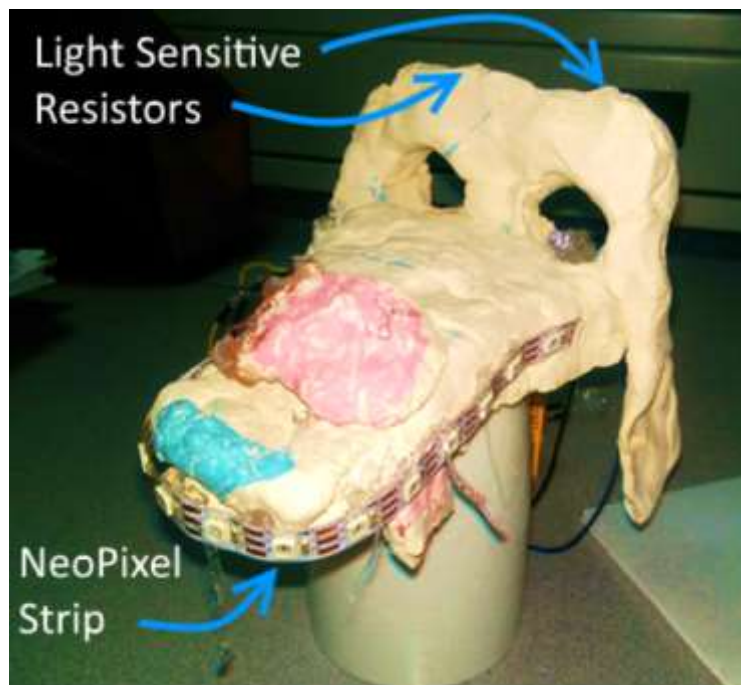


Figure 9 – Puppet of Dog with light sensitive resistor that would trigger NeoPixel strip along the mouth to light up signifying happiness

Some participants created their puppet about a person in their life that they had lost. Figure 10 is one example of this, where the participant created the puppet to represent his dad. His father passed away when he was young, but one of his distinct memories was of his father's grill (piece of jewelry for the teeth). The NeoPixel strip was programmed to light up blue and red when a button was pushed. These colors were chosen because they were the

participant's favorite colors. Within the computational piece, he merged the representation of his father with an artistic expression of his own. He made the coloring in the face similar to Harley Quinn's makeup because he said he *"likes weird things"* and he wanted it to *"look a little weird"*.

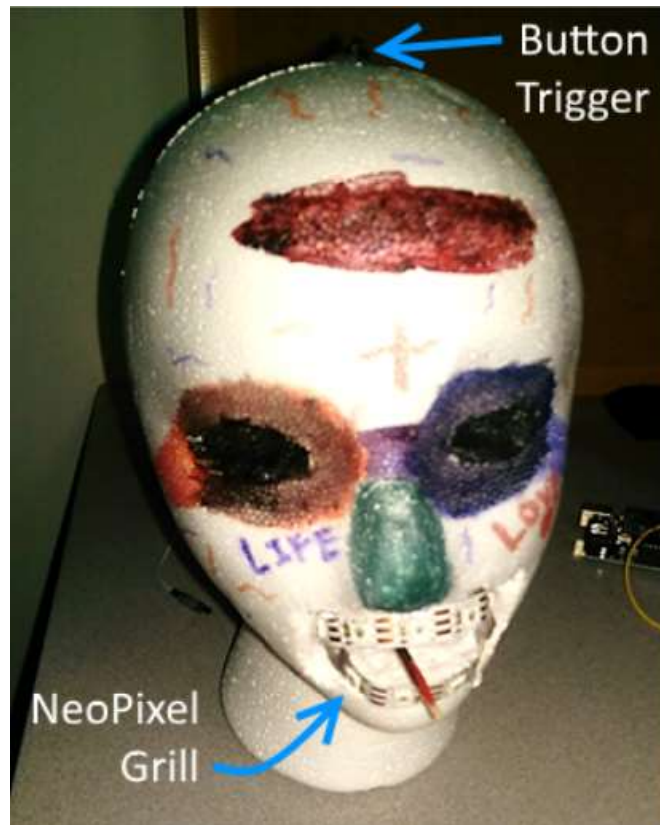


Figure 10 – Puppet of Father that had a NeoPixel strip triggered by a button that would light up red and blue to represent the participant's father's grill

Another participant represented her great aunt who had passed away. She wanted to represent her great aunt with the motorized wheelchair she remembered her sitting in. She placed the chair in a cloud that was representative of her aunt in heaven. She also integrated a representation of the necklace that her great aunt gave her and created the puppet below.

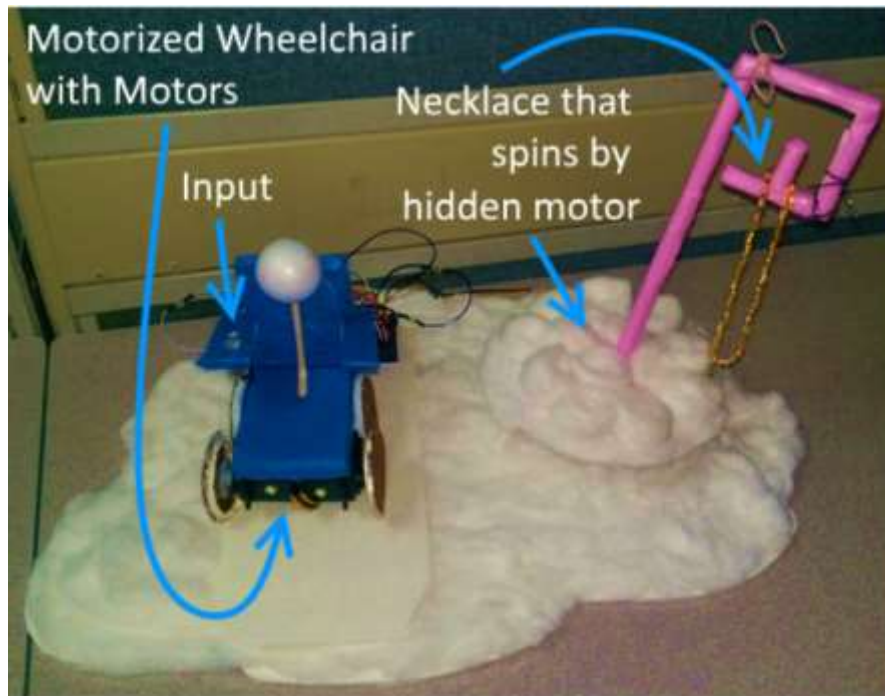


Figure 11 – Puppet of Great Aunt in heaven with motorized wheelchair. Input, located at the joystick of the wheelchair, activates motors in the wheels to move the chair back and forth and turns on the motor in the base of the necklace so that it spins.

She integrated technology in the motorized wheels of the wheel chair that would move the chair forward and backwards. She also incorporated an input sensor as the joystick in the wheel chair that would activate this movement as well as spinning the base that held the necklace. Another participant designed his puppet in a way that was representative of how the person died. He lost his aunt to a viral infection and used this as inspiration to create the puppet of the virus below in Figure 12.

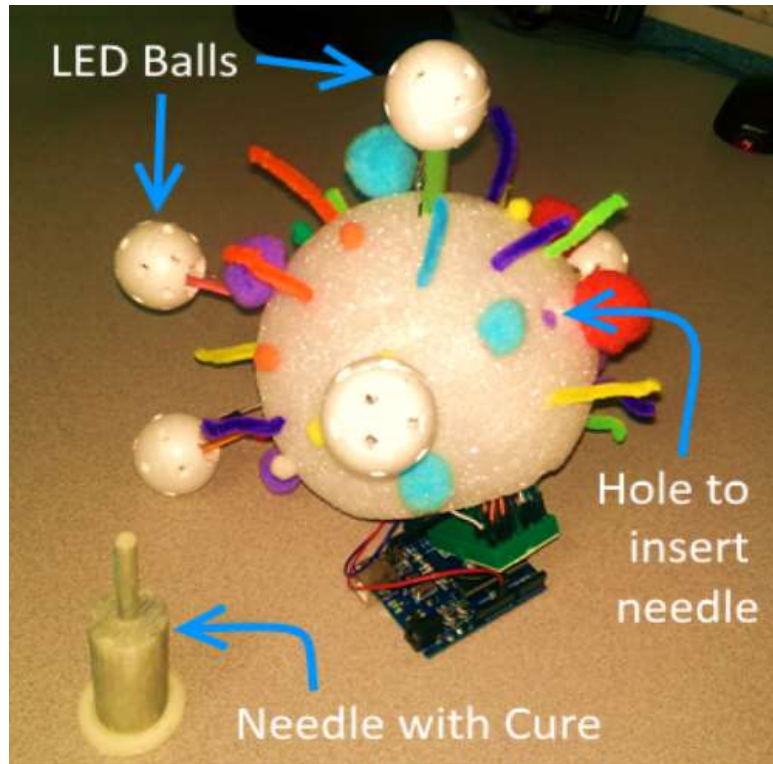


Figure 12 – Puppet of Virus had a button inside the foam body that when pushed with the needle would turn off the lights in the virus (ie LED balls) demonstrating killing the virus.

The puppet of the virus could be *killed* by injecting it with the medicine from the needle. Before being injected, the lights in the balls surrounding the main body would blink signifying it being alive. When the needle was inserted in the hole it would push a button which would kill the virus and stop the lights from blinking.

Some participants appropriated the theme of Day of the Dead to the loss of an object instead of a person. One participant had recently broken her tablet which made her angry because it is what she used to relax. She created an abstract representation of her feelings in the puppet while having it respond to a concrete representation of the tablet. Her design can be seen in Figure 13.

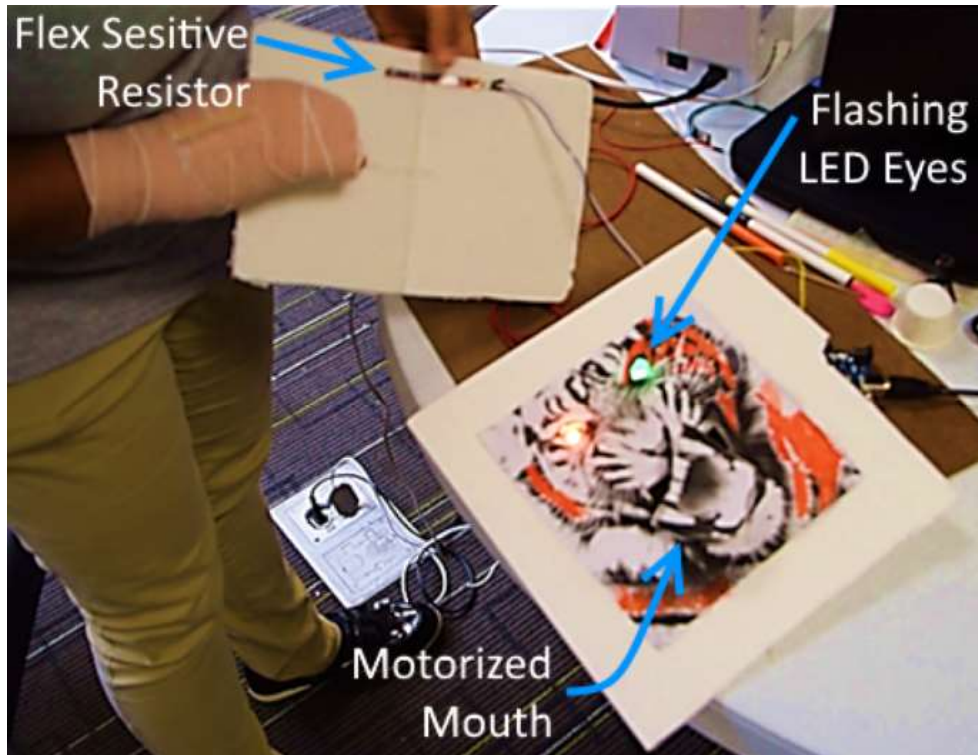


Figure 13 – Puppet of Tiger had a flex sensor attached to a fake tablet that would trigger the rage of the tiger by flashing the LED eyes green and orange and moving the mouth back and forth with a motor when bent signifying the breaking of the tablet

She describes her puppet in the quote below captured from her demoing it:

What this does, is that we have a flex thing right here [pointing to the flex sensor], so as you see it's not mad now [the eyes are green and the mouth is not moving], but people tend to actually bend their...tablet or any other device, so this basically shows like your rage of it, so when you bend it [proceeds to bend the foam core tablet], it goes like that [eyes start flashing orange and green and mouth moves back and forth].

Similarly, another participant represented the loss of technology by creating a phone charging pillow for when her phone dies. Describing what she built she says, “*I made an*

emoji pillow that detects when your phone is either dead or um fully charged...it tells you when to take it off the charger so you don't overheat your battery."

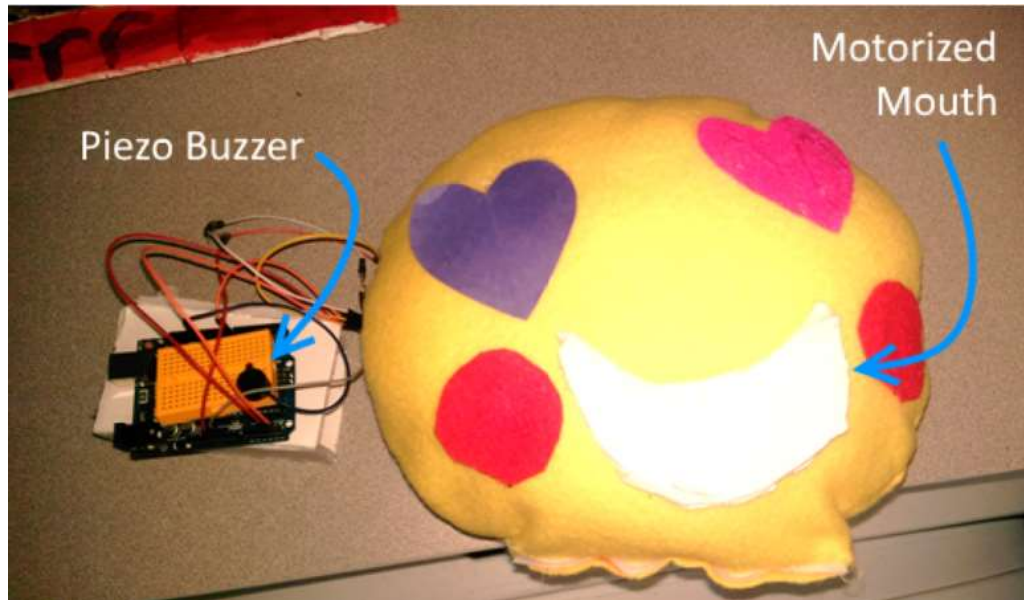


Figure 14 – Puppet of Emoji that gave you a pillow to rest on while you charged your phone. It indicates when your phone is charged by buzzing piezo buzzer and moving the mouth of the pillow with a motor. The input was actually a button.

We do not have a picture of the last participant's puppet, but he uniquely appropriated the theme of loss in his puppet. He was struggling to come up with a topic because, as he put it, *"everyone else did it about a person or a pet and I didn't have that person or a pet"*. He ended up reflecting on the fact that he missed participating in drama, which was an activity he was heavily involved in throughout the previous year. He explains the motivations for his puppet saying, *"so um for my day of the dead thing I used an experience about last time I went to a summer camp, so um one of the things I did was um when we were filming a video and um everyone was super hot and they weren't getting the video clear, so one of the things I did was bringing enough chairs and water and snacks because they were so tired."* He used a positive experience in which he was able to help

his peers in the summer camp as a way to reflect on something that he cared about bringing it into the computational design of his puppet. He describes what he ended up creating,

I had this chair and I had arm rests on it, and I had an um, we 3D printed a hand and I stuck it on a...took a big piece of foam core and then turned it into a make-shift arm and hot glued the hand on there, and then I attached it to a motor that would spin and I rested it on the arm rest. I wanted to make it so that um once it turned, it only turned up to 90 degrees...and I wanted to make it so that when it turned up to 90 degrees something would drop down to it, but um I didn't get to it.

The chair he created was representative of the moment he remembered helping out his friends in the camp. He put a hand on the chair to represent a *helping hand* and signify that the chair is there to help people.

These examples show how different participants were able to create a variety of puppets that represented loss in different ways. The puppets provided avenues for the participants to explore different sensors and actuators, as well as apply different prototyping materials. In the next section, I will present the findings from this workshop along-side a cross analysis of the MoveLab findings that examine the infrastructure in the learning environment through a lens focused on values.

3.4 Cross Analysis of Values in MoveLab and Day of the Dead

The MoveLab workshop and the Day of the Dead Puppet camp was designed to facilitate the participants in creating a personally meaningful experiences with physical computing. The original goal was to create an environment where participants would be

able to talk about their values and create an object that embodied aspects of their values. We became interested in what aspects of the learning environment were enabling participants to create personally meaningful experiences, and what aspects of the learning environment were not adequately supporting participants. We began to situate our findings from the perspective of values centered on analyzing the infrastructure in the environment. In order to keep our language consistent and clear when identifying and talking about values, we adapted use of Schwartz’s framework of value domains (Table 7 (Bardi & Schwartz, 2003)).

Table 7 – Schwartz et al.’s Value Domains replicated from (Bardi & Schwartz, 2003)

<i>Value Domain:</i> Definition of domain
Power: Social status and prestige, control or dominance over people and resources
Achievement: Personal success through demonstrating competence according to social standards
Hedonism: Pleasure and sensuous gratification for oneself
Stimulation: Excitement, novelty, and challenge in life
Self-direction: Independent thought and action-choosing, creating, exploring
Universalism: Understanding, appreciation, tolerance and protection of the welfare of all people and of nature
Benevolence: Preservation and enhancement of the welfare of people with whom one is in frequent personal contact
Tradition: Respect, commitment and acceptance of the customs and ideas that traditional culture or religion provide the self
Conformity: Restraint of actions, inclinations, and impulses likely to upset or harm others and violate social expectations or norms
Security: Safety, harmony and stability of society, of relationships, and of self

We used the value domains framework to code the projects of the participants and the data from the learning experiences, to understand how values manifested in the dialogue and the computing artifacts, and to draw insights from how it related to the infrastructure. We examined the infrastructure based on the goals of: reflecting on values, developing a shared

dialogue around values, building domain knowledge and proficiency, and organizing work processes around values. A summary of the infrastructure examined within the two environments can be found below in Table 8.

Table 8 – Infrastructure in the Learning Environments

<i>Infrastructure</i>	<i>MoveLab Workshop</i>	<i>Day of the Dead Camp</i>
<i>Building Knowledge and Proficiency</i>	<ul style="list-style-type: none"> • Dance Components: Dance dynamics and choreographic techniques • Electronic Components: Microcontroller and peripheral sensors and actuators • Constructed Electronic Design Cases: Building with example circuits with electronic components and code • Peer Teaching Protocol: sharing electronic design cases between participants • Constructed Dance Design Cases: Building phrases and sequences with dance dynamics and choreographic components • Professional Design Cases: Examples of professional artists and engineers working with dance, music and technology 	<ul style="list-style-type: none"> • Prototyping Components: foam core board, foam block, clay, brought in materials • Electronic Components: Microcontroller and peripheral sensors and actuators • Constructed Design Cases: Building example circuits with electronic components and code • Emotional Lamp Design-Case: Prototype an interactive lamp with foam block and foam core that represents an emotion • Cultural Design Cases: Day of the Dead example artifacts and Puppets • Professional Design Cases: Examples of professional artists' speculative designs
<i>Reflecting and Sharing Values</i>	<ul style="list-style-type: none"> • Driving Question Protocol: Design a dance that represents an issue you care about • Issue Diagram Format: highlight an issue, feelings associated with that issue, methods for expressing those feelings, and a final state • Dance Planning Worksheet: organizing issue into components for dances • Performance Protocol: Public performance to share work 	<ul style="list-style-type: none"> • Driving Question Protocol: Design an interactive puppet that represents someone or something you have lost • Puppet Planning Worksheet: to organize components • Interaction Diagram: highlight a starting state, an input, an output, and a final state of your puppet • Presentation Protocol: Open house to share work in a presentation
<i>Organizing Work Processes</i>	<ul style="list-style-type: none"> • Group Sharing: Sharing ideas between participants and leaders in group activities • Dance Grid Protocol: Identifying theme, electronic components, dance and choreographic components, and music • Organization Grid Protocol: Identifying what needed to be completed for the electronics, code, choreography, music, costumes, and practice 	<ul style="list-style-type: none"> • SCRUM Meeting Protocol: Share status and plans for design work • Peer Critiques: Participant and instructor critiques

3.4.1 Value Analysis of Infrastructure for Reflection on Values

3.4.1.1 Designs from Values-Based Learning

The participants in the MoveLab workshop and Day of the Dead Puppet summer camp were able to integrate values within the designs of their final projects in different ways. The MoveLab participants created dances that were coded with the various values outlined in Table 9. The interpretive nature of the dances led to the dances as boundary objects in which the participants did not have to agree on what values were being represented in order to collaborate. We coded for the values represented in those interpretations as well. However, it is important to note that we could have missed certain interpretations and therefore values that were not captured within the data gathered.

Table 9 – Dance Themes and Values Coded

<i>Dance Theme</i>	<i>Values Coded in Project</i>
<i>Deforestation of Nature</i>	Universalism – protecting nature Benevolence – (dance interpreted as bullying) protecting your peers
<i>Bullying and Standing Up for Oneself</i>	Security – physically and emotionally protecting yourself Security – (dance interpreted as overthrowing a dictatorship) safety of society and of the self Benevolence – physically and emotionally protecting your peers Universalism – helping others
<i>Building Self-Esteem and Standing Up for Oneself</i>	Security – physically and emotionally protecting yourself Benevolence – physically and emotionally protecting your peers

Overall, we saw a spread of different values that participants were integrating into their dances. The emphasis of certain values within their themes were sometimes driven by specific personal experiences. For example, the bullying group encompassed participants that had dealt with bullies in their school. Other participants chose certain themes because

of how it related to the people they cared about. For example, one participant was working with her sister in the self-esteem group and said that she was doing the dance to help her sister who struggles with low self-esteem. She wanted her sister to realize that she should be proud of who she is. The participants were in their adolescence and the *safety* and *security* of their social relationships was pertinent in many of their lives. We also saw instances of participants who reflected on issues that were outside of their interpersonal relationships. The participants who were motivated by the deforestation of nature theme representing values within the *universalism* domain, saw something in the world that they did not agree with and used their dance as a way to talk about it.

The Day of the Dead Puppets were also coded for values, as reported on in Table 10, which demonstrates some value domains that were similar and some that were different.

Table 10 – Puppets and Values Coded

<i>Puppet Theme</i>	<i>Values Coded in Project</i>
<i>Dog</i>	(none directly coded to puppet)
<i>Father</i>	Security – stability in his family relationships Tradition – being able to share his appreciation for his father
<i>Great Aunt</i>	Tradition – Christian representation of heaven
<i>Virus</i>	Security – protecting against harm of the virus Self-Direction – administering the cure for the virus yourself
<i>Tiger</i>	Universalism – helping others not break their tablet
<i>Emoji</i>	Universalism – helping others charge their phones Universalism – technology that supports all people
<i>Chair</i>	Benevolence – helping others

It is important to note that while the value domain might be similar, the underlying reason for it is not necessarily the same. For example, in the MoveLab we saw the prevalence of *security* domain based on the social relationships the participants were faced with; in the Day of the Dead Puppet camp, *security* manifested in terms of finding stability within one's family relations and *security* in terms of protecting one's health from infection. This emphasizes the use of the value domains for providing language to talk about types values, rather than for specifically identifying what people value.

One value that did not present itself directly in the dances in the MoveLab but was found within the puppets was the value of *tradition*. The problem question in the Day of the Dead puppet project prompted students create objects that provided the participants with a starting point to have a dialogue about differences and similarities in family, culture, and how technology can embody and represent these concepts. These ideas thus were integrated into the symbolism and discussions around the puppets. For example, the puppet the participant made of her aunt integrated the symbolic representation of a cloud in heaven with a cotton ball base that her puppet resided on. Another example, is of the participant who made his puppet in memory of his father who passed away. He discussed why this project was meaningful to him:

Just being able to share, to tell people about my dad, you know just about him. It's not all the time, or you know it's not every day that you get to make something you know that represents your dad. I think that's an awesome thing to do. That's a good way to show your appreciation

He was able to talk about his father and share the appreciation he felt for his father. The success of the student being able to tap into this value of tradition by showing respect to his father was directly correlated to the driving question since his father had passed away.

One participant was able to represent his values within his puppet because the puppet was a reflection of himself and the value he placed on *benevolence*. The participant who created the *helping-hand* chair conveyed the care and kindness he showed to his friends in the drama camp from the previous year. While there were various appropriations of the theme, not all the puppets were directly tied to specific values that we were able to identify. However, values did manifest in other ways participants worked on their puppets.

3.4.1.2 Results from Participatory Design Activities

In both the MoveLab and the Day of the Dead Puppet camp, the participatory design activities were generally successful in getting participants to identify, reflect on, and share their values with the other participants. Section 3.2.4.2 outlines some of the ways we saw values manifest in the participatory design activities in the MoveLab. For example, the issue diagram in Figure 2 outlining the issue of *fake friends* (from section 3.2.4.2) and the diagram in Figure 15 below outlining *abandoned homeless people, children, and animals* illustrate different values manifesting in the same infrastructure. The *fake friends* issue diagram (Figure 2) demonstrated how the student valued *security* in the trustworthiness of her friends to always be there for her. The diagram had her reflect on her cultural responses in which she illustrated two stages of expression: first it would make her *mad and irritated*, but she could only let it affect her to a *certain point* emotionally before she was *over it*.

The outlined scenario demonstrated the resilience that she felt was appropriate for dealing with the situation.



Figure 15 – Issue Diagram from a participant who chose the issue of: *abandoned homeless people, children, and animals*. They indicated this makes them feel: *empty, helpless, frustrated*. They wanted to express this using *Hip Hop and Wild dancing*. Their final stage represents a cycle: *It's just gonna keep being a cycle until I'm old enough to do something about it*.

The diagram in Figure 15, demonstrates the student reflecting on both *universalism* for being able to help the people and animals without homes, but also expressing her frustration with the lack of *power* she has as demonstrated by her final stage: *It's just gonna keep being a cycle until I'm old enough to do something about it*. Having the *power* to change the situation is something that she values because although she cannot currently solve the issue, she sees this as something she will have the ability to address once she is older.

The dance planning worksheet (ex. Figure 16) helped participants further articulate these issues and begin to associate the components that they had learned about to the expression of these issues. The student in Figure 16, was able to take contrasting components from the dance techniques thinking of both the speed and size of the movements that would represent both nature and the destruction of it. She discusses nature's movements as slow, pretty and small, while the destruction movements are fast, morbid, aggressive, and big. Furthermore, she connects her use of speed to an appropriate sensor indicating she will use the accelerometer. This demonstrates how she was able to integrate her values of *universalism* and not destroying the environment, into the design of her dance moves and the technology she chose.

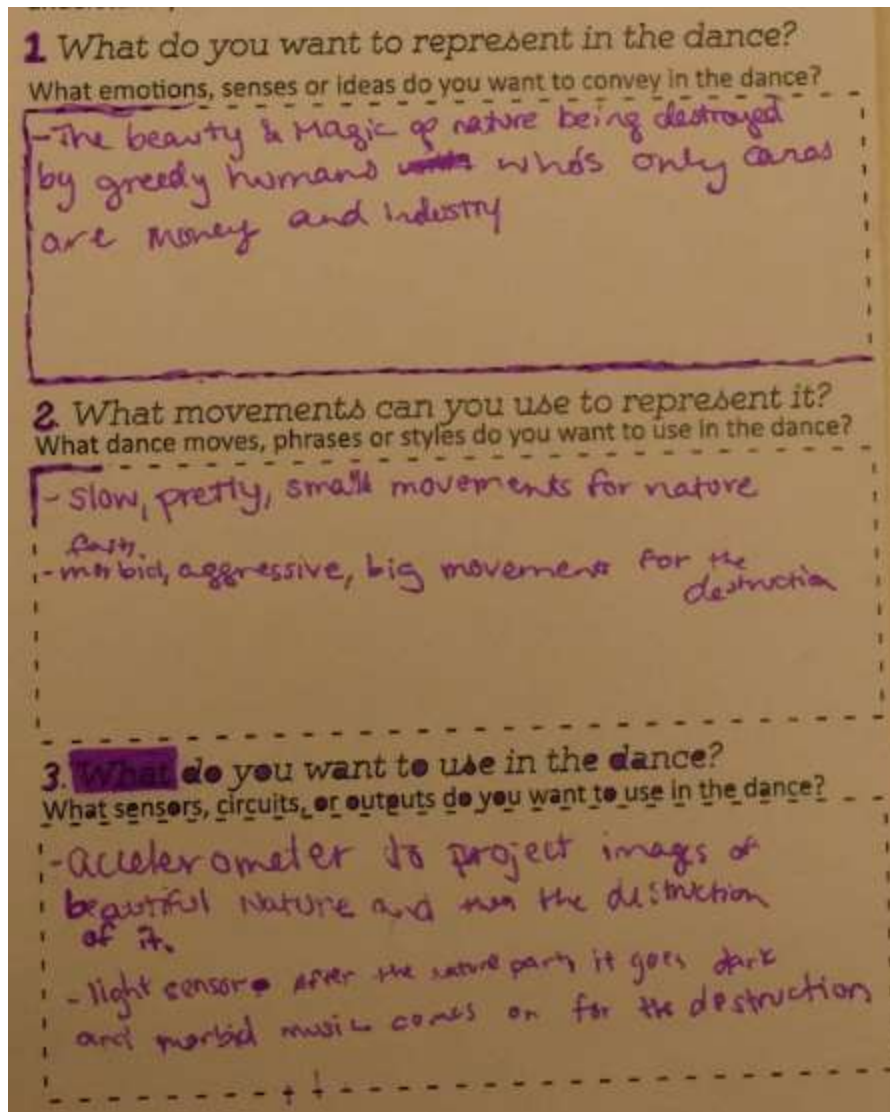


Figure 16 – Planning worksheet from student who wanted to represent the destruction of the environment in her dance. She wanted to represent: *The beauty & magic of nature being destroyed by greedy humans whose only cares are money and industry*. The movements she wanted to use were: *slow, pretty, small movements for nature and fast, morbid, aggressive, big movements for destruction*. In the dance she wanted to use: *an accelerometer to project images of beautiful nature and then the destruction of it. A light sensor after the nature part it goes dark and morbid music comes on for the destruction*.

While the majority of the students were able to begin to make these types of connections in this stage, not all participants were able to articulate their decisions on the same level as this participant. For example, one of the younger participants in the Bullying group simply

stated that she wanted to use LEDs that were red and blue for her technology. Upon later discussion, she talked about how the red represented sadness and the blue represented happiness. This was not as complex as the solution outlined in Figure 16 above; however, it still indicates that she was able to create links between her theme and how she was making choices about the technology. The ways students were making these connections indicated success for the high-level learning goal understanding the Arduino as a computing object that could collect data and output responses based on that data.

The participatory design activities in the Day of the Dead were also useful for the participants reflecting on what they valued. The activities led to participants appropriating the driving problem in creative ways that reflected how they wanted to interact with the material. One example, was the participant who took the theme and applied it to her phone instead of the loss of a person. She reflected on her choice when the focus group was discussing how they felt about using Day of the Dead, *“I chose my phone for the Day of the Dead so at the end I was happy. I didn’t feel like it was ‘I have to’ or like someone was forcing me to do it. I was like ‘I wanted this’ and I just did a pillow”*. Death and mourning are difficult topics to discuss even with adults. Some participants might not have wanted to or felt comfortable engaging with this topic and we tried to respect this. We attempted to make sure participants knew they could deviate from the themes in ways that made sense for them, and at the same time tried to ensure that it was a safe place for participants to talk about their emotions around this difficult topic. This same participant who designed the emoji pillow demonstrated how even when deviating from the theme, she was still integrating her values into the design during the participatory design activities. The interaction diagram format—START, INPUT, OUTPUT, and FINAL—forced her to

consider how someone might interact with her pillow, which she intended to use to communicate when the phone was done charging. She expressed her values of *universalism* through the accessibility choices she integrated into her pillow: *“I was being considerate in thinking of the minds of blind people because blind people have problems too but they can’t see and vice versa for deaf people. They feel vibration but they can’t hear.”* She wanted to incorporate lights, a vibration motor, and sound so that her design did not exclude anyone. These choices demonstrate her reflection on the trade-offs between various sensors.

The final showcase of the participants’ puppets in the open house gave the participants visibility to a community outside of the camp. The participants were proud of what they made, as one participant reflected, *“my favorite thing was molding and seeing my final product, and how it came out good”*. Another talked about the results of her hard work, *“I decorated and I did it with love and I put all my hard work in it and it turned out to be beautiful”*. When it came to sharing their work, they were not apprehensive, and their feelings of success were mirrored by the community members. In the focus group, the participant who created the emoji pillow proudly remarked, *“people said that I should market it”*. One of the leaders observed the reactions the participants received in the open house, *“Parents and other members of the incubator building showed up and they were all very impressed and excited. The students also felt proud showing off their projects.”* Furthermore, another leader noted his high regard for the participants’ proficiency in presenting and explaining their creations: *“All of the students were able to express a distinct well thought out narrative, and they were all really really good. It went really well and they largely hit it out of the park.”* These experiences are important for those who value

achievement in this way and the social prestige (*power*) related to other community members acknowledging the success in their work.

3.4.1.3 Valueless Designs and Value Rejections when Reflecting on Values

While we intended for the flexible nature of the participatory design activities to allow participants to bring their own values to the project, we found that the activities could still cause them to reject the values embedded within different aspects of the learning environment. In the MoveLab, we found some instances in which participants had reasons for not wanting to participate within a particular discipline because they had already decided from prior experiences that the discipline just did not fit with what they valued. While in some instances we understood the value choices participants were making, this was not always the case. As discussed in section 3.2.4.2, it would be useful to instantiate a protocol to understand value judgements participants might already have about technology, engineering, programming, art and design in order to gain a better understanding of how participants' previous encounters with dance and technology were shaping their perspectives.

Participants in the MoveLab also rejected the final performance which was designed to serve as a protocol for enabling them to share their dances with the broader community. For many participants, a public performance was something they were not excited about or comfortable with because they didn't want to *look stupid*. A public performance as a non-dancer or non-performer scared many of the students and created an environment where students were concerned about failure rather than feeling pride or self-efficacy over what they created conflicting with their value of *achievement* and social

power. The intention was to give the students a voice in the community to talk about the issues they faced while being proud of what they created, but their perspective of what they created and how they wanted to share it did not match with the infrastructure that was designed. Within the Day of the Dead this was rectified by the change in medium from the performing arts, to interactive puppets. The participants had to publicly present their projects, but they expressed more pride over what they created and were more comfortable with the public presentation as the community leaders coached them through it. The performing arts is not inherently going to conflict with all students' values, but it has implications surrounding participating and performing that need to be addressed within how the infrastructure is designed.

We also identified rejection of the values espoused in the puppet planning worksheet in the Day of the Dead Puppet camp. The worksheet contained skull images which was representative of the Mexican *tradition* and culture of the holiday. One participant interpreted these images as being anti-Christian or perhaps Satanic images. This participant's Christian identity and beliefs were so important to him he at first refused to participate. We removed the worksheet and asked him to answer the same questions without thinking about the images, as they were unimportant to the project. He began working on his design but was not fully engaged until having talked over his conflict with his parents. He reflected on this in the focus group:

Well at first, I was just trying to figure out if I was okay with it because when I think Day of the Dead I think of all these other religions and I'm not sure if I should do this so I had to talk about it with my parents, so I come out and I'm okay with it...I wish I could model it after something else

This participant was the one who reflected on the loss of his experience in drama camp. Another participant also felt the theme was a bit constraining. When asked what he would like to change about the workshop, he stated, “*broaden uh the creativity...I feel confined with the Day of the Dead*”. He struggled with coming up with something that he cared about within the driving problem. He created the puppet of his dog, and when asked why he chose what he did, he stated:

I couldn't really think of anything else to do. I didn't really want to do that...I wanted to see and think if I could do anything else but that's what I ended up having to do...I couldn't really think of...there wasn't anything else that I could think. It is like a creative spectrum and I couldn't really think of anything else that I could do...and that is what I thought of and was stuck with...I couldn't think of anything else so that is what I did

This participant was not able to use the driving problem for inspiration and it was clear that it the problem statement impeded his values within the *self-direction* domain because it was inhibiting his ability to creatively explore the design space. However, he was able to pull some inspiration from the other participants in the camp. He said that the ideas for the technology (i.e. the LED strip along the mouth) from the participant who used it for his father's grill. The SCRUM sessions and critiques facilitated sharing, on top of the sharing that happened because they were working in the same space. We will discuss how these opportunities offered opportunities to integrate participants' values in the next section.

3.4.2 Value Analysis of Infrastructure for Organization

3.4.2.1 Organizational Infrastructure Supporting Values-Based Designs

The organizational infrastructure in the MoveLab helped participants allocate roles and solidify components to use. The Dance Grid and To Do Grid both played a role in helping the groups outline their ideas in a way that the participants could reference throughout their work processes. As one participant reflected, “*We needed to learn about how to make the sensors and how to put them on the costumes and also how different movements would express different emotions and how different movements would affect the sensors as well.*” The outlines highlighted what the participants needed to do and the different roles participants could contribute to for both the dance and technology promoting the value of *self-direction*; however, in the group work there was also social tensions around this. For example, when one group wrote down what they needed to practice in the dance one of the participants suggested that she be the director while the others were the dancers. One of the other participants also wanted to be a director, so the participants had to figure out which one was going to do what. The one with more social *power* became the director. We observed various negotiations and compromises throughout the workshop with boundary objects sometimes facilitating these processes providing pathways such that the disagreements did not always compromise the values participants were able to integrate into the dances.

In the Day of the Dead, participants had other types of organizational processes built into the infrastructure. The peer-critiques and the SCRUM in the Day of the Dead workshop helped engage participants with one another’s work and build excitement. The

participants started off in each presentation relatively quietly with just the researcher asking questions. However, the discussion of each person's work would pick up when the researcher got into probing about the specifics of how the puppet might respond in certain situations. Here is one example of a critique of the participant's work who ended up creating the phone charging emoji pillow:

Researcher: *"so when it gets a message that the battery's low from the phone, what, what does the puppet do to express its panic?"*

Ashley [presenter]: *"it gets big [raises hands and expresses anger with face]"*

Researcher: *"it gets mad?"*

Ashley: *[nodding]*

Researcher: *"I think you might need some sound too"*

Jaqueline [peer]: *"it could go 'ahhh ahhhh ahhhhh'"*

Ashley [presenter]: *"the output, output sound [pointing at her storyboard] has a calming sound [referring to what she already has when its fully charged] that could change to 'wahhhhhh'"*

Researcher: *"or I mean you could also have it, I like the idea of a puppet representing your phone dying so every time the battery gets lower [demonstrates the puppet by slouching her body] and lower [slouches her body lower] and lower [slouches lower]"*

Community Leader: *"oh I can see that"*

Devon [peer]: *"Oh you could use a string"*

William [peer]: *"Oh I just got an idea from you two. What if it goes like this [puts hands up above his head and out to the side] and after a while it decreases its arms [slowly lets his arms fall] and they go doooooown"*

Researcher: *"Yeah decreases the arms"*

Jaqueline [peer]: *"Yeah and it reaches the floor"*

Community Leader: *"Yeah that's good, that's good, that's good"*

The critique session had many of the participants positively reinforcing each other's ideas. When a participant presented their work many of the other students would get excited about the idea and offer supporting ideas to help the presenter take their idea to the next level. The critiques embedded the values of *achievement* and *power* from the success participants could feel both in what they presented and in the ideas they provided to the other participants.

The critiques also served as an accessible form for one of the community leaders to contribute her ideas and expertise to the participants' designs. The group was discussing how the participant who was creating a puppet of the virus. They were in the middle of figuring out how he might represent virus *dying* possibly from the use of a needle. She offers her expertise along with the researcher in the critique:

Community Leader: *"I can see that though. Right? So, for some diseases that um could be treated at home I could see how you could use something like that [referring to using a needle]. The patient gets a chance to take it with them right"*

Researcher: *"to see how it works a little bit"*

Community Leader: *"like a trial. Take it home and then and this thing would test your blood or something that sees that it has the antibodies in it and its this light [demonstrating light blinking with hands moving in and out] emits this yellow light and and then they just have to take as many doses that it injects and then the lights gets dimmer"*

Researcher: *“I think it needs to be a green light because that seems like poison/bad thing [group laughs]”*

The common knowledge of treating your health enabled a conversation around the technology that the community leader felt comfortable contributing to with ideas that were indicative of her values. In that exchange, the community leader emphasizes the patient’s own control and agency over their health, which were indicative of values within both the *self-direction* and *security* domains. The puppets became a boundary object that the participants, researchers and leaders within the environment could use to reflect on and share their own cultural responses, expressions, and values.

The SCRUMs were noted by the leaders as engaging the participants. One leader recalled, *“People would show off their work and update people at the end of the day...they would show some cool idea that they just had and it would excite other people”*. The excitement the participants showed with one another’s work also showed in the focus group interviews. When one of the participant finished describing her project, another participant commented, *“Hers was one of my favorites. Actually, all three of them were some of my favorites”* referring to the prior three participants who had talked about their projects. The SCRUM and peer-critiques created an environment where participants felt comfortable talking about one another’s work and it became a supportive community in which they were proud of their work.

The participants were not only excited about one another’s work, but actually understood it. In the final presentation, two of the participants were not able to make it because of other conflicts. When it came time to present other participants in the workshop

stepped in to tell the story of their projects. One of the leaders reflected on this, “*That just went above and beyond what I would have expected...that’s not something that happens often...this probably couldn’t have happened if the students didn’t do the critiques or didn’t know what was going on in each other’s work*”.

3.4.2.2 Value Rejections in the Organizational Infrastructure

Within the MoveLab we identified participants who rejected working in the group activities. The infrastructure was setup to organize all of the work around group activities and sharing to promote development of a community of learners. However, we saw how participants rejected these opportunities at times because they did not find it important to participate with other participants for various reasons. Some participants expressed values within the *achievement* and *power* domain because they were scared of how the other participants might judge them. Other times it was because they felt that another person’s participation would hinder their *achievement* and therefore would rather work alone.

In the Day of the Dead Puppet camp, we noticed slightly different social dynamics. While we did witness some participants, who talked less in the group setting or seemed more unsure of their contributions in front of the whole group, we did not see anyone physically remove themselves from an activity, like we observed in the MoveLab. This could have been because more of the work was individual, because participants were not working in the domain of a performing art that they were apprehensive of, or because most of the participants were already accustomed to working with one another because they go to school together.

3.4.3 *Value Analysis of Infrastructure for Building Proficiency and Knowledge*

3.4.3.1 Supporting Values with Knowledge and Proficiency

The knowledge and proficiency that participants gained were important for being able to think dynamically about the materials, complete their designs in ways that accurately represented their ideas and values, and for their self-efficacy in the domains. Participants in the MoveLab workshop showed agency in choosing sensors, peripheral components, and choreographic and dance techniques allowing the learning experience to be formed around their decisions about how they were going to express their theme. Due to the flexibility in the design of the learning activities, the participants integrated different components from dance and computing. Two of the groups were able to dynamically think about and design interactions for integrating multiple components into their dances.

We observed that the design cases helped students gain agency over the various components by giving them experience and a place to start when they were building in the open project work sessions. In the MoveLab, participants were most likely to use the components that were integrated into the design cases. The participants were able to take the case as a starting point to remix the code into what they needed. For example, one of the design cases they were given was a capacitive touch sensor made of foil that would play music when touched. The deforestation and bullying group used this as a starting point for their capacitive touch sensors. Instead of playing music, the deforestation group played the sound of chopping down a tree when they mimed this action in their dance, and the bullying group changed the sensors to activate the color change in an LED strip in the costume.

The participants in the Day of the Dead Puppet camp were able to create designs around different electronic components, and some were able to tap into proficiencies that they already had. The open-endedness of the components allowed for the participants to bring in materials and tools to work with as they needed. One participant reflected on this: *“After I gotten my arts and crafts tools and I was really able to put my mind to it because I’m a very crafty person. I have like a whole drawer full of craft supplies, and my grandmom, at her house she has like a whole room full of craft supplies so I am a very crafty person.”* This student had experience working with crafts with her grandmother and therefore had experience with certain tools and had developed cultural practices around making. When she was enabled to work using the tools she was accustomed to, it helped her focus on her design. She valued the *power* she had over the resources to *achieve* her design.

Within the Day of the Dead Puppet camp the greater length enabled the participants to work with more design cases with the components than the participants working in MoveLab. The design cases helped get participants to experience some of the common errors that novices make when working with an Arduino for the first time. One of the leaders reflected on the errors that many of the participants faced on the initial activities with the Arduino, *“They didn’t have the right COM port, or the Arduino wasn’t plugged into the computer itself. Stuff that wasn’t a problem with the circuit itself”*. The participants encountered these errors in a simple environment before they might encounter more complex errors. The leaders also noted that the participants level of being able to debug the circuits by themselves increased over the course of the workshop. As we will discuss in the section below the participants showed greater agency in debugging their work and helping

each other out with their circuits and code in their final projects, than they showed during the design case activities. This was in part because they had a better understanding of how to fix certain issues because the design cases gave the participants multiple experiences facing different problems. However, it was also noted that participants also had a greater desire to fix their final designs.

3.4.3.2 Technology Detracting from Values-Based Learning Experiences

Within both the MoveLab workshop and the Day of the Dead puppet camp we found the scaffolding for the participants was falling short of enabling them to gain proficiency and knowledge in the computing domains. Within the MoveLab participants struggled putting together the design cases. We had originally intended the participants to put them together play with them, and then brainstorm some ideas of how they might use the sensors and actuators in a dance. We then wanted them to share their experience and what they learned with the other participants. However, due to the barriers participants faced, most of the participants were only able to put together their case with a significant amount of help from the CS and engineering leaders, and they did not have much time for exploration. The participants were given the code and given a diagram of how the circuit was to be constructed with the breadboard and components. The participants struggled with understanding the breadboard, and most did not understand how the circuits were working once the instructors helped them. The participants were not able to visualize where the connections actually were in their circuits.

When participants were working on their final projects the researchers noted that they could not collaborate well with one another because of the size of the circuit. They

were struggling to see what each other was doing with the breadboard when another participant was working on it. Because of this, one participant would often be working with the circuit while the other waited to see what they had done. It was also difficult for the instructors to help participants because the connections were close together and it was hard for them to reference a certain point on the breadboard in order to show the participants what they were talking about.

When the participants were creating the circuits for their dances, the majority of the work was completed with an instructor working directly with a set of participants. Much of the burden of completing the circuits for the performance was shifted to the leaders rather than the participants because of the timing constraints and the difficulty participants were having at troubleshooting their own errors. This caused the participants to have to pair-down their original ideas and based on our observations, we believe that this affected level of agency and self-efficacy the participants had by the end of the workshop.

The participants in the Day of the Dead had more time to work with the technology, but still faced difficulties. While the initial design cases were there to provide guidance and a base of knowledge it was not as engaging for the participants. As noted by one of the instructors, *“The initial activities the kids didn’t care about it as much. They were much more passive. If you are trying to get an LED to light up that’s not that interesting...If it was something they were excited in they weren’t passive anymore. They were pulling me over trying to make sure I could help them to get it to work.”* While they became more engaged with their project they still had a low self-efficacy for working with the technology by the end of the camp. One of the participants commented on something she would change: *“The only thing I would change is the hardware and have somebody that’s a*

professional help me with it instead of like my brother trying to figure it out cause it was just too hard for me.” Other participants had similar qualms with the hardware, “[The hardware] was like the only thing I didn’t like. The wires like trying to figure out where all the wires went...it was confusing cause like you put this wire here and the other end here and then like it don’t work, and then you do...if you switch it then it works. I was like ‘this is confusing’...” This participant articulates the confusion using the breadboard she faced in understanding where the wires were connected and why they worked in certain configurations and not in others.

While collaboration was not as significant in the Day of the Dead as it was in the MoveLab since the participants were working on different projects, they still struggled with the size of the tools when they were trying to help one another out. The participants in this workshop also demonstrated a paring down of their ideas due to the timing constraints and being able to figure out the code and the circuits within the timeframe. For example, the participant with the pillow had the idea of incorporating lights, a vibration motor, and sound into her design but ended up just including a motor that moved the mouth of the pillow to serve as the tactile response that someone who was deaf or blind could interact with. In this case, we did not see the same drop in engagement that we found in the MoveLab when participants could not do what they wanted. It seems to be linked to how authentically the participants felt they were able to still represent their ideas while being proud of the final designs. The participant who created the emoji pillow was able to *love* her creation, whereas in the MoveLab we saw several participants who liked what their dances stood for but not necessarily the dance itself. Designs will always need some reconfiguration because of resources, but the extent at which participants are able to

accurately represent their ideas poses implications for their values-based learning. The technology becomes problematic if it inhibits participants to feel ashamed of what they created or that they were not able to accurately communicate the values they have embedded in their designs.

3.4.4 *Summary of Infrastructure*

As discussed in the previous sections, the infrastructure had things that worked and things that could have been improved providing some insight into characteristics of *good infrastructure* and *bad infrastructure* in these types of environments. In the following three sections, I will discuss some of the takeaways that can be gleaned from these two case-studies.

3.4.4.1 Takeaways from Infrastructure for Reflecting on and Sharing Values

The driving questions guided the participants in various ways to create personally meaningful computing objects. The questions posed in the MoveLab workshop did not have any student lost for ideas by the end of the participatory design activities, but also did not stimulate the same types of conversations that occurred in the Day of the Dead Puppet camp. The Day of the Dead Puppet camp's driving question surrounding death, mourning and loss was able to generate discussion targeted at exploring ideas of cultural expression and practices that were not prevalent in the MoveLab Workshop. We found that when engaging in discussions surrounding culture and cultural expression it is important that participants do not feel that another culture is being imposed upon them. To ensure this, it could be important to discuss the emphasis on learning about different cultures and exploring similarities and differences between them. Depending on what the learning

environment is trying to achieve, these questions have demonstrated that they can support different types of reflections on values on culture. One important characteristic about both of the questions was that they provided an accessible topic that participants—including the disciplinary leaders and community members—could contribute to, bringing in expertise from their lives.

The participatory design activities were generally successful across the participants within both learning environments for stimulating them to reflect on things that were representative of their values. We attribute the success of the activities to a few things. First, the activities were broken down into small reflections on the problem question and how it related to emotional and physical expression. Furthermore, we scaffolded the participants to link these expressions to the technology, materials, and other expressive medium (i.e. dance/interactive object design). Second, we implemented some of the initial activities using the infrastructuring strategy of formats (P Ehn, 2008) such that the leaders could serve as a cognitive model (Collins et al., 1989) as they outlined their own reflections highlighting the important aspects of the activity.

When it came to sharing work with people outside of the learning environment, the two case studies demonstrated the constraints and requirements can have benefits and drawbacks depending on the design. The requirement having to present offered opportunities for the participants (that might have opted out otherwise) to have their achievements recognized and their ideas listened to by a supportive audience. However, if done poorly this could have an opposite effect, with intensified failure felt by the participants. Part of the failure in the MoveLab was attributed to working with novices in the domain of a performing art without the needed scaffolds. We do not claim that all

integrations between performing arts and computing are subject to failure, but suggest it creates important considerations in the design of the learning environment. For example, these learning environments could be designed to work with participants who are already skilled in the performing art that you are working with, it could be designed to support alternate ways of sharing work outside of live performances, and/or provide the participants with the extensive scaffolding and time that is required in the performing arts discipline to mentally and physically prepare participants for a live performance.

3.4.4.2 Takeaways from Infrastructure for Building Knowledge

The components and design cases formed the basis of the infrastructure for the participants to build knowledge. We found that focusing the unit of analysis on the level of the components in the environment provides an understanding of the types of concepts, opportunities, and values the learning environment prioritizes. The analysis revealed the issues surrounding learnability and usability of the Arduino and other electronic components into the participants' designs. These issues placed constraints on what and how the participants were able to build. However, we did find that the design cases the participants constructed provided a scaffolded way to interact with the components, similar to the "problem- to project-based" learning environments (Brigid J.S. Barron et al., 1998). The cases sometimes proved to be more difficult than intended, but we found benefits in cases that: (1) introduced participants to issues they might hit in their final designs, and (2) could serve as starting points for participants to remix as they created their final designs.

Furthermore, design cases that used computing as an expressive medium, stimulated engagement and motivation. For example, the fixed designs that the participants

would copy, such as making an LED blink, did not stimulate a desire to complete the task in the same way, that activities requiring students to contribute to design decisions, such as the emotional lamp. Cases like the lamp, had the added benefit that participants wanted to convey their ideas and thus had to be invested in, in order to do this. Therefore, our research demonstrates the benefit in integrating the ethos of using computing as an expressive medium even in the smaller introductory design cases.

Outside of constructed design cases, we also had design cases from professional artists and technologists. Our work suggests that professional design cases can serve as a good tool for getting students thinking of forms of expression as well as the affordances and limitations of the technology and medium the professionals are working in. The professional cases might have contributed to the high bar for success that participants in the MoveLab set for themselves. In order to set more realistic expectations, it might prove useful to talk about the constraints and capabilities of the technology components in the environment in relation to what the professional cases are accomplishing.

3.4.4.3 Takeaways from Infrastructure for Organizing Work Processes

The learning experiences had informal and formal protocols to engage participants in organizing and sharing their ideas and work with others. In the MoveLab, we found success when the protocols forced discussions around work allocation and provided the groups with a shared artifact to reference as they negotiated between roles and design decisions. The organizational protocols in the MoveLab for sharing work were rejected in activities when the end result needed to be performed (ex. Dance sequences). This

demonstrated the need for low barriers for success, while building trust in the learning environment between participants.

In contrast, SCRUMs and peer critiques in the Day of the Dead Puppet camp facilitated important engagement between participants. They were successful because they: (1) enabled opportunities for all participants to engage with each project, (2) provided a way for the participants to interact in ways they were comfortable with, and (3) stimulated ideas that could be built on by one another. The success of these discussions was also attributed to the fact that they were led by a leader would solicit feedback from the rest of the participants. Furthermore, the accessibility of the problem spaces enabled the discussion to get to a point where it was easy for other participants, including the community leaders, to contribute. These protocols offered a low bar for participants to engage in discussions while creating a space for shared reflection on cultural expression.

3.5 Conclusion

The two case studies presented focused on studying how we dynamically support students to create personally meaningful experiences with physical computing. We began this exploration with the MoveLab workshop in which we investigated ideas surrounding participants' self-concepts and how they were able to *see* themselves within the domains of computing and dance. We found participants negotiating ideas surrounding their different perspectives of the dance and technology and how it related to their view of themselves. In terms of the participants' ability to integrate a positive view of themselves with computing and dance, we noted the importance of multiple roles for participation, a supportive community, and the ability for the participant to bring in their values to what

they were creating. By engaging the participants in participatory design activities, we were able to facilitate flexibility in the environment for the participants to create a personally meaningful experience for themselves. The dances became boundary objects which embedded the values of the participants and instructors in ways that then facilitated group work and learning.

The success of these activities stimulated an investigation into how the infrastructure was both supporting and inhibiting participants to work and act in the environment based on their values. The construct of values useful in understanding not only the benefits but also the drawbacks we found within the learning environment. Dissention between participants self-concept and the disciplines was caused by participants' past experiences in which they decided the disciplines did not align with their values, rejection of the values embedded in the group work, and their fear of failure. Values was intended to provide a nuanced perspective to design and analyze the learning environment for supporting various students.

The Day of the Dead Puppet summer camp was the second study that we used to analyze the infrastructure for how participants values were and were not being supported. We analyzed the data from the perspective of values and compared it with the findings from the MoveLab. The driving questions supported reflection on participant values in different ways: the MoveLab stimulated participants to reflect on values embedded within issues they chose and cultural responses expressing emotions and reacting to stimuli associated with these issues; the Day of the Dead Puppet activity stimulated participants to reflect on cultural symbolism and responses to death and mourning as well as the expression of emotions related to these topics. We found various values were embedded

within the designs of the participants with the Day of the Dead Puppets having a more explicit integration of one's culture rather than the implicit integration we identified in the MoveLab.

Across both experiences the participatory design exercises provided the needed support and flexibility to stimulate participants to bring their values into the learning environment. The exercises guided participants to reflect on their values in ways that could be translated to design of a computational object. Furthermore, creating large diagrams and pictures in these exercises enabled these artifacts to be referenced by the participants and instructors in the learning environment, stimulating shared discourse around values and cultural responses. Furthermore, infrastructure to enable this type of discourse such as the critiques and SCRUMs were able to provide consistent engagement between the participants in the Day of the Dead Puppet camp.

The MoveLab drew our attention to the opportunities provided by integrating physical computing with artistic disciplines. The participants were able to interpret their dances in different ways which promoted coordination around these boundary objects. Participants did not need to agree on what the dances actually meant as long as they were expressing similar emotions. This provided pathways for the participants to have values-based learning experiences without having to agree on what values were being integrated into their work.

Infrastructure that detracting from student values around *achievement* and *power* could be the most detrimental to the learning environment. The instances when participants felt like they could not succeed or were faced with sharing work that they were not proud

of led to rejection of the activities and disciplines. The infrastructure needs to provide the proper scaffolding and support within the materials and tools so that participants can engage with the material to gain knowledge and self-efficacy. In both environments we saw how the tools became a barrier to their engagement and self-efficacy within the computing domains; furthermore, in the MoveLab this impacted the pride and achievement that the participants felt over their final designs. The point of values-based learning in physical computing learning environments is not just to make technology accessible within the workshop in a way that is meaningful to participants, but to provide an avenue to continue this dialogue in the participant's life. If we intend to engage participants who are stereotypically underrepresented in computing it is important that they have continued access and knowledge to apply technology in ways they find valuable. If they leave these environments feeling incompetent or that they *need a professional* in order to work with it, we have taken this opportunity away from them. In the next chapters, I explore the physical computing tools from the perspective of how they promote engagement with the concepts underlying physical computing and their self-efficacy to engage with the discipline.

3.6 A Note on Values-Based Learning and its Limitations

Values-based learning intends to situate design of learning environments in values and culture to enable the designer to be reflective of what aspects of the infrastructure might be promoting particular value domains and strategies of action that might include or exclude students. When designing these two original learning environments we were primarily concerned with providing enough flexibility and support for students to be self-directed to create something that is a reflection of their values. This was intended to create a personally meaningful experience while providing an opportunity for them to share their

values and culture with others. The concepts underlying values-based learning are not designed to stand in opposition to the concepts within constructionism or project-based learning, but they are intended to provide a way to examine and explore design choices within these types of environments.

When implementing values-based learning it is important to acknowledge the values embedded in the learning environment and their relations to when and how values manifested in the data. Incorporating values-based learning within the context of physical computing education we are implicitly suggesting that there is some alignment that can be made between the participants' values and what they might be able to create with the technology. On top of this the technology choices that we make and how we structure the work activities also has implications on the values integrated into how it forces participants to work. I made explicit choices to use an Arduino because of the flexibility it provides to enable participants *self-direction*, but at the same time it introduced constraints on the types of things participants were enabled to do. The tools and materials chosen to scaffold participants are important to how you analyze the values within the environment.

Another important note on integrating values-based learning, is that as researchers, we bring our own biases to how we interpret data and it is possible to misunderstand a participant's reasoning for engaging in a particular way or for not wanting to engage at all in an activity. This could happen because the participant has not indicated their reasoning for their behavior or because they hide the actual reason (i.e. face-saving (Goffman, 1956)). As Geertz states, "that what we call our data are really our own constructions of other people's constructions of what they and their compatriots are up to" (Geertz, 1973). Using multiple sources to triangulate findings across the course of the study (Lincoln & Guba,

1985) along with providing *thick descriptions* (or rich contextual information) (Geertz, 1973), helps to improve the understanding of the learning environment and the participants. However, it remains important to try and identify when certain data is inconclusive rather than trying to force an understanding of the data from the perspective of values of which we might not understand.

While the studies offer some promising results, the learning environments had a small number of participants and they were heavily resourced. While the smaller sample size provided the ability to closely analyze the individual experiences of the participants, it also means we cannot assume these same strategies will be beneficial in contexts with many more students or those in which the educator to student ratio is lower. More research is needed to explore the applicability of these types of interventions in different contexts to see where the infrastructure is supportive and where it breaks down.

3.7 Contributions

The work presented in this chapter explored the following research questions:

***RQ1.** How does design of physical computing learning environments affect students' ability to integrate their diverse values into their learning experience?*

- 1.1 How can we conceptualize the design of the learning environment from the perspective of students' diverse values?
- 1.2 How can we analyze the learning environment from the perspective of students' diverse values?
- 1.3 What characteristics of the learning environment in physical computing activities impact students' ability to integrate their diverse values into their learning experience?

There are five main contributions from these studies. First, the work defines and analyzes the construct of values-based learning experiences as a way to create dynamic personally

meaningful computing environments. Second, it provides a method for analyzing learning environments from the perspective of values by applying the concept of infrastructure to deconstruct the design choices. Third, it identifies ways the design of infrastructure can both support and mitigate the values of students. Fourth, the research analyzes participatory design as a way to promote student reflection on values in computing learning environments. Fifth, the work identifies and analyzes boundary objects as a useful construct for exploring computing artifacts for facilitating communication and coordination between members in the learning environment. While the MoveLab workshop and the Day of the Dead Puppet camp provided an initial insight into the physical computing tools, we did not specifically target their investigation within these studies. The next couple chapters examine the tools in both a classroom and laboratory study, which were designed to bring greater insight into how novices were working and learning with the Arduino.

CHAPTER 4. BACKGROUND: PHYSICAL COMPUTING

TOOLS

In both the MoveLab workshop and the Day of the Dead Puppet camp, the tools caused the participants to run into barriers in terms of their engagement and self-efficacy with computing. The impact of these tools matter because if we want to be inclusive and equitable in the types of people supported to participate in computing, our learning experiences need build proficiency and knowledge for learners to continue working with the material outside the learning environment. Preparing students with self-efficacy and knowledge in physical computing, enables them to continue exploring these domains in ways that align with their values. Furthermore, by providing opportunities to explore the hardware and software we offering different pathways for students to engage with computing. Whether it be the physical design of computing objects, programming of the code, or designing and building the electronics, participating in each pathway could have different implications for different students. By allowing students to learn about the different pathways, we are providing access to different avenues for participants to find value within the computing disciplines. The work presented in the following sections was part of a publication in the ACM Interaction Design and Children proceedings (DesPortes & DiSalvo, 2017).

4.1 Introduction

Physical computing in education dates back to the 1970s in MIT's Artificial Intelligence lab where early explorations showed promise for the physical computing

domain to successfully enable students to think about complex ideas while working in the physical world with tangible computing devices (Blikstein, 2015; Perlman, 1974; Mitchel Resnick & Ocko, 1990). Since then, researchers, educators and designers have developed toolkits for children and hobbyists to work more easily with tangible computational devices. The rise in devices has been propelled by the popularization of the *maker movement*, which emphasizes production over consumption of technology (Dougherty, 2013; Hatch, 2014). The maker movement has stimulated extensive use of physical computing toolkits in educational spaces to engage students in experiential learning as they explore technology through design of computational objects (ex. (Buechley & Eisenberg, 2008; Deitrick et al., 2015a; DesPortes, Spells, & DiSalvo, 2016c; 2015)). While physical computing is becoming more prevalent in education, it is still unclear what students are learning in these interventions. To understand the capabilities of these tools to facilitate learning it is essential that we begin to analyze the design choices embedded in these tools based on their ability to support exploration of computing concepts.

Physical computing offers students multiple pathways into learning about computing. The toolkits have hardware, giving the tool its tangibility, and software, giving the tool its ability to support interactive behavior. Depending on the tool, students can engage in design of computational objects, explore the programming aspects, or learn about the electronics. The ability for the student to explore these domains is completely dependent on the modularity integrated into the design of the hardware and software tools. We use Sadler et al.'s definition of modularity, as the “ability to freely recombine elements” (Sadler, Shluzas, Blikstein, & Katila, 2016). We characterize modularity by the *transparency*—what is visible or obscured from the learner—and the *affordances for*

interaction—how learners manipulate and combine the tools as they build computational objects. The effect of these two characteristics are inseparable—the transparency affects what the learner interacts with, and what the learner interacts with in turn affects what is transparent to the learner.

While the physical computing toolkits range from programmable paper robots (Kwak, Kang, Lee, & Wu, 2016), to tools that can be programmed by physically manipulating the pieces (Raffle, Parkes, & Ishii, 2004) (*see* (Blikstein, 2013b, 2015) for a comprehensive review of the tools), the modularity in the electronic elements often lies on two extremes. The tools are either highly obscured, allowing students to plug-and-play the hardware without learning about the electronics, or they are left completely open-ended, with little to no scaffolding, creating a difficult learning environment for novices. While obscuring the electronics is a valid choice for some learning environments, it would be a lost opportunity if we ignored the capabilities inherent in physical computing to support pathways for students to explore the electronics. We draw attention to design decisions embedded in the modularity of the hardware tools to examine their ability to support learning.

In this chapter, I first introduce the perspective of scaffolding used to analyze the modularity of the hardware tools. We then discuss the background literature based on this construct. This sets the stage for a systematic breakdown of the hardware tools. We then outline the spectrum of modularity, analyzing the transparency and affordances for interaction. This enables us to discuss how these design decisions are tied to the tools capabilities to facilitate learning. By highlighting possibilities within the design space not often explored, the framework contributes to our understanding of how to create

educational physical computing toolkits. The goal of this breakdown is not to provide *the answer* for designing physical computing tools, but to provide a framework for examining the design choices embedded in the modularity of the tools in order to understand their capabilities for learning. After exploring the design space, I will wrap up the chapter with a discussion of possible misconceptions that students could encounter when working within the tools in our physical computing experiments.

4.2 Black-Box and Glass-Box Scaffolding

In order to examine how design decisions, affect learning, the modularity of the tools is viewed as a way to provide scaffolding. By supporting certain interactions and making certain concepts transparent, the modularity can facilitate learning. I examine the range of scaffolding that the physical computing tools can provide from the perspective of Hemlo and Guzdial's *black-box* and *glass-box* scaffolding (Hmelo & Guzdial, 1996). They describe black-box scaffolding as something that both helps a student perform an action that they could not otherwise perform, while shielding them from having to learn about how the scaffolding works. They state, "*Black-box scaffolding performs a task in place of the student performing that performance goal, usually because learning to perform that goal is determined to be unimportant*" (Hmelo & Guzdial, 1996) (Hmelo & Guzdial, 1996, p. p128). Conversely, glass-box scaffolding helps the student understand how the scaffolding works. They state that, "*it is important for the student to understand what glass-box scaffolding is providing because we want the student to be able to take on the functions that the glass-box scaffolding is providing*" (Hmelo & Guzdial, 1996, p. p128). They suggest that the types of scaffolding should differ based on the learning goals and objectives. If it is unimportant for the student to learn how to do what the scaffolding is

doing, it should be black-boxed and not fade; however, if the function of the scaffolding is important for the student to learn, it should be glass-boxed and fade as the learner gains the knowledge and skills to carry out the task for herself.

In a physical computing environment that exposes students to electronics and programming, there will be a time for black-boxing and a time for glass-boxing in both the hardware and software tools. Unfortunately, prior literature shows that the hardware elements in physical computing are usually either fully black-boxed, without a pathway for fading or having little to no scaffolding (Blikstein & Sipitakiat, 2011). This leaves us to wonder, *where are the glass-boxes?* In order to take advantage of physical computing tools' ability to teach students about the electronics, we need to examine the design space in this realm. The framework presented is one perspective that can be used to tie the design to the learning affordances in the hardware tools.

4.3 Previous Analyses of Physical Computing Tools

Other researchers have analyzed the types of physical computing tools that are available. In one analysis, Blikstein and Sipitakiat categorized the tools into two main models: the *cricket model*—plug-and-play features that hide complexities in the electronics (ex. LEGO Mindstorms, LittleBits); and the *breakout model*—open-ended designs that enable the user to extend functionality through interfacing with other electronics (ex. Arduino, Raspberry Pi, etc.) (Blikstein & Sipitakiat, 2011). The plug-and-play or *cricket models* enable users to explore various inputs and outputs to the microcontroller, but the electronics themselves are completely *black-boxed*. A prevalent argument against this

model is that, “*In packaging electronic components into higher-level modules, toolkits can obscure the technology they seek to make accessible*” (Mellis et al., 2013, p. 83).

The open-ended or *breakout models* are more complex. These solutions require that users combine individual electronic elements into circuits, understand where the connections need to be made, and how to troubleshoot the circuit when things go wrong. Depending on the shields and the packaging of the peripheral electronic elements, the modularity and therefore scaffolding can vary within these tools. One of the main arguments against this model is that it is difficult for novices, and this difficulty then creates barriers for novices to learn the programming and design concepts (Blikstein, 2015). Chan et al. (Chan, Pondicherry, & Blikstein, 2013) also discuss the need to address the gap between the accessible platforms and expert platforms through design. These researchers argue for tools that can begin to scaffold students between these the two extremes (Blikstein & Sipitakiat, 2011; Chan et al., 2013; DesPortes, Anupam, Pathak, & DiSalvo, 2016).

In a later analysis, Blikstein critically analyzes the range of designs and argues for the importance of attention to the *selective exposure* embedded within the design choices. This construct is similar to the glass- and black-box scaffolding. He discusses how what is visible and what is hidden determines the contrasting goals of *usability* and *power*. The more exposed the device is, the less useable it is; and the less exposed the device is, the less power and flexibility the tool has (Blikstein, 2015). His analysis reveals that the usable *education friendly* tools obscure many of the electronics concepts, while the open-ended kits, which are less usable, have the capability to cover the electronics concepts (i.e. the Arduino and BASIC Stamp) (Blikstein, 2015). Within his discussion he argues for explicit

analysis and understanding of the affordances of the design choices within the toolkits before introduction into an educational domain. He states, “Instead of exposing children to inadequate technologies, a more productive approach would be to identify the many toolkits that children should use throughout their school years, understand what each can accomplish, and task designers with the creation of better *bridges between toolkits*” (Blikstein, 2015). Our paper explores a way to conceptualize the design of these *bridges* in physical computing toolkits in which the electronics and programming are not obscured.

4.4 Analyzing Modularity

The design of the entire physical computing learning environment can be explored for the modularity. Depending on how the hardware and software has been packaged holds implications for what is accessible to the learner both in how they apply the modules and in what they learn from the modules. The software for example, can have complex code such as face recognition available to novices by putting it in an easily referenced module in which users do not need to understand how it works (Chailangka, Sipitakiat, & Blikstein, 2017). I’m going to focus on the modularity of the hardware for this analysis.

4.4.1 Segmenting the Hardware

In order to analyze the ability to create *bridges* between hardware tools, we segment the hardware into three categories: the electronic connectors, the programmable electronics, and the peripheral electronics (Figure 17). We use these categories to illustrate how differences in function drive differences in design explorations.

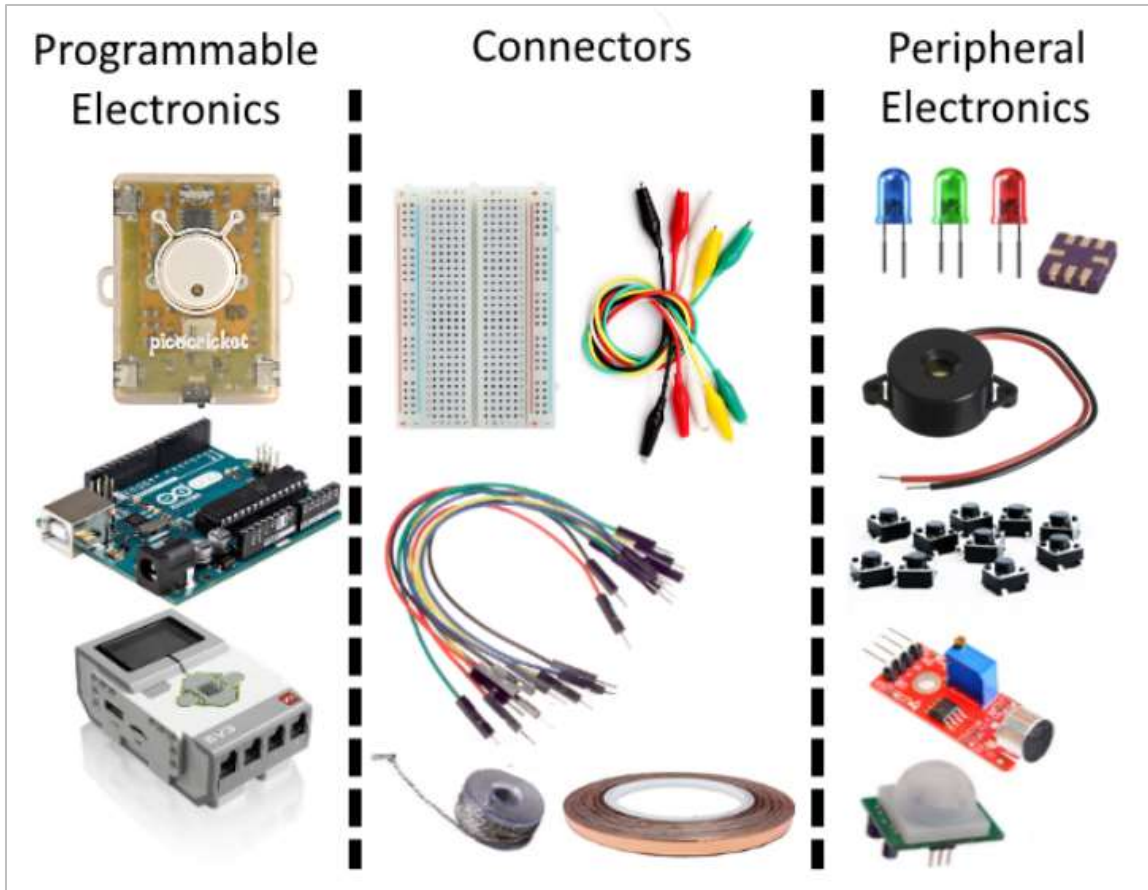


Figure 17 – Segmentation of hardware for design and analysis

A high-level discussion of these three categories examines the related literature illustrating how modularity plays an important role in each aspect of the hardware when designing for learning. We then take a deeper dive exploring the design space of the peripheral electronics. While all of these categories are important, the current literature has few examples exploring how the design of peripheral electronics affects learning. We believe many of the *bridges* between levels of modularity can be created through examining the *affordances for interactions* and *transparency* of these tools.

4.4.2 *Electronic Connectors*

Within the various designs of physical computing kits, researchers and designers have explored different methods for connecting and constructing circuits. The capabilities of these connections have been conceptualized based on their materiality and their ability to scaffold students. Eisenberg's (2002) examination of construction kits identifies the *means of connection/construction materials* as one of the central dimensions of design. He argues that the materials and their capabilities guide the user to certain creative possibilities based on the materials' affordances and constraints. He argues that this affects what the user experiments with and therefore learns about (Eisenberg et al., 2002). The materiality of the physical computing kits has been explored by researchers and designers investigating materials such as thread (Buechley & Eisenberg, 2008), ink (Mellis et al., 2013), stickers (Qi, Huang, & Paradiso, 2015), play dough (Johnson & Thomas, 2010; Schmidtbauer, Johnson, Jalkio, & Thomas, 2012), fiber optics (Eisenberg et al., 2002), magnets (Bdeir, 2009; Chan et al., 2013), and button snaps ("Snap Circuits," 2017).

The choices made about the material and how users create connections affects its use in certain situations. These capabilities make certain domains more or less *transparent* to the learner. For example, the Chibitronics circuit sticker kits (Qi et al., 2015) that enable students to lay conductive tape as traces between electronic elements lend themselves to surfaces in which the tape sticks well (ie. paper, cardboard, hard smooth surfaces, etc.). This would guide users to explore designs such as interactive paper creations but would be more likely to obscure domains such as wearables where the sticky conductive traces might not stick. A tool such as the LilyPad Arduino would be more appropriate for bringing transparency to wearable computing.

The materiality also guides the permanence and therefore the learners' *interactions* as they experiment with the tools. In playdough circuits (Johnson & Thomas, 2010; Schmidtbauer et al., 2012), there is a distinctive ability to continuously mold, reposition, and repurpose the dough as a user is working with it. This stands in contrast to circuit stickers, which once laid are more difficult to undo and reuse. While there is no empirical data on the differences between the modularity of these two models in electronics education, research in other educational fields suggests that being able to receive feedback, reflect on mistakes, and iterate upon the design can be an essential part of the learning process (Ericsson, Krampe, & Tesch-Römer, 1993; Schön, 1987).

The types of connections can also affect students' interactions by determining how easily students can make connections to recombine elements. Chu et al. investigate six types of connectors in an investigation of children's motor skills (ages 8-11) (Chu, Saenz, & Quek, 2016). They found that children often either misused the connectors due to an incorrect mental model or they had difficulty physically operating the connector. Children were most successful at using the electronics when the connection points were obvious and there were few steps to use it (Chu et al., 2016).

Blikstein takes a slightly different approach to analyzing the connections between the tools. He describes the connections through the lens of *embedded error correction* classifying tools based on their ability to guide students into making connections correctly and thus avoiding errors. He draws attention to the spectrum of design choices from no error correction (ex. LilyPad, Arduino, Chibitronics), to error correction in the form of specifically shaped connectors designed to guide the user to correctly insert them (ex. Phidgets/GoGo board/Cricket), to correction in the form of connectors that cannot be

inserted incorrectly based on size, shape or connection capabilities (ex LittleBits/LEGO MINDSTORMS) (Blikstein, 2015).

Embedded error correction relates to modularity in the students' ability to know how and why to combine the elements together and the ease at which they can complete this. Embedded information can make interactions more seamless by scaffolding students to more rapidly combine elements with fewer mistakes. Depending on how these characteristics are implemented the *transparency* into the tool often decreases. Tools like LittleBits and LEGO MINDSTORMS *black-box* much of the electronics and do not require students to think about *how* to connect the elements but instead whether they *can* connect them. For example, LittleBits alters the perception of a circuit, which becomes a line of *Bits* instead of something that resembles a complete circuit. Within this level of transparency, there is no visibility into how electricity is flowing through the elements and so it is impossible for a student to learn anything about it. However, the embedded error correction in the magnetic connectors that guides "legal" connections, enables rapid prototyping. This opens up opportunities to learn about design without the overhead of learning the electronics.

On the flip side, when working with tools like the LilyPad Arduino that do not scaffold connections, students might experience more errors when interacting with the elements. However, the *transparency* into the connections enables the possibility for students to learn about electronic concepts. For example, in an E-Textiles workshop, Peppler and Glosson demonstrated that students learned about current flow, important aspects of connections between the elements, and the polarity of elements (Peppler & Glosson, 2013). These concepts were transparent to the learner, who could not create a

circuit without implementing knowledge of these concepts in the construction of their circuits.

4.4.3 *Programmable Electronics*

The programmable electronics are the interface between the software and the hardware in a physical computing toolkit. The programmable electronics are often the central entities of physical computing projects because they control the interactive behavior of the hardware. The modularity of the programmable electronics affects the *transparency* of the communication between the hardware and software. When the programming is not obscured, the user uploads code to a programmable integrated circuit (IC). The code can create sophisticated behavior that would be more difficult to create without programming (i.e. independent logic gates).

The transparency determines the accessibility of the IC and affects the user's control of the IC. For example, the Arduino's popularity was due in part to its inexpensive scaffolded packaging that enabled untrained professional hobbyists and designers easy access to powerful hardware (Mohomed & Dutta, 2015). The Arduino platform makes the pins transparent and therefore allows the users to capitalize on their functionality. The markings on the Arduino platform and documentation guides *interactions* with these pins making it easier for a professional to combine into designs with other hardware.

The scaffolding of the programmable electronics goes from none at all (i.e. just the IC), to scaffolding that completely obscures the pins of the IC. Mellis et al.'s "Untoolkit" is on the side of no scaffolding. They have their users directly integrate the microcontroller chip into a conductive ink circuit (Mellis et al., 2013). *Interactions* of users to combine

elements together are only guided through documentation because there is no opportunity for scaffolding in the hardware. On the highly scaffolded end of the spectrum, there are tools such as LEGO MINDSTORMS. This tool has specific places for proprietary peripheral components to be plugged into the main controller. The IC and access to its pins are completely obscured from the user. The GoGo board is somewhere in the middle because it obscures the particular microcontroller pin connections but has transparency into the connections for particular inputs and outputs to the microcontroller circuit. There is flexibility in terms of what inputs or outputs go where; for example, there are 8 analog inputs, but a particular set of sensors are not specified for these pins (Blikstein, 2015; Sipitakiat, Blikstein, & Cavallo, 2004). Similarly, shields for the Arduino such as the Grove Sensor shield also strike a balance between what is and is not visible in the connections to the IC. The shield gives you access to different types of ports such as digital, analog, I2C (Inter-Integrated Circuit synchronous protocol for sending data), and UART (asynchronous serial protocol for sending data). The connectors are the same across the types of ports so the user has to make an informed decision about when to use which one.

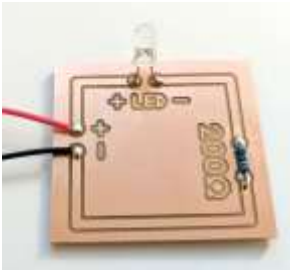

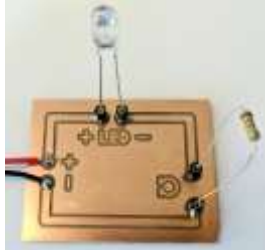


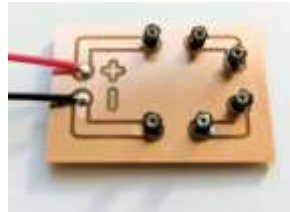
The modularity of the programmable IC is tied to the capability of the toolkit to support students in understanding how the software is communicating with the hardware. Depending on how the design exposes or obscures connections to the IC pins designates the set of *interactions* required by the user to facilitate communication between the hardware and software. This affects how the students understand the connections and the *transparency* of the processes through which the code is affecting the peripheral electronics. Booth et al. demonstrate the importance of this based on their finding that a breakdown in a user's understanding of the interactions between the hardware and software

can lead to difficult errors for the user to overcome (Booth, Stumpf, Bird, & Jones, 2016). Research has yet to empirically investigate what types of modularity facilitate understanding of the hardware and software interactions, and how this affects the usability of the tools.

4.4.4 *Peripheral Electronics*

The modularity of peripheral electronic elements used in physical computing kits can exist in a wide variety. These elements vary in complexity and transparency affecting users' interactions when combining them. A closer examination of the design space within these abstraction layers can be the key to enabling designers to create the *bridges* between the *usable tools* and the more *powerful tools* that were highlighted by Blikstein (Blikstein, 2015). We have created a design spectrum to examine the affordances of the various electronic elements for learning. For the sake of simplicity, I will discuss these tools in the context of wired connections between electronic components. Wires were chosen because of their prevalence and the ease of iteration using tools on the market. Focusing on wires places constraints on the design space, which are useful now, and can be removed in future explorations when they are needlessly inhibiting. Table 11 traverses six levels of modularity using circuit boards that we have designed

Table 11 – Spectrum of Modularity for Peripheral Electronic Components

	LEVEL	EXAMPLE
Multiple Component Circuit Boards	Level 1: Plug-and-Play Circuits with Static Electronic Components	
	Level 2: Plug-and-Play Circuits with Static and Variable Electronic Components	
	Level 3: Plug-and Play Circuits with Variable Electronic Components	
Single Component Circuit Boards	Level 4: Static Single Electronic Component Modules	
	Level 5: Variable Single Electronic Component Modules	
Multiple Component Circuit Board	Level 6: Open Circuit Modules	

In the discussion of this spectrum, we refer to the electronic *components* as the individual electronic pieces such as an LED or a resistor. We combine electronic components with circuit boards to create glass-box scaffolding for the learner. The modularity is discussed based on the types of *interactions* that are possible with the circuit boards, which make certain concepts and electronic component behaviors more or less *transparent* to the user. The spectrum starts with Level 1 circuit boards that have the least transparency, and traverses to the Level 6 circuit boards with the most transparency into the concepts and components. In describing the levels, we highlight differences in *interaction* and *transparency* that could help identify design choices to facilitate learning.

The spectrum serves as a guide for creation of glass-box scaffolding that can fade throughout the learning process. The levels draw attention to design choices that could be useful for scaffolding students to learn about physical computing concepts, while emphasizing the levels as steps for designers to explore the trajectory of glass-boxes for students to transition between. Using examples from industry, the literature, and some of our own, we will examine areas for exploration in the design space of physical computing tools.

4.5 Spectrum of Modularity

4.5.1 Level 1: Plug-and-Play with Static Components

The first level of modularity comprises of multiple electronic component modules where the components are soldered into a circuit that can be directly connected to a microcontroller or another circuit. Table 11, contains an example of an LED and resistor circuit with the components soldered to the board. There are many examples of these tools

in use today. Figure 18 shows two examples of circuits that have been packaged into modules: (a) a LilyPad Arduino light sensor, and (b) a microphone breakout board. Circuit boards at this level can be seen as a tool that gives students the ability to play with particular sensor, actuator, or pre-programmed module but only requires them to understand the circuit as a black-box with inputs and outputs.

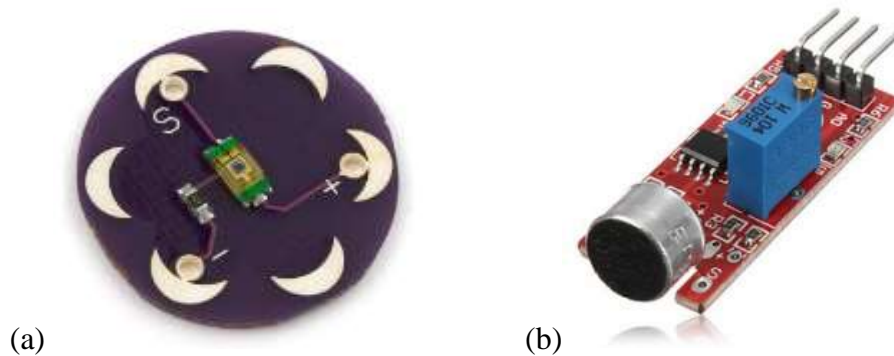


Figure 18 – Example plug-and-play circuits that provide users with simple input and output pins to gather or provide data: (a) LilyPad Arduino light sensor (“LilyPad Light Sensor,” 2017) and (b) Microphone breakout board (“High Sensitivity Sound Detector,” 2017)

Each of the modules in Figure 18 has power connections, ground connections, and signal connections. All of the internal connections are masked from the user, which can have only a few components, such as the light sensor (Figure 18(a)), which has 2 components—photoresistor and pull-down resistor—or several components in the case of the microphone breakout board (Figure 18 (b)), which has a microphone, analog-to-digital converter, operational amplifier, resistors, etc.

Using plug-and-play static modules allows the user to *interact* with the circuit boards based on a signal that they either send or receive, and there is *transparency* in how the circuit board interacts with the world. Depending on how it is used, the student can

learn about the signal in a variety of ways. For example, the user could observe the signal from the light sensor in Figure 18(a) in an abstract way, such as a number representation output from a microcontroller, or as a concrete physical representation, such as connecting it in series with an LED to visualize how the light sensor affects the brightness. These various representations could lead to different understandings of what is going on in the circuit and possibly how signals are being manipulated in the code.

Within the literature, Sadler et al. begin to investigate some of the affordances of this plug-and-play design using an LED circuit in a virtual world (Sadler et al., 2016). They found that having a plug-and-play design, in which the user doesn't have to wire the LED, increased the number of designs the users could make with the LEDs. However, by adding a simple wiring task they could raise the cognitive flow—or the alignment between the user's perceived difficulty of a task and their perceived ability (Csikszentmihalyi & Csikszentmihalyi, 1992)—of the users as long as the task was not too difficult (Sadler et al., 2016). The authors echo the importance of understanding the learning goals in order to choose how to package components, “Modules may be a poor choice if one's goal is to educate a designer on the technical aspects of a system, since the details are hidden. However, if the primary goal is to enable functional prototyping of a creative idea as quickly as possible, then modules are effective candidates” (Sadler et al., 2016, p. p144). Their study examines virtual circuit boards only within this first level of modularity. In the following sections, we examine other levels that might enable rapid prototyping along with educating the designer about electronics concepts.

4.5.2 Level 2: Plug-and-Play with Static and Variable Components

The second level of modularity involves predefined circuits, similar to Level 1, but integrates pin headers for some of the components so the user can experiment with particular aspects of the circuit to change the behavior. Table 11 shows an example of the LED circuit in which the resistor becomes a variable component. By creating an aspect of the circuit that is variable, the learner is guided to experiment with specific manipulations to scaffold her into learning about a certain concept. The variability can create *transparency* of a component and/or a particular circuit concept as it guides *interactions* with the circuit. In the case of the LED circuit, the concept of resistance is now accessible to the learner through tangible interactions that were not possible in the first level of modularity.

There has been little exploration of the design space in this level of modularity; however, we see potential to facilitate learning. The most similar breakout boards are ones like the LCD breakout board in Figure 19.



Figure 19 – LCD Breakout board that enables the user to control the brightness of the LCD using a potentiometer (“16x2 LCD Keypad Shield for Arduino Version B,” 2017).

The LCD circuit has a potentiometer that controls the screen brightness, which gives the user some ability to manipulate a component in the circuit. However, because a knob is a conventional way for users to interact with electronic devices (ex. radio knobs), it is unclear if the learner would gain any conceptual insight from this manipulation. A more effective way of promoting students to learn the underlying concepts, might be to force the learner to make intentional changes to the circuit components in ways that build their conceptual models about electricity, the components in the circuit, or circuit itself. This intentionality should guide reflections on the changes within the circuit. While it is possible for students to complete many circuit manipulation tasks without reflecting, as designers of educational technology the goal is to attempt to increase the possibility of this reflection occurring.

Figure 20 provides two other examples of circuits that meet this level of modularity. They are designed to use the pin headers as guides for students to make intentional changes within the circuit and observe the behavior that ensues from these changes in order to make a particular concept transparent.

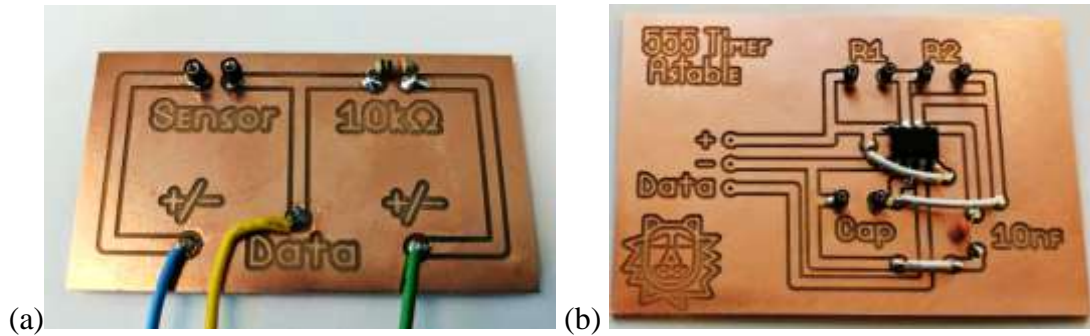


Figure 20 – Two level 2 modules: (a) two lead sensor circuit with pull-up/pull-down resistor in which the sensor can be changed; (b) 555 timer (astable) breakout enabling user to manipulate the resistances and capacitance to change signal frequency

Figure 20(a) shows a design of a module that can either be a pull-up or pull-down resistor circuit. This circuit board intends to bring transparency to how information can take different forms based on how electrical signals are translated into data (i.e. high vs. low). The students must make intentional decisions about where to connect the power and ground and can explore differences in the signals and how the code interprets the signals. The students using this board can also manipulate the sensors used in the circuit. For example, they can explore different types of variable resistors such as a flex sensor (flex-sensitive resistor), a photoresistor resistor (light sensitive resistor), and a thermistor (temperature sensitive resistor). This brings transparency to similarities between how these two lead sensors work.

Figure 20(b) shows the design of a module that demonstrates how students can be scaffolded into using more complex components, such as the 555 timer. This circuit puts the 555 timer into its astable mode that outputs a stream of rectangular pulses, of which the timing of the low and the high pulses is determined by the variable resistors and capacitor in the module. This circuit board brings transparency to the role that these variable

electronic components have on the output of the 555 timer. The students can learn about the relationships of these components without having the overhead of connecting all the pins in the IC or having to sort through the differences in possible modes of the timer. Instead of learning about an overarching concept of electronics like in the previous example, this circuit board scaffolds students into learning about the functionality of the 555 timer. While manipulating these aspects of the circuit does not teach students everything about the 555 timer, it scaffolds students into understanding how to use hardware to manipulate the 555 timer into specific circuit behavior.

4.5.3 Level 3: Plug-and-Play with Variable Components

The third level of modularity is similar to the second level, but there is less scaffolding within each circuit because all the parts are variable. There is now *transparency* brought to all components within the circuit. This level of modularity is similar to the electronics kits popularized in the 70s and 80s, in which the base board design allowed you to insert different components in guided configurations to create a variety of projects. For example, Velleman's 130 in 1 electronics lab kit (ex. Figure 21).



Figure 21 – Velleman Electronic Lab Kit 130 in 1 (Velleman, 2018)

One could consider this level of modularity as similar to blocks-based programming languages, which enable students to modify certain aspects of the functions. For example, a *move* function that lets the user tell the computer how far, but not have to deal with the entire syntax or building the function. Blocks-based programming languages have been shown successful in giving students access to the power of computing and computational thinking (Brennan & Resnick, 2012), while creating an environment that is easier for students to *tinker* with code and understand the functionality of the various blocks (Weintrop & Wilensky, 2015a). We anticipate that Plug-and-Play Variable Component tools can achieve similar goals in the electronics side of physical computing.

The example circuit in Table 11 is of the LED circuit in which both the resistor and LED can be manipulated. The information on the module still provides scaffolding to help

students correctly assemble the circuits (i.e. polarity of LED), but logically there are more mistakes that can be made in this stage when compared to the previous. Depending on the circuit and its implementation in a learning activity, it could be more difficult for an instructor to draw a students' attention to one aspect of the circuit now that all aspects are variable. The entire circuit becomes transparent and open for the user to *interact* with. For example, if a student is playing with the LED circuit she might end up switching out the LED each time she switches out the resistor. When she is done experimenting, she might not understand the cause for the change of brightness she would have experienced. However, the greater number of interactions that this design affords enables an instructor to come up with more complex problems. For example, the student can be tasked with experimenting with the multiple variables to determine which electronic component is responsible for the changes they are seeing in the circuit. This type of design can enable an educator to incorporate larger ideas into their activities such as systematic analysis of a problem and experimental design. This type of educational design is similar to the inquiry-based physics research where students explore concepts of mechanics as a way to understand complex phenomena (ex. (Laws, 1991)). Researchers Schulz and Pinkwart have begun to explore the capabilities of physical computing to support scientific inquiry (Schulz, 2016; Schulz & Pinkwart, 2016).

4.5.4 *Level 4&5: Static & Variable Single Component Modules*

Level 4 and 5 are the levels of modularity in which we transition from multiple components packaged together to single components packaged by themselves. These two levels are presented together because conceptually that are very similar but have different affordances for scaffolding between the levels of modularity. The *interactions* with the

components are now on a component level rather than based on components integrated into a circuit. The connections between components is no longer scaffolded leading to an exponential number of concepts that a learner can be exposed to.

The single component circuit boards are useful for their *embedded error correction* (Blikstein, 2015) that give a little bit more information on the component than would within an open environment. Snap Circuits (Figure 22) and LightUp (Chan et al., 2013) are both examples of this type of module.



Figure 22 – Snap Circuits (“Snap Circuits,” 2017) are an example of a Level 4 Single Component Static Circuit Board

Table 11 shows a resistor module in both level 4 and 5 forms. The circuit boards offer transparency into information about the component that would not be visible otherwise. The Level 4 circuit boards have the component physically soldered to the module so there can be specific information such as the resistance of the resistor, or polarity (ex. in the case of an LED) to ensure the LED is plugged into other components with the correct directionality. Many of the tangible electronics tools that have existed for decades have similar affordances. For example, the *Thames & Kosmos Electricity Master Lab* (Figure 23) that has circuit diagram symbols on the blocks with the components embedded inside; or Vat19’s Draw Circuits Kit in which the components are separate but you can draw conductive ink to connect them (Figure 24).

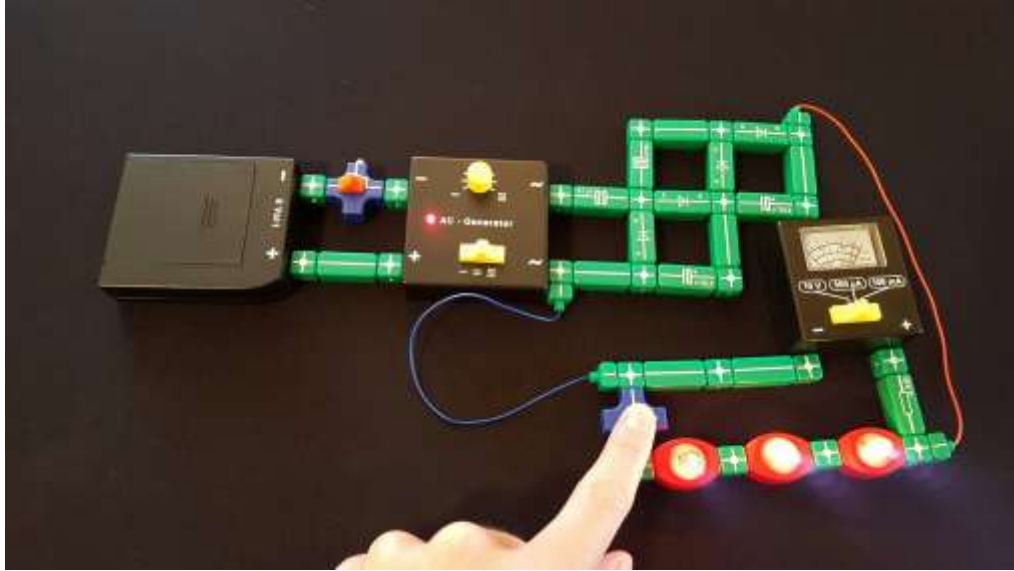


Figure 23 – Thames & Kosmos Science Experiment Kit Electricity Master Lab (Thames & Kosmos, 2018)



Figure 24 – Vat19's Draw Circuits Kit ("Draw Circuits Kit," 2018)

The Level 5 modules give this same type of error correction for things such as directionality but depending on the component may have less specificity; for example, the resistance module is not restricted to a particular resistance. While there is no guarantee that the Level 5 component will be used properly (i.e. the LED placed into the module the way the board designates), it does have the affordance of enabling the user to transfer components between abstraction layers and possibly into a final permanent solution. While we do not have any empirical data, this interaction with the component could be beneficial.

4.5.5 *Level 6: Open Circuit Modules*

The final level of modules is less defined in terms of the components but focuses instead on the relation between components. The connections between components are set (i.e. if two things are in parallel or in series), but the modules do not guide the student into using any specific component in the various slots. Depending on the number of leads the component has, the student is still restricted to some degree. Figure 36 shows examples of various types of connections in a circuit that an educator could allow their students to experiment with using two lead components. Using these circuit boards allows an educator to bring *transparency* to concepts such as resistors and capacitors in series versus parallel.

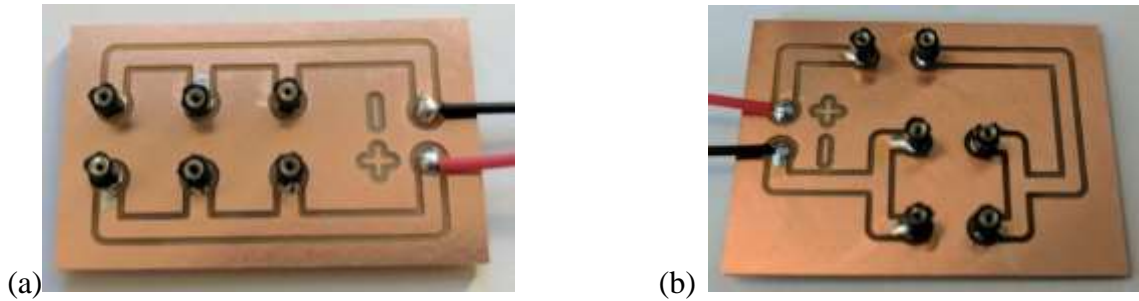


Figure 25 – Level 6 Modules: (a) parallel circuit, and (b) combination parallel/series circuit

These circuit boards scaffold the students in terms of creating a specific circuit, but they still enable the students experiment with various concepts depending on the components used. There are no examples of these particular types of modules that we know of, so it is unclear if students would remember what the connections were between the components they explored. This would be necessary for them to understand the underlying concepts. We believe this level could be worthwhile for designers to experiment with.

4.5.6 *Open-Ended Prototyping Tools*

The level above the modular components are open-ended prototyping tools that have less restrictions on use and components. With the current tools on the market there is potential for them to be improved through designs focused on usability. Breadboards are often used to test out circuits in this environment before the learner makes a circuit more permanent. A breadboard is used for easily connecting wires and other electronic elements when prototyping circuits. Breadboards often take on a standard design, which I hypothesize is difficult for new students to work with due to its complexity and lack of visibility into the connections. I have created another design called BitBlox. The BitBlox modules were designed to decrease the complexity while still keeping the open-ended

affordances that the standard breadboard contains. The design of BitBlox bring modularity and visibility to the design for the purposes of increasing its usability. Figure 26 shows the same circuit prototyped with the two different tools. In CHAPTER 5, I present a study in which I examine the use of these two tools in a classroom setting, and in CHAPTER 6, I expand upon my insights through use of these tools in a laboratory study along with Level 1 (Plug-and-Play with Static Components) and Level 3 (Plug-and-Play with Variable Components) modules.

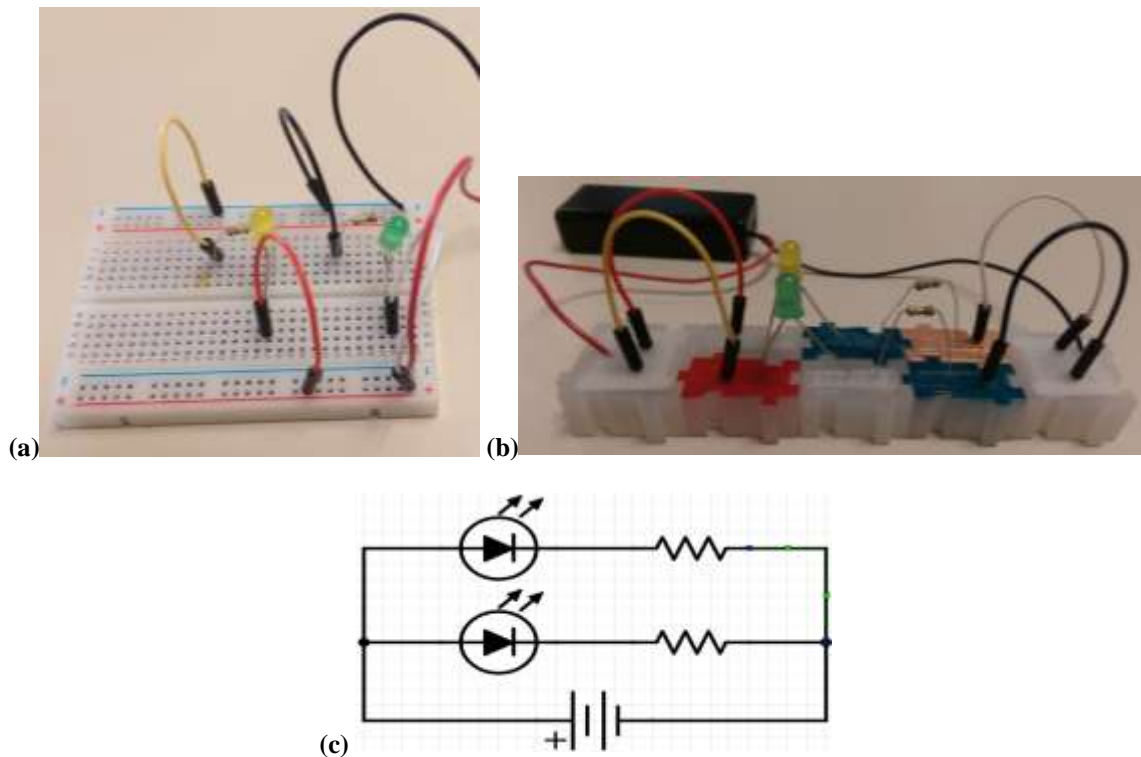


Figure 26 – Showing the same circuit prototyped with (a) the standard breadboard, and (b) the BitBlox modules. (c) Shows the circuit diagram that represents these circuits

4.5.6.1 Standard Breadboard Design

The standard breadboard design has two main sections—the outside rows and the middle columns. The holes or contact points in the row sections are only electrically connected to contact points in the same row. The contact points in the column sections are only electrically connected to the contact points in the same column (see Figure 27 (a) and (b)). There is an electrical break going across horizontally in the middle of the board separating the two sections of columns above and below the break. The other common design is to have a breadboard that consists of just the center column section from the first design (see Figure 28).

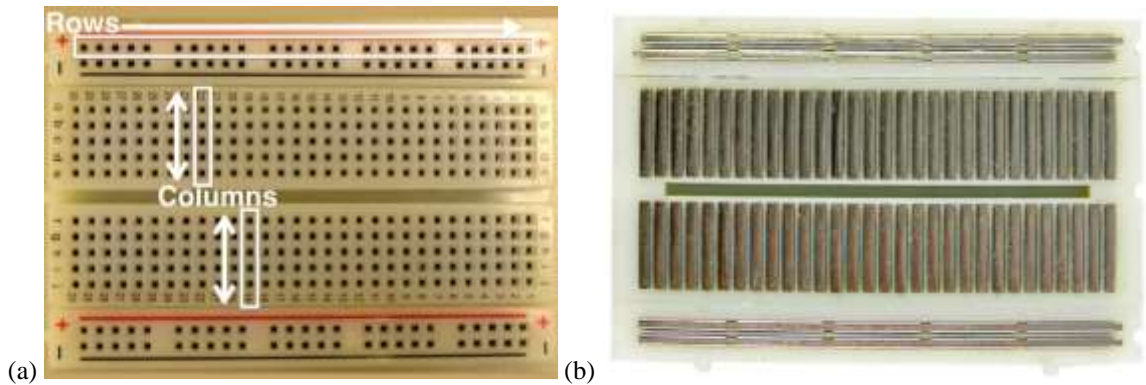


Figure 27 – Most common standard breadboard design with two sections: every hole in each row is interconnected, and every hole within each column is interconnected (a) shows the front or top, with (b) showing the back or bottom of the breadboard

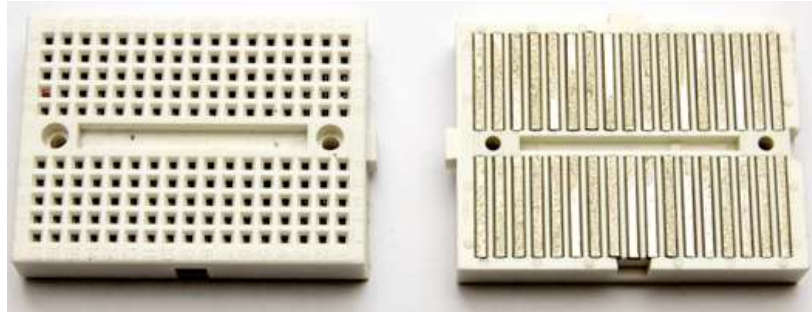


Figure 28 – Another common breadboard design in which there are only the columns with interconnected holes.

4.5.6.2 BitBlox Design

BitBlox are similar to Breadboards in that they also have holes or contact points into which wires can be plugged in. Each of these contact points is electrically connected to other contact points that have the same color and are connected. In some variations, all contact points in the board are electrically connected (see Figure 29 (a) and (b)). Other BitBlox modules are color coded to identify the connections (see Figure 29 (c)(d)(e)(f)). The multi-connection blocks have been made in order to enable components in the market to fit on them. For example, Figure 29 (d) enables users to use a button that physically would not otherwise be able to span between two BitBlox modules. Figure 29 (c) could be used with transistors, (e) could be used with something with even more pins, such as an LCD module, and (f) could be used for integrated circuit (IC) chips.

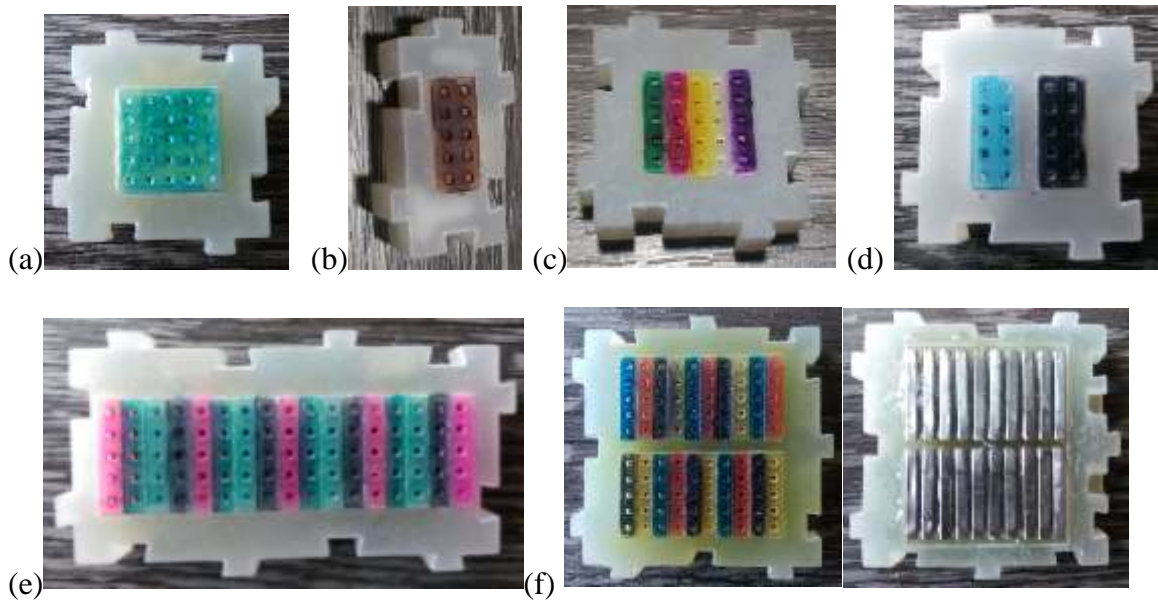


Figure 29 – BitBlox modules (a) and (b) completely connected modules, (c) five separate columns of connected holes, (d) two separate sections of connected holes, (e) seventeen separate columns of connected holes, and (f) front and back of two sections of 10 separate columns of connected holes.

This connection scheme was designed to require less memorization than the standard breadboard’s. The individual BitBlox modules can snap together like puzzle pieces. The flexibility of snapping together the BitBlox allows the user to modify the structure of their breadboard giving them the ability to organize and adapt the tool for their needs. For example, if they wanted to, they could mirror a circuit diagram more closely, like in Figure 26. We hypothesized that this would also allow the students to break up the circuit in various ways for either conceptual or collaborative purposes.

4.5.7 *A Note on the Spectrum of Modularity*

Physical computing has been explored for its ability to engage students with technology (ex. (Blikstein, 2008; Deitrick et al., 2015a; DesPortes, Spells, et al., 2016c; K. Searle & Kafai, 2015)), but it is often unclear what students are learning or how they might

transfer this knowledge to other situations. With each design decision in a physical computing toolkit there are choices being made about what is important for the students to be able to modify, and each decision affects what the students are able to learn. By focusing our attention to the modularity in the design of learner-centered tools we explore the affordances design choices could have in scaffolding students to build their conceptual models.

The spectrum outlined above is one way to conceptualize the glass-box scaffolding that can exist within the electronic components in physical computing toolkits. By designing based on the *transparency* and the *interactions* that the tools afford, the modularity physically contains the ability for the learner to explore concepts in physical computing in a constrained way that still supports constructionist learning (S Papert, 1993). The theoretical ideas underpinning this scaffolded exploration of physical computing are analogous to those offered in the problem- to project-based work (B.J.S. Barron et al., 1998; Blumenfeld et al., 1991). Educators can systematically expose students to particular concepts that can facilitate transitioning students to an open-ended scenario in which students are empowered to create with technology. It is possible that the ease of use provided by these glass-boxes enables students to go through the material too quickly without reflection on what each level of the circuit design highlights. However, if these circuit levels are seen as a set rather than designating one level as *the answer*, then these risks might be mitigated because the scaffolding will fade and the prior modules can serve as cases that are physical references as learners proceed through various activities.

The spectrum of modularity provides a framework to empirically analyze design choices successes and failures in educational interventions. The spectrum is outlined such

that it can be investigated, iterated upon, and improved rather than to mandate a final state that the design of these educational circuits should take. By drawing attention to the modularity of design choices we can begin to create learner-centered physical computing tools providing learners access to the powerful concepts of computing. I will now go over the background literature around misconceptions which will be useful for situating the findings from the studies investigating tools.

4.6 Misconceptions

The misconceptions that are within the domains of CS and electronics are important for understanding how the findings from the comparative classroom study and the laboratory study investigating the tools. I will go over the literature within the computer science, electronics and physical computing literature that shed light on misconceptions relevant for this dissertation.

4.6.1 Computer Science Misconceptions

The programming concepts that students can be exposed to in a physical computing environment are vast. Within introductory experiences, students are generally exposed to: loops, variables, and conditions.

4.6.1.1 Interpretation of Notional Machine

One of the misconceptions that spans almost any programming language/environment is a student treating the computer as an agent that can interpret information like a human instead of a notional machine. Pea coined this as the “super bug”. Students believe that a computer “has *goals*, and *knows* or *sees* what will happen elsewhere

in itself” (Pea, 1984, p. p30). This type of error can lead to many mistakes due to lack of specificity in a student’s code. For example, Pea notes a prevalence of a *parallelism bug* in which “the assumption that different lines in a program can be active or somehow known by the computer at the same time, or in parallel” (Pea, 1984). These types of misconceptions could be present in a physical computing context since the Arduino programming environment is a notional machine that has analogous requirements in how the students need to communicate instructions to it as compared with other programming environments.

4.6.1.2 Iterative Logic

Misconceptions surrounding loops or iterative logic are also expected to prevail within the physical computing context. The premise of the software that runs on the Arduino is that there is a “forever loop” that the code runs so that the microcontroller is continuously manipulating the hardware. Iterative logic is important even for a simple program like *Blinky LED*. Blinky LED turns on and off an LED, and it is often used as the introductory activity for novices (Figure 30).

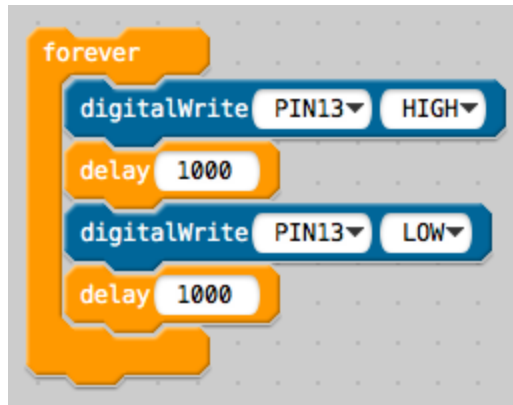


Figure 30 – Blinky LED code that is used to continuously turn on and off an LED connected to pin 13.

In order to blink an LED, most students seem to understand the concept of turning something on then off and the need for a delay in between those commands. However, the second delay that is not physically *between* anything, is more confusing. The command to turn on the LED after you have turned it off is not below it, but instead at the top of the loop.

In prior literature, novices have been recorded having several misconceptions concerning loops (Kaczmarczyk, Petrick, East, & Herman, 2010). Based on the prevalence of loops within physical computing these are probably relevant to a physical computing environment as well. Students have been shown to have difficulty understanding, “which statements are repeated in loops and where loops begin and end” (Sleeman & others, 1984, p. p17). Some of this might have been due to using Pascal as the textual programming language in this study. Research concerning loops in programming languages found great importance in the semantics used for the looping structure (Stefik & Gellenbeck, 2011; Stefik & Siebert, 2013; Weintrop & Wilensky, 2015b). Using “repeat” instead of “for” showed an improved success rate with novice programmers (Weintrop & Wilensky,

2015b). Within the blocks-based language we are using, “repeat” and “forever” are the terminology used in the looping functions. It is unclear from the literature how “forever” will be interpreted by the students. Students using “for” loops have also shown misconceptions about the control variable, thinking that it “does not have a value inside the loop” (Sleeman & others, 1984, p. p18). Sleeman et al. also identified that students sometimes believe a “for” loop will change how many times it is iterated based on irrelevant information inside the loop (Sleeman & others, 1984).

4.6.1.3 Variables and Assignment Statements

Physical computing often uses variables which can hold things such as pin assignments or values that are read from a particular sensor. Depending on the project students might be able to get away without having to implement variables, but we discuss some of the misconceptions seen with variables that could be relevant. Variables can be set to a specific value or to the result of a statement which is evaluated. Sorva found that students with a limited understanding of expressions were not able to understand the concept of evaluation of a statement. Furthermore, students believe a variable would store an unresolved expression (Sorva, 2012). Putnam et al. found that novices demonstrated frequent misconceptions with variables. One frequent misconception was that a variable could contain more than one value. They noticed other occasional difficulties that novices had with variables, such as not realizing the change in value of a variable when it was set, and novices would confuse two variables (Putnam & others, 1984). Sleeman et al. investigate novice high school students using Pascal and found that students made several errors using variables. When it came to assigning variables to one another there was a confusion in what happens when variable A was said to equal variable B (ie. $A=B$). Some

felt that the variables should be swapped, some felt that nothing would happen, and one decided that the variables would be printed (Sleeman & others, 1984). When two variables are interacting, novices seem to misinterpret the outcome. Another issue that has prevailed in the literature, is that variable naming affects how students interpret what the variable contains. The name of variables can be confusing if they are not mapped onto what a student expects them to mean (Sleeman & others, 1984). As Kaczmarczyk states, “Student applies real-world semantic understanding to variable declarations.” (Kaczmarczyk et al., 2010).

4.6.1.4 Conditional Logic

Physical computing projects often use conditional statements in order to control how the inputs to the microcontroller effect the outputs from the microcontroller. Students often have difficulties with various aspects of conditional logic. Students often harbor misconceptions about “if” statements. In Putnam’s study, students believed that a program would terminate if the condition of an IF statement was not true (Putnam & others, 1984; Sleeman & others, 1984). Another version of this misconception is that the code will return back to the beginning if a statement is untrue (Putnam & others, 1984; Sleeman & others, 1984). When an “if” statement did evaluate as true some students believed that it would execute regardless of the program’s location within the code (Pea, 1984). The spatial location of code near an “if” statement could also cause confusion. Sleeman et al. found that statements that were not in an “if” statement block but proceeded it was interpreted by some of the students as being the “else” statement. Sleeman also found that some students thought their print statements within conditional logic statements would always be executed regardless of their location within the statement. Other students demonstrated that

they did not understand conditional logic at all and believed that both the “if” and the “else” statement will be executed when the program runs (Putnam & others, 1984; Sleeman & others, 1984).

4.6.2 *Electronics Misconceptions*

Physical computing electronics in the Arduino environment often involves students reproducing circuits that they are given diagrams of how to construct. When students make mistakes, debugging often requires a base level understanding of some electronics concepts. I will discuss the misconceptions within the conceptual areas of current flow, polarity, voltage and resistance. While the literature offers some research to inform what might happen in a physical computing environment, many of the studies do not use physical circuits so it is difficult to understand what will and will not transfer into this learning environment.

4.6.2.1 Current Misconceptions

The concepts within current flow could be some of the most important concepts in physical computing in terms of being able to get a circuit working. These concepts encompass being able to: identify a complete circuit and a short circuit, understand how to handle a node (ie. multiple connections at the same point) within a circuit and understand how current splits between different paths. Engelhardt and Beichner created a validated test instrument to understand novices’ misconceptions of direct current circuits, which tested physical aspects of circuits, energy, current and potential difference. Within their study they used circuit diagrams and illustrations of real circuits. Regarding current, they found that many students were incapable of identifying a complete circuit and incapable of

identifying a short within a circuit (Engelhardt & Beichner, 2004). This is an important concept in physical computing because students need to be able to understand why parts of their circuit are not working. Students also thought of a battery as a current source (ex. thinking current does not change in relation to resistance), thought current always split evenly through parallel junctions, and reasoned sequentially about the current, viewing it as being “consumed” as it traveled through various components (Engelhardt & Beichner, 2004). In a study building upon this work, DesPortes et al. investigated some of the questions within their instrument comparing students who reasoned about both a physical circuit and a circuit diagram. In both the physical form and the circuit diagram forms, they found similar issues with students not being able to identify a short circuit, viewing the battery as a current source, not understanding current splitting and reasoning sequentially about current (DesPortes, Pathak, & Anupam, 2016). These misconceptions are prevalent throughout much of the literature (Cohen, 1983; Jaakkola & Nurmi, 2004; Küçüközer & Kocakulah, 2007). Furthermore, DesPortes et al. found that students were more likely to reason sequentially about current in the physical form than in the circuit diagram form (DesPortes, Pathak, et al., 2016), which suggests this might be more likely to occur in the physical computing environment. Sequential reasoning is one of the most common misconceptions about electricity that students harbor (Beheshti, Aljuhani, & Horn, 2014; Cohen, 1983; Grotzer & Sudbury, 2000; Küçüközer & Kocakulah, 2007) and could be difficult to correct due to its emergent nature (Chi, 2005). The misconceptions about current misconceptions could be dangerous if students are not able to identify faulty behavior in a circuit if they think it is correct based on their misconceptions.

4.6.2.2 Polarity Misconceptions

Polarity is a concept that students are exposed to in physical computing when working with the Arduino. The Arduino can be used as a voltage source and is often used in conjunction with LEDs into activities (ex.(Peppler & Glosson, 2013)). There are a set of misconceptions within the literature associated with polarity of the voltage source and how it works. One misconception is that current comes out from the positive side of a battery and the negative side isn't needed (Küçüközer & Kocakulah, 2007). In the physical computing environment this might translate into a student not understanding the necessity to have their circuit connected to ground. Another common misconception is that current comes out of both sides of the battery and clashes within circuit components (Butts, 1985; Küçüközer & Kocakulah, 2007). This might not cause them to create an incomplete circuit, but it could affect their ability to reason about and debug and incorrect circuit. This misconception would likely hinder a students' ability to understand a diode which only allows current to flow in one direction.

4.6.2.3 Voltage Misconceptions

The Arduino often acts as a voltage source for the various circuits that are attached to it. It is unclear what level of voltage is necessary in order for a novice to be successful in physical computing. Students often need to understand the concept of connecting circuits to a common ground, and they might need to understand voltage drop across components to understand how certain sensors work. One error that students often make is that they do not distinguish their views of current and voltage (Cohen, 1983). This error could cause them to misunderstand circuit behavior. In one study, where students did not have a

conception of voltage, they were shown to make up what they believed to be voltage drops across components (DesPortes, Pathak, et al., 2016). This could lead to an improper understanding if something is working correctly or incorrectly in a physical computing scenario.

4.6.2.4 Resistance Misconceptions

Resistance is an important concept that applies to many of the physical computing components and sensors (ex. resistor, potentiometer, flexsensor, photoresistor, etc.); however, it is also a concept that many students confuse. Many studies have identified that some students get confused about what happens to resistances in series and parallel, do not understand that series or parallel resistances could have an effect on the circuit, or think that resistance is something that changes with changing current (DesPortes, Pathak, et al., 2016; Engelhardt & Beichner, 2004; Küçüközer & Kocakulah, 2007). This could affect how a student interprets what is going on in a particular circuit and what information is getting sent to the Arduino.

4.6.3 *Physical Computing Misconceptions*

Currently, within the physical computing literature, there are few studies that focus on how learners are thinking about the physical computing artifacts, tools or concepts prevalent in this domain. Little work exists which uncovers the misconceptions learners create and even fewer focus on studying how complete novices are thinking about these concepts. I will review the literature that exists.

Granott's dissertation was one exploration that focused on how computational artifacts are understood. Her work focused on how adults explored computational robots built with Lego Bricks and pre-programmed with particular behavior that was complex and difficult to identify. For example, "one robot moved forward and sideways toward shadows, and after bumping into an obstacle it retreated and turned around". Both of her studies comprised of adults. She identified three phases within the developmental process as the participants began to understand the objects: *actual phase* → *representational phase* → *abstract phase*. The *actual phase* comprised of the participants understanding the behavior of the robots through its "actual actions and perceptions". The *representational phase* comprised of participants understanding the causality of what stimuli in the environment caused the behavior. Lastly, in the *abstract phase* participants connected participants understanding of the robot's behavior to how the robot's structure actually generated the behavior. The adults in this experiment were able to gain a high level understanding of how and why the robot was able to interact in certain ways using an object in which the programming and electronics were highly obscured (Granott, 1993).

When we work with students in physical computing we hope that they are able to traverse similar phases of understanding such that they gain some knowledge of the underlying concepts of computing; however, it is unclear what phases of understanding they actually traverse and what knowledge they gain from the experience. While Granott's studies focus on exploration of computational artifacts rather than creation of them, the levels of understanding could be useful in relating them to what might be seen in novice students as they create computational artifacts. It is important to note that the population in these studies consisted of elementary school teachers and graduate students, each of whom

has advanced knowledge of learning that could result in vastly different behavior and understanding than students.

The most informative prior literature in understanding what people are thinking and what issues they face during physical computing tasks are Booth et al.'s investigation of end-user developers working with Arduinos (Booth & Stumpf, 2013; Booth et al., 2016) and Sadler et al.'s analysis of beginners learning and prototyping with Arduino and LED circuits (Sadler, Shluzas, & Blikstein, 2017). In Booth et al.'s one study (Booth et al., 2016) they investigate a set of adult participants who constructed a *Love-o-Meter* which used an Arduino microcontroller to visualize a reading from a temperature sensor using three LEDs. The participants in their study were not novices and were able to use the internet to solve the problem. The participants had on average about 10yrs programming experience, 7yrs electronics experience, and 3yrs physical computing experience. Despite their experience and a reasonable level of self-confidence only 6 of the 20 participants successfully built the computational artifact with the right specifications. They found that more obstacles and breakdowns occurred when the participants were programming as opposed to creating their circuit. However, when they investigated the issues that inhibited the participants in completing the task successfully, they found that the majority of *fatal* issues were related to errors in the circuit construction. Some of the errors the participants demonstrated in the electronics were:

- Incorrectly wiring components (ex. the temperature sensor leads)
- Incorrectly identifying a sensor as not working when their circuit was wrong
- Using incorrect resistances (ex. with LEDs)
- Incorrectly inserting a component into the breadboard so it does not make contact
- Improperly testing their sensor

The authors propose two suggestions for improvements to the environment: creating better tools for testing and debugging physical computing, and figuring out how to better educate the end-user developer (Booth et al., 2016).

In another investigation, Booth et al. (Booth & Stumpf, 2013) investigate the programming environment in the context of a physical computing task, comparing blocks-based versus text-based languages. The participants had never programmed an Arduino but had some programming experience (i.e. familiarity with loops, variables, and ‘if statements’). Each participant completed two programming tasks one in each environment: one task involved remixing code of a program to extend its functionality, and the other task involved creating code from scratch to complete a task. The authors gave half the participants the text-based program for task 1 and the blocks-based program for task 2, and flipped the order for the other half. The participants were given the completed circuits to use with their code. The authors analyzed the participants think-aloud and video data for the system *learning barriers* (Ko, Myers, & Aung, 2004) to understand the usability of the various tools. They quantified success of the task completion and measured the self-efficacy and the cognitive load of the participants in each environment. They found that the task completion was low in both environments but lowest in the text-based environment. They also found that users had a lower perceived workload and a higher perceived self-efficacy in the visual programming language as opposed the textual programming environment (Booth & Stumpf, 2013).

In terms of the specific errors participants encountered, there were significant numbers within both cases. The text-based environment contributed to syntactical issues that slowed the participants down and affected their ability to focus on the program design.

These findings mirror many of the issues identified when students are using textual languages in computer science education (Weintrop & Wilensky, 2015b, 2015a). The blocks-based environment was difficult for many of their participants to make the blocks do what they wanted them to do (i.e. coordination barrier). They had issues connecting blocks and knowing whether they were connected or not. Furthermore, it was difficult for users to navigate the variety of blocks in the environment and understand what the various blocks did. Both environments encountered issues with visibility into what the code was doing. While the incorrect circuit behavior enabled them to identify that an error existed in their code it was difficult for them to understand why it was happening. As shown in Booth et al.'s other study, the complexity of isolating the issue is compounded if they also do not know whether their circuit is correct. This study provides an understanding of some of the usability issues students might face as they encounter physical computing with the Arduino for the first time. However, in the first study their participants had knowledge of programming, circuits and working with the Arduino before, and in the second study the participants knew how to code and were provided with a working circuit.

Sadler et al.'s study is one of the few studies investigating high school students working and learning with the Arduino (Sadler et al., 2017). They use data from a study conducted by Jung et al. (Jung, Martelaro, Hoster, & Nass, 2014) who investigated design of an animated agent to guide participants through an Arduino tutorial. The study encompassed 68 participants, and, although the participants' experience levels were not reported on, the authors report that they accepted complete novices. The animated agent, embedded in an LCD screen, walked participants through a 10min Blinky LED tutorial using the standard breadboard and a text-based Arduino IDE. After the tutorial, the

participants could experiment prototyping with more LEDs, resistors, paper, scissors, and markers without the help of the agent for 15min. If the participants became stuck, the researcher would help them through the error. Sadler et al. used the video data to analyze the types of errors that were made during the tutorial and exploration (Sadler et al., 2017).

Sadler et al. reported that 47% of the participants could not get through the Blinky LED tutorial without researcher intervention. Furthermore, their analysis of the errors participants made within environment found that the biggest error was a slip in one connection in their circuits leading to an open circuit (18 participants). The other errors identified were: theoretical misunderstanding creating incorrect circuits (10 participants), syntactical programming errors (9 participants), LED backwards (7 participants), misunderstanding the breadboard connection scheme (4 participants), crashing the software (3 participants), missing wires in their circuits (2 participants), electronic components breaking (2 participants), confusion of the microcontroller pins (2 participants), software usability error (1 participant) and confusion between the components (1 participant) (Sadler et al., 2017). They also found that four participants decided to not touch the electronics in their prototyping session and eight participants had a visible lack of interest in the task.

These findings provide a start for understanding the numerous obstacles and mistakes high school students face with the Arduino. The open questions from this work are: what are the theoretical errors that participants struggled with in the hardware, how were participants thinking about the tools and concepts, how did novices approach problem solving, and how would different types of prototyping tools impact the experience. While the limited amount of time given to the participants for learning how to use the Arduino

might be similar to an individual learning at home on their own, it is unclear how the experiences would be different if the participants were provided with more scaffolding.

In the following two chapters, I explore two different methods for examining the hardware tools and the experiences of novice high school and college students working with the Arduino. The first investigation is a comparative classroom study that looks at the errors and interactions between different groups using the Breadboard and the BitBlox. The comparative classroom study served as a pilot that investigated the tools within one class period of novices providing insight to the socio-technical implications of the tools. The second study, was a laboratory study designed to provide a closer look at how novices are thinking about these tools for the first time. It gave insight into the conceptual complexities beginners are faced with when working with these tools.

CHAPTER 5. BITBLOX COMPARATIVE CLASSROOM STUDY

5.1 Introduction and Context of Study

The pilot study was comprised of two class periods designed to compare the use of BitBlox to the standard Breadboard. While there are imperfections with comparative studies *in the wild*, the design of this study allowed us to observe how the various tools affected learning and collaboration, bringing to light future directions for further investigation. The work presented in this chapter was largely part of a publication in the Interaction Design and Children conference (DesPortes, Anupam, et al., 2016).

Table 12 – Student demographic information

Students	Gender		Ethnicity		Age		
	M	F	White	Black	16	17	18
Breadboard (22 total)	9	13	20	2	14	7	1
BitBlox (22 total)	14	8	21	1	11	11	0

Table 13 – Student prior experience

Students	Had Experience	No Experience	Unreported
Breadboard (22 total)	6	15	1
BitBlox (22 total)	3	18	1

The participants that completed the pre-study survey and class activities comprised of 44 high school juniors enrolled in a rural, southern US school. The majority of the students were 16 or 17 years old identifying as White or Caucasian (see Table 12). The study took place during a regularly scheduled Drafting class designed to introduce students to topics

in engineering. The majority of the students had never worked circuits before (see Table 13).

The study incorporated two classes—one class used BitBlox and one used the Breadboard. The students worked in pairs with a computer and a basic Arduino kit. There were 11 pairs on each day. Each class lasted 1hr and 40min, following an identical curriculum: surveys → presentation → simple circuit exercise → challenge circuit exercise. The presentation introduced the Arduino, explained either the BitBlox or the Breadboard, and walked through building a basic circuit for blinking an LED (Figure 31(a)). In the final step, the pairs of students worked without direct guidance to build the challenge circuit (Figure 31 (b)).

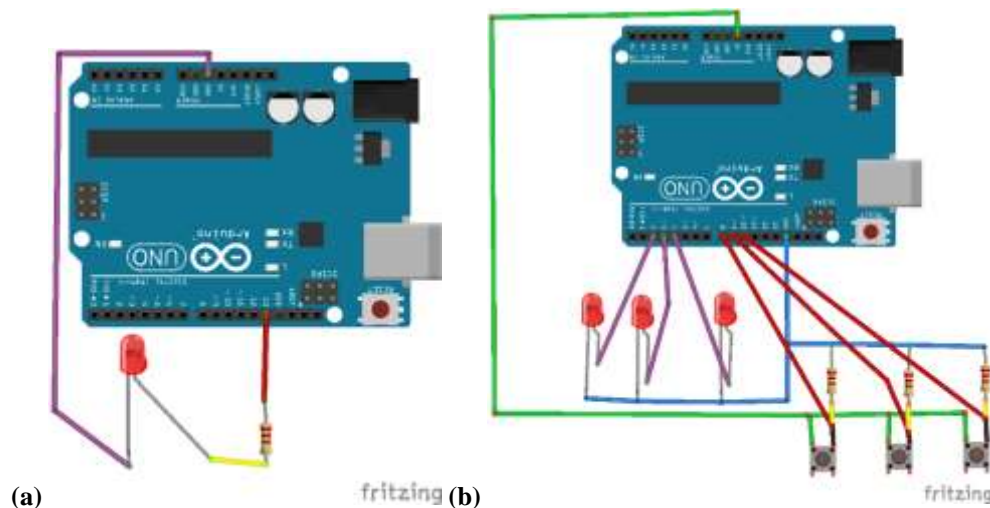


Figure 31 – (a) The schematic for the introductory LED circuit. (b) The schematic for the Simon memory game circuit.

The challenge circuit involved building the circuit needed to create the Simon memory game. The game involves the computer randomly generating an ever-increasing pattern of blinking LEDs for the player to replicate by pressing buttons connected to the LEDs. The

students had to construct the circuit and then upload the pre-programmed code that they received.

5.2 Data Collection and Analysis

Data was gathered through pre-study surveys administered at the beginning of the class and through participant observations during the class periods. The pre-study survey asked students their age, ethnicity, gender, and if they had experience with building circuits, working with the breadboard, or working with the Arduino. Three researchers taught the lesson material, distributed the supplies in the class, and gathered the observations. This enabled them to be alerted when there were issues and listen to each group explain their circuit. They agreed to split their attention on different sections in the classroom to ensure no group was left unnoticed. The researchers agreed to focus their observations within the broad categories of: mistakes, misunderstandings, questions, collaboration patterns, moments when a feature in the tool seemed helpful or problematic, and patterns in tool use. They didn't pre-define a list of codes for the observations because this was a pilot study and sticking to a rigid list would run the risk of missing key observations. Each of the researchers took hand-written notes during the class, recording a diversity of observations throughout the class because they each worked with a variety of pairs. After each class, the researchers audio-recorded a conversation amongst themselves discussing their written observations. They first talked about general observations of the class, and then addressed each group to discuss what each one of them found in their personal notes. This strategy ensured that the audio recording incorporated all of the data from each researcher.

The audio recording of the researchers' conversation was transcribed and used for analysis because it contained all of the data from the observation notes and reflections made throughout the classes. The researchers independently coded the transcription for patterns in the categories defined above. They then met to agree on a final code book with appropriate labels and settle discrepancies regarding transcribed text with different labels. Reaching a consensus, they solidified the codes (see Table 14). Observation notes for each code was then cross-referenced for patterns across and between conditions.

Table 14 – Codebook with descriptions and examples

<i>Category</i>	<i>Description</i>	<i>Example</i>
<i>Tool Usage</i>	How they used the Breadboard or BitBlox	Not connecting any of the BitBlox during their circuit construction
<i>Student Hesitancy</i>	Students demonstrating doubt and uncertainty	Wanting to be told if their approach was correct before building it
<i>Student Agency</i>	Students demonstrating self-efficacy/autonomy	Debugging circuit on their own
<i>Concept Errors</i>	Errors made based on not understanding basic concepts	Not being able to transfer a signal from one point in a circuit to another
<i>Component Errors</i>	Using components (ex. LED) incorrectly	Mixing up the button orientation
<i>Tool Error</i>	Errors with using or understanding the tool	Plugging both ends of an LED into one BitBlox module
<i>Collaboration Within Pair</i>	How students worked together in their pair	One student directing while the other was plugging in the wires
<i>Collaboration Between Pairs</i>	How different pairs interacted	One pair using a finished group's circuit for comparison
<i>Work Strategies</i>	Strategies students used to get the circuit working	Color coding of their circuit to the diagram

5.3 Findings

Below we present the findings that are most relevant to how the tools affected the students and learning environment.

5.3.1 Circuit Layout

The students using the Breadboard only had small variations between the circuit layouts. The variations primarily encompassed using different combinations of rows or columns (see Figure 32 for an example circuit). The compact nature of the Breadboard made debugging the circuit difficult for the students. We observed students having difficulty tracing out their circuit for the instructors.

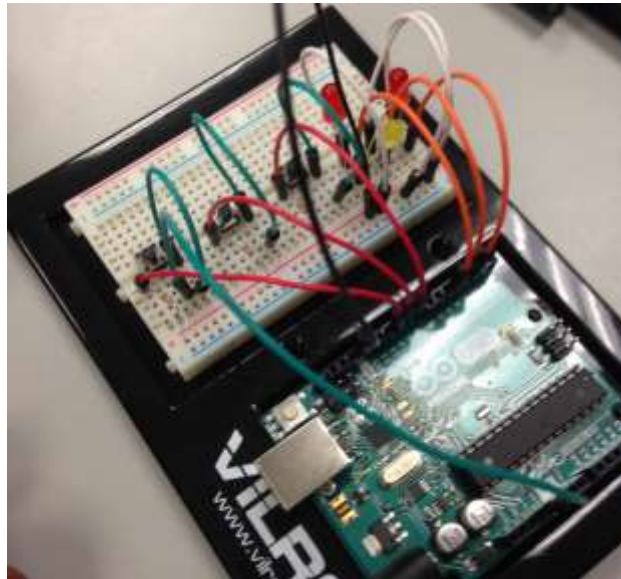


Figure 32 – An example Breadboard circuit

The BitBlox allowed for more variation since there were different modules the students could use and different ways they could connect them. Figure 33 (a), (b), and (c) show some of the various layouts from the different groups. Group F used many blocks and only

connected a few (Figure 33 (a)). Two of the groups made an island of BitBlox for the buttons and an island for the LEDs (Figure 33 (b)). Other groups used many blocks but snapped them together making a circuit that was more compact and organized (Figure 33 (c)).

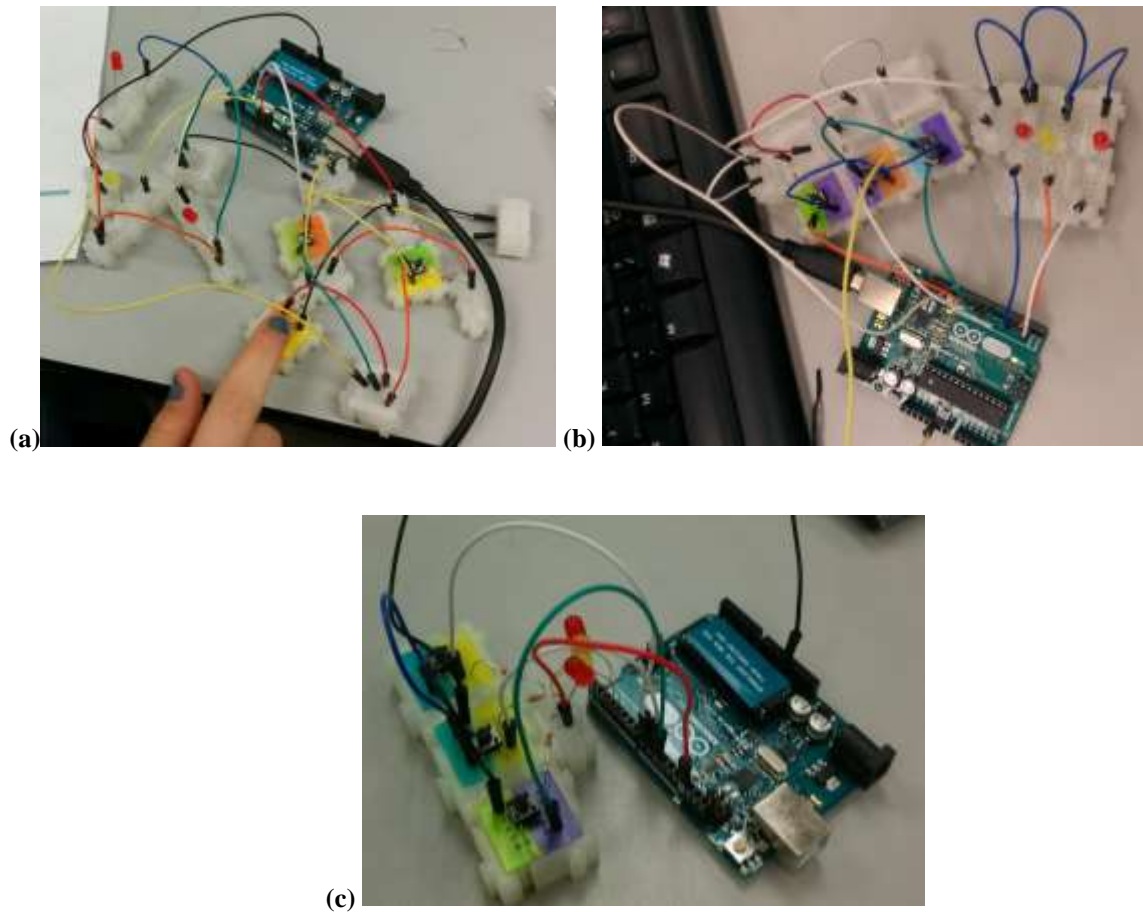


Figure 33 – (a) Group F’s spread out circuit, (b) Group D’s large organized circuit, and (c) Group A’s small compact circuit

5.3.2 Tool Issues

Confusing the Breadboard row and column connections caused issues with all of the students. The researchers observed all 11 pairs mixing up the underlying connection schemes between the row and column. This error persisted throughout the length of the

class, despite students hearing repeated explanations. For example, after running out of contact points to place wires in a column, group B began to group the wires in the columns next to it, as if being close together meant that they were connected. After the Breadboard rows and columns were explained to them, they corrected this error and continued to construct their circuit using the columns properly, but incorrectly extended the column connection scheme to the outer rows of the Breadboard.

Observations showed the BitBlox connections caused confusion in four groups. In all four cases, they tried connecting a component with both sides into one block rather than connecting it across blocks or across colors. The BitBlox enabled many block layout options, which led to decision paralysis in some of the students. For example, Group F was stuck for 10 minutes trying to decide between multiple ways of connecting the ground wire for the components. *“These are all connected, but should they be in one block or separate blocks?”* they asked. Even though they could build it either way, they wanted a straightforward solution. They finally decided to use separate blocks and ended up with the circuit in Figure 33 (a).

5.3.3 Component Errors

The buttons were a pain point for almost all of the pairs in both classes. Because the buttons had four pins, students had trouble understanding their orientation. The students using the Breadboard struggled with connecting the button and then connecting wires to the button. All of the students had the opportunity to hold and examine an example button on the Breadboard enabling them to see how it bridges across two columns. Yet when applying that to their own circuit, many of them confused the *rows* and *columns* and had

difficulties comprehending what happened to the connections when the button bridged the two columns.

Students using the BitBlox also had issues with connecting the button to the block because they confused the orientation. However, the students used the visual color cues on the block to fit it into their circuit properly.

5.3.4 *Terminology*

The majority of the students were novices and learning new circuit terminology. We observed that in both conditions, students were initially confused with terms such as *resistor*, *LED*, and what *positive* and *negative* meant in this context. We observed additional confusion during the Breadboard class with the added terms of row and column. Further, students initially confused the '+' and '-' annotations on the Breadboard with the positive and negative sides of the LEDs. While these annotations did lead to confusion, we also observed that the annotations were a useful way for students to reference parts of their circuit as they were explaining them to researchers.

5.3.5 *Other Circuit Building Issues*

The researchers observed students struggling to transfer connections, such as power and ground, from one point in the circuit to another. At least six groups faced this issue with the Breadboard. This problem occurred at times when students exhausted all holes in a column and needed to connect more wires to it. Only two groups were observed struggling to transfer connections with the BitBlox. No students were observed running out of holes while using the BitBlox, which may be related.

When some students encountered an issue, instead of debugging their circuit, they took it apart and rebuilt it. Of the Breadboard groups four were observed using this technique, as opposed to one Bitblox group. Roughly the same number of groups in both conditions built their circuit color coded to the diagram—i.e. they matched the wire color to the diagram. Observations showed two Breadboard groups tried to map the number of wires to the schematic. This caused them to leave out wires because of the lack of a one-to-one correspondence between what they considered a separate wire in the diagram versus their actual circuit.

5.3.6 Collaboration

Some students disengaged from the circuit building activity, either not paying attention at all or merely observing and not contributing anything. However, this was more pronounced amongst students using the Breadboard. Five groups using the Breadboard had at least one disengaged student as opposed to only one group using the BitBlox.

The Breadboard was more compact, so it was not placed in between the members, but instead resided in front of one person. This made it difficult for one student to see it as the other worked on it. A few groups were struggling with the close connections on the Breadboard. In one pair, the male student said to the female student, “*your fingers are smaller, maybe you can make this work*”.

Students using the BitBlox often had the modules physically placed in front of both people as they worked together. Each member of the pair was not necessarily plugging in wires at the same time, but they could clearly see and engage with what the other student was working on.

5.4 Discussion

The findings from the comparative study between the Breadboard and BitBlox led to interesting design considerations for creating tools for building circuits.

5.4.1 *Simplicity in Design*

While neither connection scheme was flawless, we observed that the BitBlox's simpler connection scheme resulted in fewer user errors. The simpler scheme also meant that fewer terms were needed to describe the tool to the students. This proved beneficial for decreasing the complexity of the learning environment as the students had many other terms to learn and new concepts to understand.

Several groups were observed struggling with the underlying connections of the Breadboard, but far fewer had similar problems with the BitBlox. We attribute this to the simplicity of BitBlox design as it set a lower level of entry for novice students, allowing them to build the same complex circuits that are built on breadboards.

5.4.2 *Affordances for Organization*

Circuitry, even at a basic level, can be difficult to understand. Intermediates and experts become trained over time at understanding complex circuits and are able to use this skill when building and debugging them. To an untrained novice, these tasks can lead to frustrations. We noticed groups took apart their circuit instead of trying to debug it more often in the Breadboard groups than in the BitBlox groups. On both days, some students used the wires to color code their circuits; however, students using the BitBlox could snap together their BitBlox in different ways resulting in different organizations of their circuits.

The differences in organization may help instructors discern students' misconceptions. These findings suggest that affordances for flexible organization within these tools can help students decrease the complexity of their circuits and serve as a physical manifestation of their knowledge about their circuit that instructors can use for providing personalized assistance.

5.4.3 Designing for Appropriation

We observed that the BitBlox gave the students opportunities to create their circuit in many different ways. Instead of using a one size fits all model, the flexibility in the tool allows for greater diversity in learning approaches. This can empower learners by giving them greater control over their learning environment (Fischer, 2013). However, the limitations of useful flexibility should be further explored. It is possible that some students may end up more confused and hesitant about building circuits due to the extra degree of freedom. Without guidance, students may construct circuits that they themselves find difficult to debug making it harder for them to get assistance (see Figure 33 (a)).

5.4.4 Including Identifiers

We observed that annotations on the Breadboard caused confusion for some students, but it also served as a useful tool for students to describe sections of their circuit. This suggests that having identifiers on the tool that allow students to reference their circuit may be useful for collaborating with other students and the instructor.

5.4.5 Size of Tool

One of the main differences between the BitBlox and the Breadboard is the size of the circuit that students produce. This affected how easily students could work on their circuit and how they collaborated within their pairs. Consistent with research in collaborative learning, creating a larger tool allows multiple access points for students to work more seamlessly together (Rick, 2012). Therefore, the number of students to be working together should be considered when determining the size for a tool in this space.

5.5 Contributions and Limitations

We presented a pilot study of BitBlox, a new tool for building circuits in a classroom. The study provided insight to the following research questions:

***RQ2.** What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?

***RQ3.** How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability in individual and group settings?

The study offers three main contributions. First, the findings identified difficulties associated with understanding how the tools worked based on the design choices embedded within the tools. Second, the study provided an understanding of how design characteristics impacted the social interactions between participants in the learning environment. Third, we highlight benefits (i.e. organization and appropriation) and draw backs (i.e. increased

decisions and confusion) to added flexibility within the prototyping tools. We found it useful to consider the simplicity of the design, affordances for organization, affordances for appropriation, identifiers for referencing, and the size of the tool. We hypothesized that the increased complexity of the Breadboard causes a higher strain on the students' working memory. Throughout our investigation we found a greater confusion with the underlying design and terminology associated with the Breadboard in comparison to the BitBlox. This suggests that a further investigation of the cognitive load theory within this domain could prove useful.

The study was limited both in its time and our ability to capture everything that was occurring in the learning environment. The class was only one day, and while we did capture observations we were still limited by what we saw and recorded since we did not have video of the participants interactions. Furthermore, we could identify what the students were doing, but we had little understanding into what they were thinking. We use the laboratory study in the next chapter to complement the findings from this classroom study and offer further insight into the design of tools for physical computing.

CHAPTER 6. ARDUINO LABORATORY STUDY OF PHYSICAL COMPUTING TOOLS

6.1 Introduction

The prior investigation demonstrated both benefits and drawbacks of the designs within the Breadboard and BitBlox prototyping tools in the comparative classroom study (CHAPTER 5) (DesPortes, Anupam, et al., 2016). However, the previous study design was not able to capture the individual students' processes and experiences. The laboratory study of physical computing tools was designed to capture these experiences, investigating novice students working with the Arduino (Figure 34) for the first time. Although the Arduino has been critiqued for not being a suitable for beginners (Blikstein, 2015) it is one of the few tools that offers opportunities for the participants to engage conceptually with the electronics. Part of the contributions of this study is to understand the difficulties beginners face and how we improve the scaffolding in the hardware instead of black-boxing it and making it inaccessible. Specifically, this study was created to tease apart differences in how students are thinking about and using prototyping tools with different affordances, while gathering information on the mistakes, errors and misconceptions that arise. The study explores research questions RQ2 and RQ3 below:

***RQ2.** What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?
- 2.4 What are novice students' reflections on working with the Arduino?

***RQ3.** How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability in individual and group settings?
- 3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?



Figure 34 – Arduino Open Source Prototyping Platform




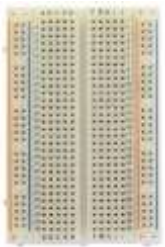

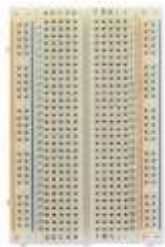

6.2 Study Design

The study comprised of two sections of novice participants learning about and working with the Arduino: a think-aloud section and a non-think-aloud section. The think-aloud section comprised of 15 participants and was designed to provide rich qualitative data capturing how novices conceptually approached the material and their processes, mistakes and errors as they worked. The non-think-aloud section comprised of 30 participants and was designed to provide quantitative insights on the impact of the tools used on participants' self-efficacy and knowledge in conjunction with providing a larger data set of the participants processes, mistakes and errors.

For the quantitative analysis, the non-think-aloud section was a within and between-subjects design with a pre-test and post-test that gathered data on self-efficacy and

knowledge. The participants within each section were split into three groups based on rolling admission to the study. Each group used a different combination of hardware tools (see Table 15). The three types of tools used were: circuit modules, BitBlox, and the standard Breadboard.

Table 15 – Tools Used in Each Group

	<i>Group #1</i>	<i>Group #2</i>	<i>Group #3</i>
<i>First</i>	<p style="text-align: center;">Circuit Modules</p> <div style="display: flex; justify-content: space-around;">   </div> <p style="text-align: center;">Static Components Variable Components</p>	<p style="text-align: center;">BitBlox</p> 	<p style="text-align: center;">Breadboard</p> 
<i>Second</i>	<p style="text-align: center;">BitBlox</p> 	<p style="text-align: center;">Breadboard</p> 	<p style="text-align: center;">BitBlox</p> 

The circuit modules were intended to decrease the amount of problems the students would run into. The static modules have the resistor and LED soldered to it, and the user just needs to plug in the power and ground. The variable modules are similar, but instead of soldered components there are pin headers and labels for where the LED and resistor should be placed. The BitBlox and Breadboard are more complex in their design. They require participants to understand how to make connections between the components using the connection schemes built into the tools. They each have a set of holes that are

electrically connected in specific orientations; by plugging the end of two components into holes that are connected the user connects the components—ex. plugging the positive end of a battery and an LED into holes that are connected. The connection scheme for the BitBlox and Breadboard can be seen in Figure 35 and Figure 36.

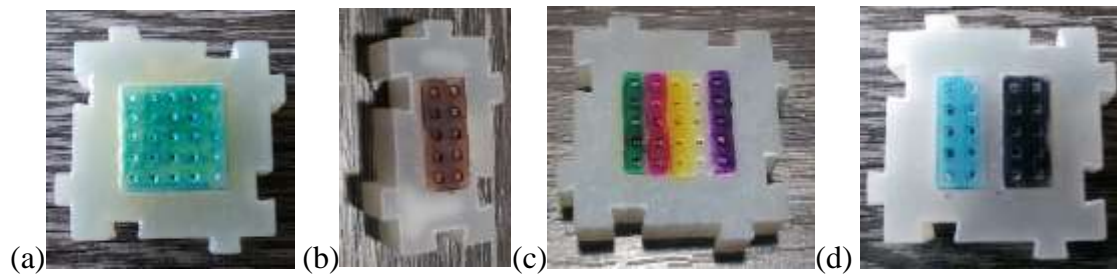


Figure 35 – BitBlox modules with connection scheme conveyed through coloring: (a) and (b) have all the holes connected to one another, (c) has five separate columns of connected holes, and (d) has two separate sections of connected holes

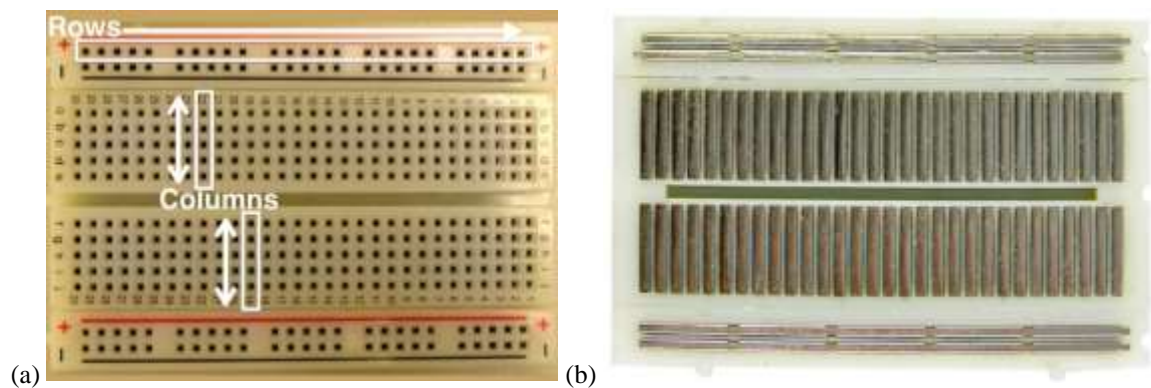


Figure 36 – The standard breadboard has two types of connection schemes: every hole in each row is interconnected and every hole within each column is interconnected—(a) shows the top of the breadboard with rectangles around select rows and columns of connected holes and, (b) the bottom of the breadboard illustrating metal connections in the rows and columns

Each group used two tools: Group 1 started with circuit modules then transitioned to BitBlox; Group 2 started with BitBlox then transitioned to the Breadboard, and Group 3 started with the Breadboard, then then transitioned to BitBlox. The hardware tools served

as the independent variable while the self-efficacy and knowledge tests served as the dependent variable. Having the participants use two tools provided an understanding of how their knowledge transferred between the tools and data on how use of one tool might impact use of another.

The software that was used was a blocks-based environment called Modkit. This software IDE (integrated development environment) was chosen because of the affordances prior literature has identified for beginners working with blocks-based environments (see (Bau, Gray, Kelleher, Sheldon, & Turbak, 2017) for short review). An image of the coding software can be seen in Figure 37.

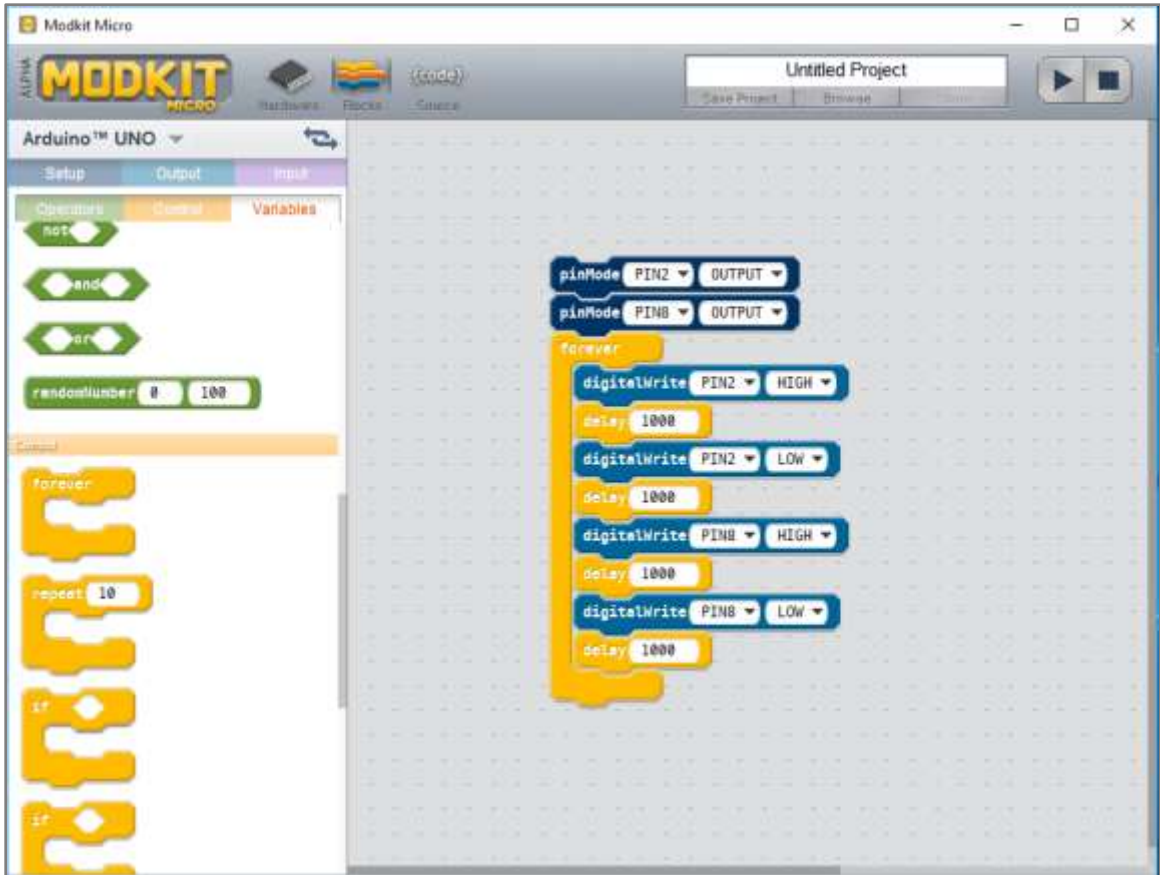


Figure 37 – Screenshot of the Modkit Blocks-Based Integrated Development Environment

6.3 Study Procedures

6.3.1 Study Protocol

The participants from both the think-aloud and non-think-aloud section went through a similar procedure that lasted roughly 2hrs per participant. The participants were guided through the study with a PDF document that walked them through the information and activities that are outlined in Table 16. The researcher first introduced the study to the participant and discussed any questions or concerns about the consent document and data collection methods. In order to preserve consistency across participants, the researcher then

told the participants to try and complete the study without asking the researcher any questions. The document contained the links to the pre-tests and post-tests, the information for the participants to learn from, as well as the instructions for all of the activities and tasks that the participants needed to complete. The group assigned to the participant designated the different tools he/she would use throughout the procedure. The differences in the tool use for each group can be seen in Table 16. The steps without a tool designation means that there was consistency across groups during these parts of the procedure.

Table 16 – Arduino Laboratory Study Protocol

1. Consent		
2. Pre-Test: Self-Efficacy		
3. Pre-Test: Knowledge		
4. Electronics Review with LED Circuit		
5. Introduction to the Arduino and Modkit (IDE)		
6. Guided Blinky LED and Intro to Hardware Tool (Figure 38)		
Group 1: Plug-and-Play Static Module	Group 2: BitBlox	Group 3: Breadboard
7. Task #1: Blinking Two LEDs (Table 17 and Figure 48)		
Group 1: Plug-and-Play Variable Module	Group 2: BitBlox	Group 3: Breadboard
8. Introduction to New Prototyping Tool		
Group 1: BitBlox	Group 2: Breadboard	Group 3: BitBlox
9. Task #2: Two LED Circuit with New Prototyping Tool (Table 17 and Figure 49)		
Group 1: BitBlox	Group 2: Breadboard	Group 3: BitBlox
10. Post-Test: Self-Efficacy		
11. Post-Test: Knowledge		
12. Post-Study Debrief Interview		
13. Demographics Survey		
14. Compensation and Wrap-Up		

All participants started off with taking the pre-test of self-efficacy and knowledge. When they had completed the pre-tests, they could begin learning about physical computing. The participants were first given a refresher to basic high school electronics covering how electricity flows through a circuit and the functions of the battery, LED, and resistor. As part of the refresher, participants were walked through creating a circuit with

the battery, LED, and resistor using alligator clips. Next, the participants were introduced to the Arduino and Modkit IDE and guided through creating *Blinky LED*, which is a beginner activity for the Arduino that consists of creating the circuit and code to blink an LED forever. Figure 38 shows a diagram of the circuit and code needed to successfully complete the activity.

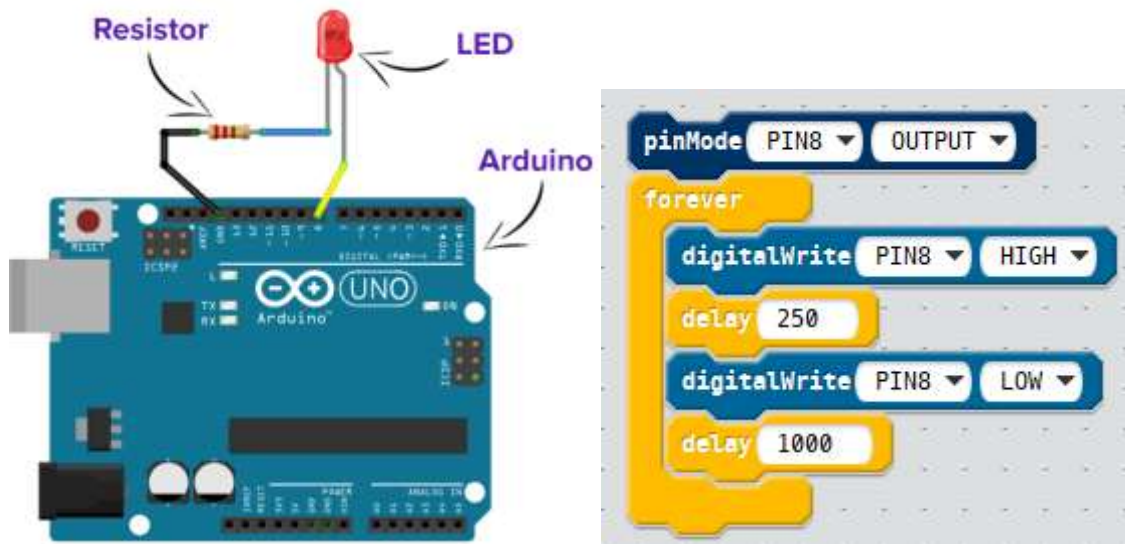


Figure 38 – Blinky LED Arduino circuit (left) and Modkit code (right)

In this section, the participants were walked through how to create the code with Modkit, and how to use their prototyping tool with the Arduino to make the circuit. For the software, participants were stepped through: using and initializing pins with the *pinMode* block; using the *forever* block; how to send digital signals with the *digitalWrite* block; and the use of the *delay* block. For the hardware, Group 1 was given a brief description of the Plug-and-Play module with static components, which only required them to plug in the power and ground of the module correctly into the Arduino power and ground ports; Group 2 was taught how to use the BitBlox and then walked through connecting each component in the

circuit; and Group 3 was taught how to use the Breadboard and then walked through connecting each component in the circuit. The circuits built with each tool can be seen in Figure 39 below.

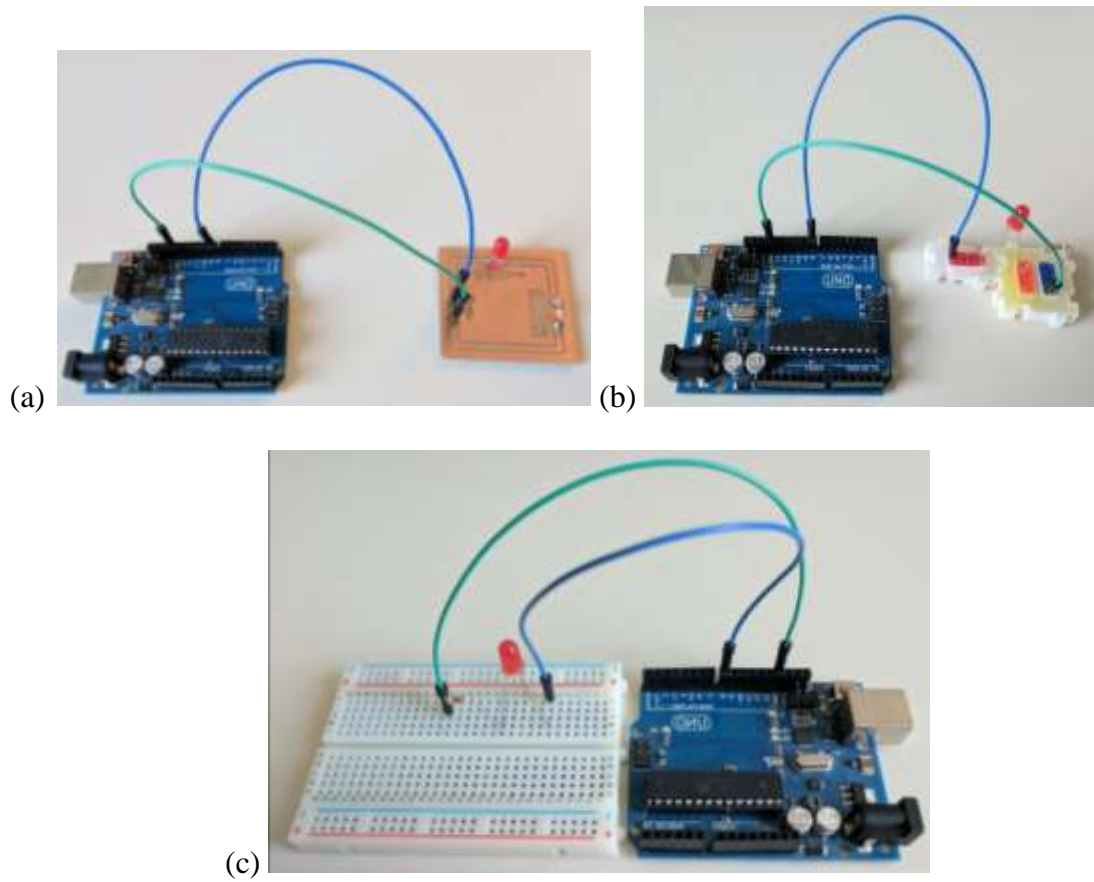


Figure 39 -- Blinky LED Circuits for (a) Group 1: Plug-and-Play Module with Static Components, (b) Group 2: BitBlox, and (c) Group 3: Breadboard

Upon completing the Blinky LED activity, they moved onto the Tasks section of the study.

The tasks section was composed of two unguided tasks that the participants had to complete. The two tasks are outlined below in Table 17.

Table 17 – Study Tasks

Task #1: Blinking Two LEDs

Create two LED/Resistor circuits and hook them up to two different pins. Create a sequence of code to blink one of the LEDs twice then blink the second LED once. Repeat the entire sequence over and over again forever. The code should have the following effect:

Blink LED 1 -> Blink LED 1 -> Blink LED 2 ->

Blink LED 1 -> Blink LED 1 -> Blink LED 2 -> (continue forever).

Task #2: Two LED Circuit with New Prototyping Tool

You are going to build the circuit to the right. Instead of just one LED at pin 3 you're going to hook up two LEDs with a resistor to pin 3. Go ahead and create the circuit and modify your code appropriately to make the LEDs continuously blink forever.

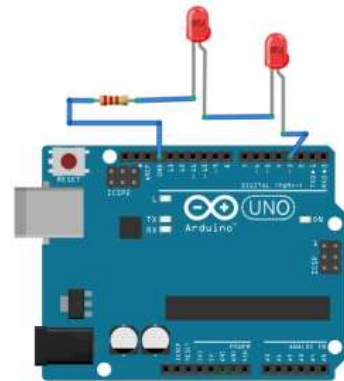


Figure 40 – Plug-and-Play Circuit Module with Variable Components used by Group 1 for Task #1. The pin headers provide positions for the LED, resistor, power and ground to be connected.

The second task was intended to be easier from a software perspective but required all the participants to learn a new prototyping tool. Group 1 transitioned from Circuit Modules to BitBlox, Group 2 transitioned from BitBlox to the Breadboard, and Group 3 transitioned from the Breadboard to BitBlox. When the tasks were complete, or the time was up (in the case of the non-think-aloud section), the participants moved onto take the post-test of self-efficacy and knowledge. Then a post-study debrief interview was then conducted to capture the students' reflective thoughts on the experience and the tools that they used (explained in more depth in section 6.4.4). At the end of the debrief interview the study was wrapped up with the participants completing a demographics survey and receiving compensation for the time they spent in the study. The think-aloud participants all received a \$30 Amazon gift card, and the non-think-aloud participants all received a \$20 Amazon gift card for their participation.

6.3.2 Protocols for Providing Hints to Participants

The participants inevitably had questions or problems that hindered their ability to continue through a task or activity as they completed the study. If this occurred, the participant would get a hint. The hints were handled differently for the think-aloud section and non-think-aloud section. In the think-aloud section, we wanted to ensure that all the participants made it through all of the activities since there were a limited number, and we wanted to understand their thoughts throughout the different sections of the procedure. In this section, the researcher followed a consistent protocol (outlined in Figure 41) for giving participants hints to help them get through the study. Based on specific student issues and needs, there were some variations. For example, if the student had multiple issues and they were running low on time the researcher might identify a specific problem rather than going

through all the steps. Since this section was used for its descriptive power, the need for consistency was less strict. Providing hints based on the participant’s specific situation did not inhibit our ability to understand what the participants were thinking and, in some cases, provided insight based on how they responded to the hints.

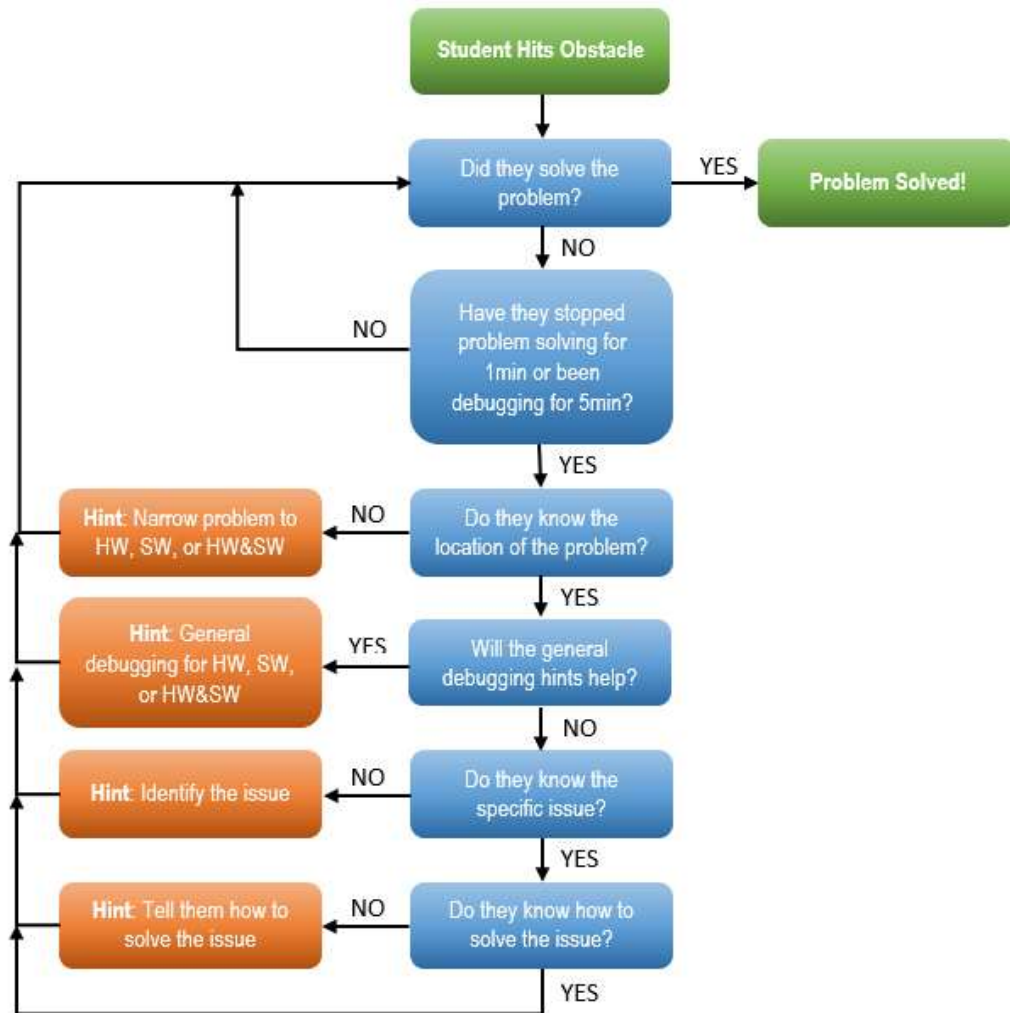


Figure 41 – Protocol for hints in Think-Aloud Section

In the non-think-aloud section, a more consistent protocol needed to be followed to ensure that biases from the researcher in how they applied the protocol and inevitable differences in how certain things were explained did not taint the data. We therefore

changed the method for distributing hints within the non-think-aloud section. The researcher created four sets of hints based on the errors that the participants in the think-aloud group faced: (1) hardware debugging hints, (2) software debugging hints, (3) Task #1 hints, and (4) Task #2 hints. The hardware and software hints each consisted of a sheet of debugging checks the participants received all at once. The hardware hints were composed of checks that the participant could conduct on their circuit to see if they had an unexpected bug, and the software hints were a similar set of checks but for the code. The hardware hints consisted of hints such as a reminder to verify the correct Arduino pin was being used, or a reminder to ensure their electronic components are pushed all the way down in the hardware tools. The software hints consisted of checks such as verifying the correct pin was being referenced in their code blocks, or a reminder to ensure that all their blocks were connected properly in the code.

The task hints were slightly different, as they were designed to help the participant achieve the task. The task hints were distributed one at a time and successively explained how to complete the task. For example, the first hint for Task #1 was: *You must create two circuits each with a different digital pin on the Arduino and each containing an LED and resistor. Make sure that each circuit is connected to ground, and remember, it doesn't matter which ground you use.* The final hint for each task described how to connect the electronic components and how to create the code to complete the task. For example, the fourth and final hint for Task #1:

To complete the task, you should do the following:

- *Create two circuits each with a different digital pin on the Arduino:*

- *You should have an LED and a resistor in each circuit with one of the ends connected to a digital pin and the other side of the circuit connected to a ground connection.*
- *In your code:*
 - *Setup two pinMode commands, one for each of the digital pins you are using with your circuits. Set each of the pins as an OUTPUT*
 - *Use a forever loop in which you will put the following code:*
 - *To blink the first LED setup code that sends a digitalWrite HIGH, delay, digitalWrite LOW, delay (then do it again) digitalWrite HIGH, delay, digitalWrite LOW, delay. By doing it twice it will make it blink twice*
 - *Then blink the second LED by sending a digitalWrite HIGH, delay, digitalWrite LOW, delay. You only need to blink the second LED once so you don't need to do this more than once*
 - *Make sure that the code above is all contained within the one forever loop*

The hints were intended to be agnostic to the prototyping tool the participant was using, so no pictures were used. If the participants had difficulty understanding the hints, using their tool, or had an error in the code or circuit that they could not identify, they might not be able to complete the task, even with all of the hints.

At the beginning of the study, the researcher introduced the participants to the hints for debugging and for the tasks. The researcher conveyed that they would need to request a hint when they wanted one, otherwise they would not receive one. The guide also introduced the participants to the hints, reiterating what the researcher had already gone over. As the participant transitioned from the guided activities (Blinky LED) to the Tasks section of the procedure, the guide reminded the participant of the hints. If the participant asked the researcher a question, the researcher would remind them that they are not allowed to answer the participant's questions and informed them that there were debugging and task hints they could use. If the participant was stuck and told the researcher they did not know what to do, the researcher would again remind them that they had hints that they

could ask for and that they could try re-examining the hints they had already received to solve their problems.

The one exception to this procedure was when the participant created a short between power and ground. The researcher would tell them they needed to remove a connection because they were creating a short circuit by connecting power and ground. The participants did not necessarily understand what a short was but they did understand that the connection was incorrect. This was done to ensure the Arduino was not damaged.

6.3.3 *Protocol for Concurrent Think-Aloud*

The 15 participants who completed the study using a concurrent think-aloud protocol (Ericsson & Simon, 1980) were intended to provide a better idea of what novices were thinking when presented with the tools and concepts in a physical computing task for the first time. Concurrent think-aloud protocols require the participant to verbalize their thoughts as they are working on a task (Ericsson & Simon, 1980). This method was chosen over a retrospective think-aloud (Ericsson & Simon, 1980) because previous research suggests there is more accurate *action information* about why an action was taken and how the participant was thinking about the task (van den Haak, De Jong, & Jan Schellens, 2003; van Gog, Paas, van Merriënboer, & Witte, 2005).

The think-aloud was introduced to the participants at the beginning of the experiment. Ericsson et al. suggest using “warm-up” exercises in order to introduce the process to the participants and get them used to thinking out loud (Ericsson et al., 1993). For the sake of time, we had the participants practice on the “Review of Electronics” section since the other groups went through this material as well and it proceeded the

physical computing activities. In this section, they reviewed the basics of electronics and built a resistor-LED circuit with a battery using alligator clips. At the conclusion of the activity the researcher provided feedback to the participants, which often consisted of instruction to increase their volume, and/or if possible, try to vocalize more of their thoughts as they are going through the information and activities.

6.3.4 Collecting Video and Audio Data

Throughout both sections of the study, a total of about 85hrs was recorded of both video and audio data across all 45 participants. A web camera was mounted to shoot directly down on the workspace and linked to a screen recording of the laptop the participants used (see Figure 42). To ensure we were able to capture what was going on with the small components, we also setup a backup camera that focused on the workspace from a different angle. The backup recording was used in instances when we could not decipher what was happening from the webcam footage that was captured. Since the non-think-aloud section did not have the added audio to contextualize the participants' actions with the hardware components, the researcher recorded notes throughout each of the studies to log the participants' actions with the hardware in conjunction with the status of where there were issues in the code and circuit to contextualize the notes.

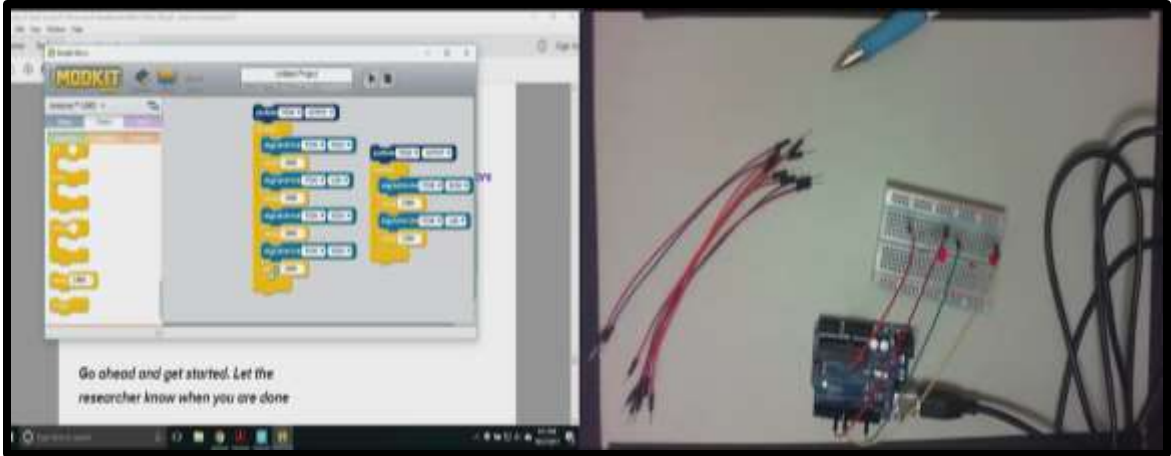


Figure 42 – Example of Screen Recording with Webcam Capture

6.4 Study Materials

The study materials were developed, tested and refined through observation and analysis of three pilot tests of graduate students who were novices in physical computing. The graduate students completed the study using a think-aloud protocol. At the end of each participant's study we reviewed the materials with the participant to understand where the materials could be improved. The instructions, procedures, and questions were improved after each pilot test. The changes included things such as, reordering the content, rewording the tasks, adding information they were confused about, and expanding and labeling the pictures to improve their readability. After the last changes were made, another novice graduate student reviewed the materials to ensure the clarity and catch any last improvements that should be made.

After the initial set of 15 think-aloud participants, slight modifications were made again to improve areas in which the materials caused confusion. For example, we extended the positive leg of the LED in any diagram in which it appeared to ensure participants could

read it correctly. We also found that our explanation of the delay block caused misconceptions within the participants of how the code was causing the LED to blink (more on this in section 6.6.6.3), so we reworded it to improve the understanding within the next set of participants.

6.4.1 Pre-Study Screening Survey

In order to ensure the participants were novices, students who wanted to participate had to first complete a pre-screening survey (Appendix: A.1 Pre-Screening Survey). We then excluded students who: had taken a college electronics course, had taken more than one high school electronics course, had taken a high school or college CS course, rated their expertise in CS higher than three on a nine-point Likert scale, rated their expertise in electronics as higher than six on a nine-point Likert scale, rated their expertise in physical computing as higher than three on a nine-point Likert scale, had used a microcontroller before, or have had informal experience indicating they have legitimate expertise in CS, electronics or physical computing (ex. created a cell phone app, participated in FIRST robotics, etc.). We were originally also going to use the AP physics test as a way to exclude students, and we were going to exclude those who rated their electronics expertise higher than a 3; however, with those criteria we were left with few participants who could participate so we expanded the selection criteria by getting rid of the requirement around AP physics and raising the allowable expertise for electronics to a five.

6.4.2 *Tests of Self-Efficacy*

In order to understand how the tools might affect one's motivation to work with physical computing in the future, I measured the participant's self-efficacy in the pre-test and post-test. Self-efficacy links one's perceived competence in a particular domain to perform certain tasks and overcome certain challenges (Bandura, 2006). As Bandura states, *"Unless people believe they can produce desired effects by their actions, they have little incentive to undertake activities or to persevere in the face of difficulties."* (Bandura, 1994, p. p1). The self-efficacy test covered three domains: programming, electronics, and physical computing. For programming self-efficacy, I adapted four questions from Ramalingam and Wiedenbeck's computer programming self-efficacy scale and created three more questions mirroring the structure but changing what the questions asked about based on skills learned in the study. I applied a similar method to create seven questions to measure electronics self-efficacy and seven questions to measure physical computing self-efficacy. The test can be found in Appendix A.2 Pre-/Post-Test of Self-Efficacy and Knowledge.

6.4.3 *Tests of Knowledge*

In order to understand what the students were learning throughout the study and how the tools may have impacted this, I measured the participants' knowledge in the pre-test and post-test. The participants took the test of knowledge directly after the test of self-efficacy in both the pre-test and post-test. This order was chosen because it was more important to understand the effect of the participants' experience in the study than the effect of their perceived achievement on the knowledge test. For the first three questions in

knowledge test, I used Osborne's (1983) method of drawing a complete circuit using a power source and electronic components. This was used because of its prior use in the study assessing participants' knowledge of the electronics concepts students acquired during an E-Textiles program (Pepler & Glosso, 2013). We developed three questions using this method—two questions using the Arduino and one question using a battery as the voltage source. Since there was no validated test for use with the Arduino, I created a 9-question multiple choice test—three questions that require analysis of a circuit, three that require analysis of code, and three that require analysis of a circuit and code together. The test was designed to understand participants ability to analyze forever loops, the effect of code on a circuit, delays in various locations, pin numbers within the write functions, polarity of the LEDs, sequential reasoning in electronics and the effect of using multiple resistances. The test can be found in Appendix A.2 Pre-/Post-Test of Self-Efficacy and Knowledge.

6.4.4 Post-Study Debrief Interviews

In order, to capture the participants' reflective thoughts on the experience, a short post-study debrief interview was conducted. All of the think-aloud participants answered the first four questions of the set outlined below. The questions were intended for the participants to reflect on their experience and the tools that they used. The questions were:

1. What did you think about this experience?
2. What did you think about the tools you used?
3. What did you find difficult as you were going through?
4. What did you find easy as you were going through?
5. How would you feel about working with the Arduino again?

The non-think-aloud section first answered two extra test-like questions that were intended to probe into how the participants were thinking about the physical computing tasks. Then, they answered all five of the questions above. The first test-like question asked them to describe how they thought they would need to setup the circuit and code for an LED to blink twice when a button was pushed. The second question required the student to analyze a broken Arduino circuit and program in order to identify the mistakes in the circuit and the code. The participants in the non-think-aloud section only completed the post-study interview if there was time. Since participants were not helped through completing the task beyond the hints, the tasks sometimes took up a greater portion of the study time in the non-think-aloud section.

6.5 Participants

A total of 48 participants were recruited for the study: 15 for the think-aloud section and 33 for the non-think-aloud section. Two participants (one from the non-think-aloud section in group 1 and one from group 2), had to have their data thrown out because the researcher made an error during the participant's study. The researcher intervened when they should not have providing information that could have compromised the data for comparison across groups. This left us with 31 participants in the non-think-aloud section—Group 1: 10 participants, Group 2: 10 participants, and Group 3: 11 participants.

Participants were recruited from eight universities: Georgia Institute of Technology (GT), Georgia State University (GSU), Agnes Scott College (ASC), Emory University (EU), University of Colorado Boulder (CUB), Columbia University (CU), University of Florida (UF) and University of Nebraska Omaha (UNO). A range of universities were

recruited from because of the difficulty of finding novices at the college level that met our requirements and would sign up for the study.

We first recruited for the think-aloud section which consisted of participants from GT and GSU (*see* Table 18) and then recruited the non-think-aloud participants, which had participants from the various universities spread out throughout the three groups (*see* Table 19). Within the think-aloud and non-think-aloud sections, participants were split into the three groups based on a rolling admission into the study.

Table 18 – Participants by College in Each Group for Think-Aloud

	<i>GT</i>	<i>GSU</i>	<i>Female</i>	<i>Male</i>
<i>Group 1</i>	3	2	5	0
<i>Group 2</i>	3	2	2	3
<i>Group 3</i>	3	2	4	1
<i>Total</i>	9	6	11	4

Table 19 – Participants by College in Each Group for Non-Think-Aloud

	<i>GT</i>	<i>GSU</i>	<i>ASC</i>	<i>EU</i>	<i>CUB</i>	<i>CU</i>	<i>UNO</i>	<i>UF</i>
<i>Group 1</i>	2	1	0	0	1	3	1	2
<i>Group 2</i>	1	0	1	0	1	3	1	3
<i>Group 3</i>	1	0	1	1	2	2	2	2
<i>Total</i>	4	1	2	1	4	8	4	7

The demographics for the groups were also collected in order to understand how similar the groups were. Overall, there was an over representation of women, with a mixed racial demographic in each group, and an age range of 18-32 (see Table 20).

Table 20 – Demographic Makeup of Groups

	<i>Num. Participants</i>	<i>Gender</i>			<i>Race/Ethnicity</i>					<i>Age</i>
		M	F	O	B	W	L	A	M	Range
<i>Group 1</i>	10	1	9	0	2	6	0	2	0	19-32
<i>Group 2</i>	10	2	8	0	1	3	1	4	1	18-25
<i>Group 3</i>	11	3	7	1	1	5	0	4	1	18-26
<i>Total</i>	31	6	24	1	4	14	1	10	2	18-32

Gender: M=Male, F=Female, O=Gender Fluid
Race/Ethnicity: B=Black/African American, W=White/Caucasian,
L= Latinx/Hispanic, A=Asian, M=Multi-Race

6.6 Data Analysis and Findings

The data from the think-aloud and non-think-aloud sections complement each other and therefore will be presented alongside one another and at points together where it makes sense. I will first go through and discuss the tests of self-efficacy and knowledge and my findings from the statistical analysis between the groups. The data from this section will provide a high-level overview of the groups from which the qualitative data can be situated. I then go over the completion rates between the groups in the non-think-aloud section highlighting how participants used the hints. Next, I dive into a discussion on the qualitative coding of the participants experiences during the study. The findings conclude

with a discussion of the insights from the post-study interviews where participants reflected on their experiences.

6.6.1 Tests of Self-Efficacy and Knowledge

The Self-Efficacy test was composed of 21 Likert Scale questions gauging participants self-efficacy within programming, electronics, and physical computing. The total points within the self-efficacy test was 145 points for the think-aloud section and 147 for the non-think-aloud section. For the think-aloud section, 20 of the questions were on a 7-point scale, but one question was accidentally clipped to a 5-point scale. Since the mistake was on both the pre-test and post-test, we were not concerned with this mistake, which was corrected for the non-think-aloud section. The knowledge tests were out of 15 points and had the participants drawing circuits, analyzing diagrams of circuits, and analyzing code. The first three questions in which the participants drew diagrams were worth two points while the other 12 multiple choice questions were all worth one point. The data from the self-efficacy and knowledge tests were collected twice: once directly after the introduction and consent procedures and again after the participants had finished going through the tasks section. While the data from both the think-aloud and non-think-aloud sections provided a source to contextualize and triangulate the qualitative data, the data from the non-think-aloud section was also statistically analyzed to see if there was a significant impact on learning gains and self-efficacy in relation to the tools used.

6.6.1.1 Pretest and Posttest Data from Think-Aloud Participants

In the think-aloud section, all three groups demonstrated an increase in both their self-efficacy and knowledge (see Figure 43 and Figure 44).

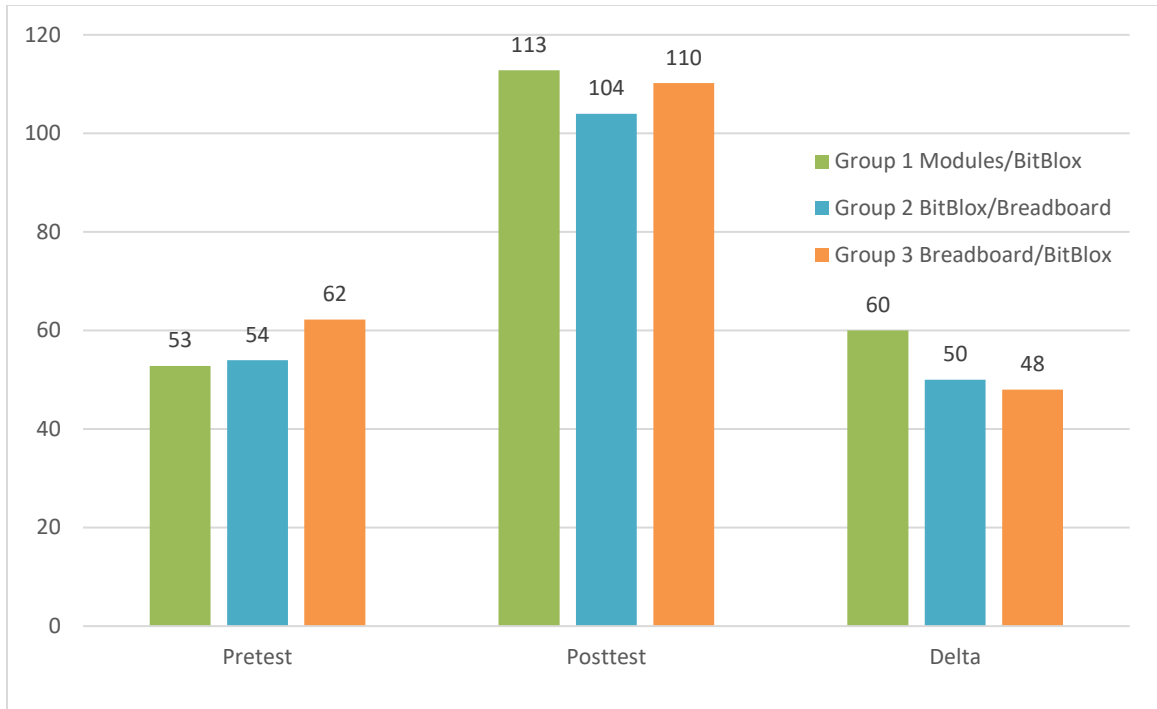


Figure 43 – Average Self-Efficacy scores of Think-Aloud participants by group out of 145 points



Figure 44 – Average Knowledge test scores of Think-Aloud participants by group in a 15-point test

While Group 3 (Breadboard/BitBlox) had a slightly higher average pretest self-efficacy score than the other two groups (a score of 62 compared to Group 1 at 52 and Group 2 at 53), Group 3's knowledge scores were about the same with an average score of 2.6 compared to Group 1 and 2's average score of 3. We saw the greatest change in average score of self-efficacy within in Group 1 (Modules/BitBlox) which was 10 points higher than either of the other groups. However, the participants had the greatest change in knowledge in Group 3 (Breadboard/BitBlox). These scores can be used to contextualize and triangulate and understanding of the participants' experiences with the tools based on the qualitative analysis.

6.6.1.2 Pretest and Posttest Data from Non-Think-Aloud Participants

Summary. We conducted a statistical analysis on the data from the participants in the non-think-aloud section as a within and between-subjects design using a Mixed Multivariate Analysis of Variance (MANOVA). The hardware tools used by each of the groups served as the independent variable while self-efficacy and knowledge test scores served as the dependent variables, measured in the pre-test and post-test¹. The statistical analysis found no statistically significant difference between the participants based on their group. The data was also analyzed from the perspective of a between-subjects design using an Analysis of Covariance (ANCOVA). The ANCOVA was run on the data for the post-test knowledge and self-efficacy of the groups using the pre-test scores as a covariate. These tests also did not find a significant difference between the groups. The low number

¹ The repeated measures test fit the comparison of Group 2 and 3 since they were both exposed to the Breadboard and BitBlox however the statistical test did not directly align with the data from Group 1 who used the modules and were not exposed to both of the other two tools.

of participants in each group and an issue identified with the hardware of the modules group both present reasons for these non-significant results. An improved study design is presented in CHAPTER 7, that would be more appropriate for understanding the questions that these statistical analyses were targeted at. Below are the methods and results from the two tests that were run.

MANOVA. The initial descriptive statistics were analyzed and one outlier was identified in the data for Group 3 (Breadboard/BitBlox). The descriptive statistics with the data point can be seen in Table 21.

Table 21 – Descriptive Statistics of Self-Efficacy (SE) and Knowledge (KN) (N=31)

Group		N	Minimum	Maximum	Mean	Median	IQR	Std. Deviation
1	Pre-SE	10	21	90	44.60	36.5	35	22.172
	Post-SE	10	70	140	98.50	93	32	21.183
	Pre-KN	10	0	7	2.70	1.5	4	2.452
	Post-KN	10	5	13	8.70	8.5	5	2.627
2	Pre-SE	10	30	97	53.00	46	36	22.151
	Post-SE	10	59	129	104.20	113	48	25.494
	Pre-KN	10	0	7	3.20	3	5	2.573
	Post-KN	10	4	14	10.30	11	5	3.020
3	Pre-SE	11	26	104	56.73	55	37	22.078
	Post-SE	11	20	139	110.27	114	40	34.252
	Pre-KN	11	1	8	4.18	5	6	2.750
	Post-KN	11	1	15	10.91	11	4	3.754

The participant had a posttest knowledge score of 1 and a posttest self-efficacy score of 20. Her posttest knowledge and self-efficacy was more than 2.5 standard deviations less than the mean in her group. Her data for self-efficacy and knowledge was identified as a minor outlier (major outliers were not possible in this data set, see Table 22).

Table 22 – Outlier Calculations

GROUP 3	Q₁	Q₃	IQR	MINOR OUTLIER Q₁-(IQR*1.5)	MAJOR OUTLIER Q₁-(IQR*3)
POST-SE	10.5	13	4	4.5	-1.5 (min=0)
POST-KN	98.5	134.5	40	38.5	-21.5(min=0)

With the outlier data removed we had the following descriptive statistics in Table 23. The skewness and kurtosis were within the +/-2 acceptable range for the pretests and posttests of knowledge and self-efficacy (see Table 23). Figure 22 and Figure 23 below show the average scores of pre-test and post-test self-efficacy and knowledge scores presented by group.

Table 23 – Descriptive Statistics of Self-Efficacy (SE) and Knowledge (KN) (N=30)

Group	N	Min.	Max.	Mean	Std. Deviation	Skewness		Kurtosis		
						Statistic	Std. Error	Statistic	Std. Error	
1	Pre-SE	10	21	90	44.60	22.172	.972	.687	.254	1.334
	Post-SE	10	70	140	98.50	21.183	.762	.687	.136	1.334
	Pre-KN	10	0	7	2.70	2.452	.829	.687	-.865	1.334
	Post-KN	10	5	13	8.70	2.627	.361	.687	-.920	1.334
2	Pre-SE	10	30	97	53.00	22.151	.871	.687	-.031	1.334
	Post-SE	10	59	129	104.20	25.494	-.679	.687	-1.003	1.334
	Pre-KN	10	0	7	3.20	2.573	.409	.687	-1.326	1.334
	Post-KN	10	4	14	10.30	3.020	-.994	.687	.883	1.334
3	Pre-SE	10	26	104	56.80	23.270	.856	.687	.461	1.334
	Post-SE	10	95	139	119.30	17.538	-.296	.687	-1.767	1.334
	Pre-KN	10	1	8	4.50	2.677	-.369	.687	-1.434	1.334
	Post-KN	10	9	15	11.90	1.912	.296	.687	-.795	1.334

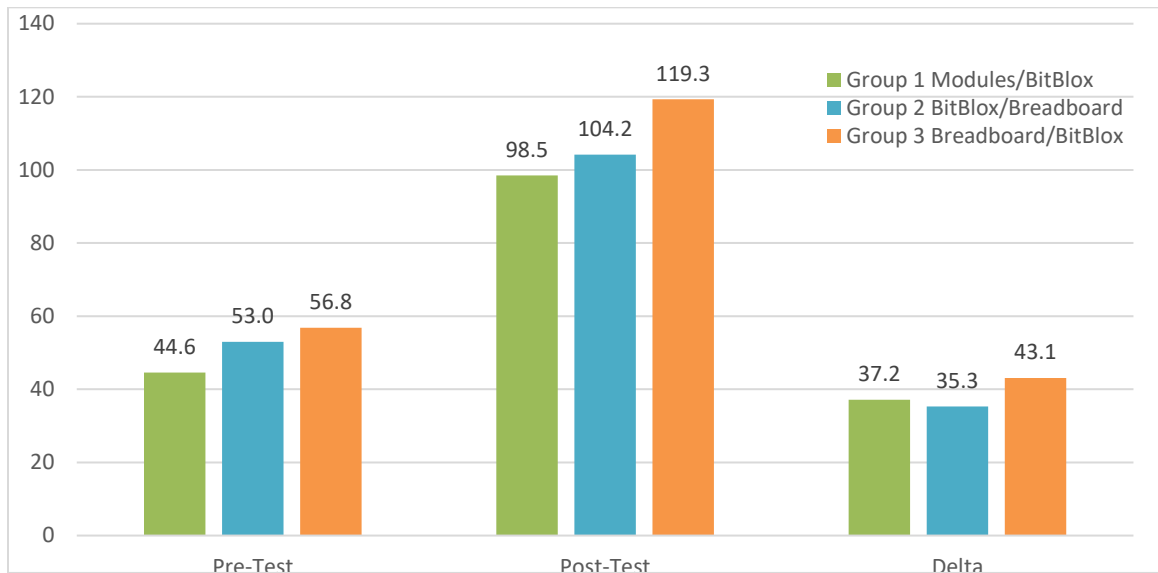


Figure 45 – Average self-efficacy scores of Non-Think-Aloud participants by group out of 147 points

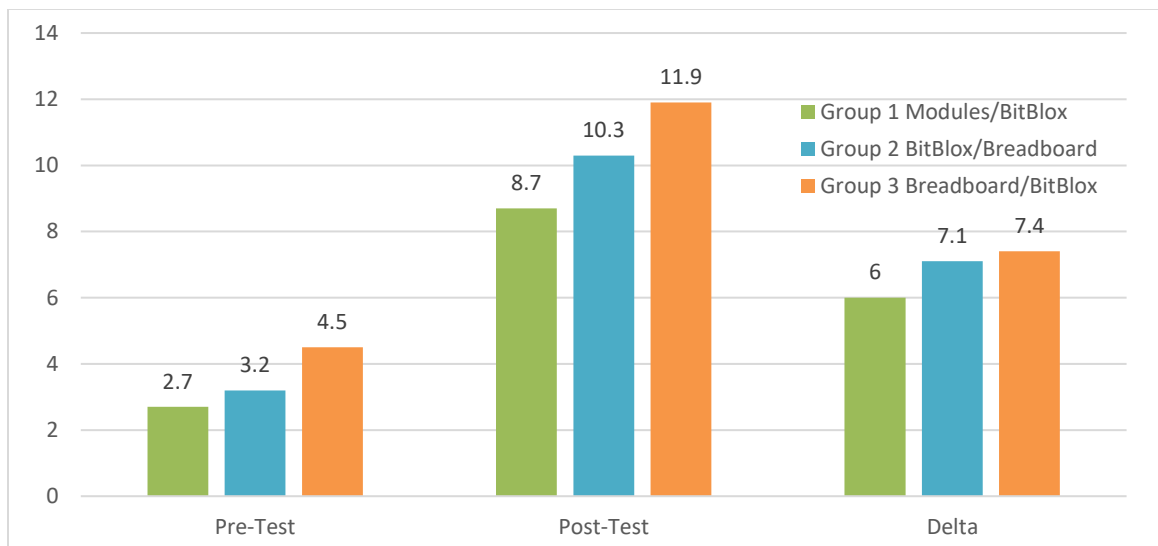


Figure 46 – Average knowledge test scores of Non-Think-Aloud participants by group in a 15-point test

All of the average scores for self-efficacy and knowledge increased between the pre-test and post-test. Group 1 (Modules-BitBlox) entered the study with a slightly lower self-efficacy and knowledge score, followed by Group 2 (BitBlox-Breadboard); Group 3

(Breadboard-BitBlox) had the highest. The post-test results mirror this trend with Group 1 having the lowest overall posttest score on both the knowledge and self-efficacy pretests and Group 3 having the highest. In order to understand if a significant difference between the groups existed, a Multivariate Analysis of Variance (MANOVA) was conducted on the data.

The data was checked for multicollinearity by running Pearson Correlation tests between the pre-test and post-test self-efficacy and knowledge data as reported in Table 24 below. As expected, there is a moderate positive correlation between the pretest self-efficacy and the pretest knowledge scores ($r = 0.421$, $p < 0.05$) and a moderate positive correlation between the posttest self-efficacy and posttest knowledge scores ($r = 0.473$, $p < 0.01$). None of the scores are above 0.9; therefore, there is no multicollinearity.

Table 24 – Correlation Tests Between Dependent Variables

<i>Comparison</i>	<i>Pearson Correlation Results</i>
<i>Pre-SE, Post-SE</i>	Not Significantly Related
<i>Pre-KN, Post-KN</i>	Not Significantly Related
<i>Pre-SE, Post-KN</i>	Not Significantly Related
<i>Pre-SE, Pre-KN</i>	Weak Positive Correlation ($r = 0.421$, $p < 0.05$)
<i>Post-SE, Post-KN</i>	Weak Positive Correlation ($r = 0.473$, $p < 0.01$)

The data was analyzed using a Mixed Multivariate Analysis of Variance (MANOVA) to understand the possible interactions between the pre-test and post-test measurements of the dependent variables of self-efficacy and knowledge, with the independent variable of the tools used by the participant. There was not a significant interaction between groups as shown in the Type III Sum of Squares and Wilks' Lambda in Table 25 and Table 26.

Table 25 – Tests of Between-Subject Effects

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Intercept	223344.408	1	223344.408	637.164	.000
Group	1818.067	2	909.033	2.593	.093
Error	9464.275	27	350.529		

Table 26 – Multivariate Tests

Effect		Value	F	Hypothesis df	Error df	Sig.
Tests	Pillai's Trace	.955	571.895	1.000	27.000	.000
	Wilks' Lambda	.045	571.895	1.000	27.000	.000
	Hotelling's Trace	21.181	571.895	1.000	27.000	.000
	Roy's Largest Root	21.181	571.895	1.000	27.000	.000
Tests * Group	Pillai's Trace	.117	1.788	2.000	27.000	.186
	Wilks' Lambda	.883	1.788	2.000	27.000	.186
	Hotelling's Trace	.132	1.788	2.000	27.000	.186
	Roy's Largest Root	.132	1.788	2.000	27.000	.186
PrePost	Pillai's Trace	.854	157.813	1.000	27.000	.000
	Wilks' Lambda	.146	157.813	1.000	27.000	.000
	Hotelling's Trace	5.845	157.813	1.000	27.000	.000
	Roy's Largest Root	5.845	157.813	1.000	27.000	.000
PrePost * Group	Pillai's Trace	.038	.529	2.000	27.000	.595
	Wilks' Lambda	.962	.529	2.000	27.000	.595
	Hotelling's Trace	.039	.529	2.000	27.000	.595
	Roy's Largest Root	.039	.529	2.000	27.000	.595
Tests * PrePost	Pillai's Trace	.791	102.096	1.000	27.000	.000
	Wilks' Lambda	.209	102.096	1.000	27.000	.000
	Hotelling's Trace	3.781	102.096	1.000	27.000	.000
	Roy's Largest Root	3.781	102.096	1.000	27.000	.000
Tests * PrePost * Group	Pillai's Trace	.032	.442	2.000	27.000	.647
	Wilks' Lambda	.968	.442	2.000	27.000	.647
	Hotelling's Trace	.033	.442	2.000	27.000	.647
	Roy's Largest Root	.033	.442	2.000	27.000	.647

ANCOVA. The data was also analyzed from the perspective of an Analysis of Covariance (ANCOVA) to understand if the posttest knowledge and self-efficacy scores would highlight any differences between groups taking the pretest scores as covariates in the analysis. There was no significant interaction between groups for either the self-efficacy (Table 27) or knowledge (Table 28) as shown in the Type III Sum of Squares below.

Table 27 – Tests of Between-Subjects Effects for Post-Test Self-Efficacy

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	3207.598 ^a	3	1069.199	2.364	.094
Intercept	39155.518	1	39155.518	86.575	.000
Pretest Self-Efficacy	897.131	1	897.131	1.984	.171
Group	1680.651	2	840.326	1.858	.176
Error	11759.069	26	452.272		
Total	360580.000	30			
Corrected Total	14966.667	29			

a. R Squared = .214 (Adjusted R Squared = .124)

Table 28 – Tests of Between-Subjects Effects for Post-Test Knowledge

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	61.099 ^a	3	20.366	3.167	.041
Intercept	892.362	1	892.362	138.764	.000
Pretest Knowledge	9.899	1	9.899	1.539	.226
Group	35.455	2	17.727	2.757	.082
Error	167.201	26	6.431		
Total	3411.000	30			
Corrected Total	228.300	29			

a. R Squared = .268 (Adjusted R Squared = .183)

The results demonstrate that the quantitative analysis was inconclusive; however, analysis of the video and audio data of both the think-aloud and non-think-aloud section provided insights to the research questions. We will first discuss how participants completed the activities in the non-think-aloud section, before presenting the findings from the qualitative coding analysis of the video and audio data.

6.6.2 Completion of Activities Non-Think-Aloud Section

The data from the participants was analyzed for the number of participants that completed the various activities within each group in the non-think-aloud section. While the researcher ensured that participants in the think-aloud section were able to complete the activities by providing hints as needed, this was not true for the non-think-aloud section. The task completion rates along with the average time to completion can be found in Table 29. Since the participants were able to request hints, it is important to understand how the hints relate to the completion rates. The time to completion was averaged across the participants who did complete the activity. The number of participants that used the Task hints, hardware debugging, and software debugging per group is broken up by activity in Table 30.

Table 29 – Task Completion Rates by Group

	<i>Blinky LED</i>		<i>Task #1</i>		<i>Task #2</i>	
	Num. Completed (Attempted)	Avg. Time (min)	Num. Completed (Attempted)	Avg. Time (min)	Num. Completed (Attempted)	Avg. Time (min)
<i>Group 1 (N=10)</i>	<i>Modules</i> 10 (10)	13.05	<i>Modules</i> 9 (10)	30.30	<i>BitBlox</i> 6 (10)	11.37
<i>Group 2 (N=10)</i>	<i>BitBlox</i> 10 (10)	19.33	<i>BitBlox</i> 7 (10)	14.22	<i>Breadboard</i> 7 (8)	6.92
<i>Group 3 (N=11)</i>	<i>Breadboard</i> 11 (11)	20.25	<i>Breadboard</i> 9 (11)	17.89	<i>BitBlox</i> 8 (8)	7.23
Total	31	17.54	24	20.80	21	8.51

Table 30 – Hints Requested by Group

	<i>Blinky LED</i>			<i>Task #1</i>			<i>Task #2</i>				
	N	HW	SW	N	Task	HW	SW	N	Task	HW	SW
<i>Group 1</i>	10	0	0	10	7	7	7	9	5	3	3
<i>Group 2</i>	10	1	0	10	6	3	3	7	0	0	0
<i>Group 3</i>	11	2	2	11	5	2	3	9	0	1	1
Total	31	3	2	31	18	12	13	24	5	4	4

We ensured that all participants were able to complete Blinky LED, however, the rates at which they completed the activity varied. Group 1 participants averaged around 13 min while the other two groups took around 20min. No hints were needed for Group 1 to complete the activity, while one participant in Group 2 and two participants in Group 3 needed debugging hints in order to get through the activity.

In Task #1, all but one of the participants in Group 1 were able to complete the activity; however, they also took the longest time with an average time of about 30 minutes with Groups 2 and 3 averaging about 14 and 18 minutes respectively. Furthermore, just over twice as many participants in Group 1 needed the hardware and software debugging hints than in the other two groups in order to do so. Of the five participants that did not complete Task #1 in Groups 2 and 3, only two received all the hints, one received no hints, and two each only received the first task hint (out of four). Therefore, the higher completion rate in Group 1 for the first task could be attributed to the fact that the participants received more hints.

Task #2 was completed by all of the participants who attempted it in Groups 3 and all but one participant in Group 2, while four of the ten participants in Group 1 were not able to complete the task. Of the Group 1 participants that did complete the task, it took them just under twice as long on average to complete it when compared to the other two groups. This data provides context for the results of what happened in each of the three groups during the various activities, but it does not provide insight into what was going on during the participants' experiences. In the next sections, I present the types and prevalence of the bugs, breakdowns, and obstacles that participants encountered that contributed to these results.

6.6.3 Codebook for Qualitative Analysis

When the study was designed, two coding schemes from prior literature were identified as schemes that might fit our data because of the similar nature of the studies investigating skilled adult participants working with the Arduino in a laboratory setting (Booth &

Stumpf, 2013; Booth et al., 2016). The first coding scheme involves coding of: *obstacles*—where participants needed to overcome hurdles, *breakdowns*—where participants demonstrated errors in their actions or thinking, and *bugs*—where participants introduced a fault into the system (Booth et al., 2016). The second relevant coding scheme was Booth et al.’s application of system Learning Barriers that she derived from Ko et al. (Ko et al., 2004) who defined six categories of learning barriers, which Booth then applied to the Arduino environment (Booth & Stumpf, 2013). To understand whether either of these two schemes would fit the data collected in this study, one researcher coded a third of the data for emergent themes. The researcher then compared the themes she identified to the coding schemes above. A close mapping was identified between the *obstacles/breakdowns/bugs* scheme. The researcher’s codes fit into the original scheme (see Table 31) and provided an opportunity to expand on it by identifying the types of breakdowns the novice participants encountered. The researcher proceeded to use the expanded codebook to code the final two thirds of the data.

Table 31 – Original Obstacles/Breakdowns/Bugs coding scheme from Booth et al.’s study of the Arduino in a laboratory setting (Booth et al., 2016)

Code	Description
<i>Obstacles</i>	Hurdles to overcome: <ol style="list-style-type: none"> 1) <i>Participants stated that they did not know or understand something</i> 2) <i>Participants said they needed to do something but there was evidence that they faced a problem doing it</i> 3) <i>Participants showed signs of confusion or frustration</i>
<i>Breakdowns</i>	Evidence of errors in action or thinking: <ol style="list-style-type: none"> 1) <i>Participants carried out a wrong action making a slip or mistake</i> 2) <i>Participants made an incorrect assessment</i> 3) <i>There were observed faults in their knowledge or reasoning</i>
<i>Bugs</i>	Fault that a participant introduces through their actions or beliefs

To ensure the validity of the coded data, a second researcher contributed to the coding process. Both researchers reviewed the codebook over transcriptions of 5% of the coded segments. The codebook was clarified and refined as well as a process for coding each segment such that each researcher felt it represented the data appropriately and there was consistency in identifying the codes. The process involved going through the following questions for each code segment:

- (1) Is there an *obstacle*?
 - Did they ask a question, say or indicate that something was confusing?
- (2) Is there a *bug* (also) introduced?
 - Did they introduce an error they would have to fix in order to complete the task?
- (3) Is there a *bug* (also) being fixed?
 - Did they change something that fixed a bug that was previously identified?
- (4) Is there a *breakdown* (also)?
 - If a bug was introduced, what incorrect assessment or thought process led to this? Was there more than one issue with their thought processes/actions?
 - If there wasn't a bug introduced, did the participant have any incorrect thought processes/actions?

The first and second researcher separately analyzed and coded another 10% of the video transcriptions using the updated codebook. The researchers reached an inter-rater reliability Pooled Kappa Cohen (De Vries, Elliott, Kanouse, & Teleki, 2008) score of 84% based on this analysis. The first researcher then used the refined code book to go back through the data reviewing and correcting the rest of the segments. The final codebook used for the Obstacles, Breakdowns, and Bugs can be found in Table 32, Table 33, and Table 34. This validated coding scheme was then re-used for the non-think-aloud section. Combined, the think-aloud and non-think-aloud section had 27hrs of participant data working with the Arduino that was coded by the researchers using this codebook.

Table 32 – Codes for Obstacles: *Hurdles that the participant had to overcome*

	Description	Example
HW	Question or confusion about a HW component, prototyping tool, building the circuit, or debugging a HW error	<i>“But how do I connect two grounds if there is only one on the Arduino?”</i>
SW	Question or confusion about how to use the SW IDE, a code block, accomplishing a task in the code, or debugging a SW error	<i>Participant is staring at her code: “So I have it blinking once...how do I make it blink twice?”</i>
HW+SW	Question or confusion about the interaction of HW and SW, or unsure where to debug because there are issues are in HW or SW	<i>“It doesn’t matter which debugging hints [referring to HW or SW] ‘cause I have no idea what’s wrong”</i>

Table 33 – Codes for Breakdowns: *Errors in participant’s thoughts and/or actions*

	Type	Description	Example
HW	Incorrect Circuit Formation	Demonstrates error in thoughts or actions creating circuits that include components connected in parallel or series that should not be	<i>Creates parallel circuit connection between the LED and resistor when they should be in series</i>
	Leaves out Component	Neglects to include a HW component that should be in the circuit they are building	<i>Leaves resistor out of LED resistor circuit</i>
	LED Backwards	Demonstrates error in thought or action in the orientation of the LED	<i>“The negative side is the long leg”</i>
	Open Circuit	Creates an open circuit when building their circuit (does not include using the wrong Arduino pin)	<i>Doesn’t push component all the way into the Breadboard</i>
	Short Circuit	Creates a short in their circuit by connecting components in a way that short out parts of the circuit	<i>Connects both ends of the resistor to the same connection block in a BitBlox module</i>
	Wrong Arduino Pin	Connects the circuit to a wrong pin in the Arduino either purposefully or accidentally	<i>Plugs jumper into pin 8 when they thought it was in pin 9</i>
	Other	Other incorrect assessments, actions, or faults in knowledge about the HW	<i>Thinks that the resistor needs to be closer to ground than the LED</i>
SW	Blocks Not Connected	Attempts to connect SW blocks but fails to actually do so	<i>Places forever loop under pinMode block but doesn’t get them to lock together</i>
	Delay	Demonstrates error in thought or action in the purpose, usage or effect of the delay block	<i>Tries changing the delay to make the LED brighter</i>
	Pin Signal (HIGH/LOW)	Demonstrates error in thought or action in the purpose, usage or effect of setting the pin signal to HIGH/LOW	<i>Sends two HIGH signals in order to get the LED to blink</i>
	PinMode	Demonstrates error in thought or action in the purpose, usage or effect of the pinMode block.	<i>Forgets to initialize the pinMode for a second output</i>

Table 33 – Continued

SW	Sequentiality of Code	Demonstrates error in thought or action in the sequentiality of how the code runs	<i>Connects code under the end of a forever loop</i>
	Wrong Arduino Pin	Demonstrates error in thought or action in setting the pins in any of the SW blocks	<i>Includes a digitalWrite for an LED but forgets to identify the pin</i>
	Other	Other mistakes, incorrect assessments, or faults in knowledge about the SW	<i>Uses while loop to get LEDs to blink together</i>
HW + SW	Programming w/o Plugging in Arduino	Fails to have the Arduino plugged in while code is being uploaded to the board	<i>Unplugs Arduino when code has not finished uploading</i>
	Reprograms Arduino With Same Program	Re-uploads same code that is already on the Arduino	<i>Didn't see change from their new code changes so they reprogram the board again</i>
	Wrong Assessment of Bug Location	Demonstrates error in thought or action in narrowing down whether the bug is in SW or HW	<i>Blocks aren't connected properly in SW, but introduces a backwards LED to try and fix their problems</i>
	Incorrectly Thinks Problem is Solved	Indicates that they have completed the task or fixed a problem when they have not	<i>Participant says "I'm done." When they still haven't added a pinMode to initialize one of the pins for their LED circuits.</i>

Table 34 – Codes for Bugs: *Faults introduced and fixed*

	Type	Description	Example
HW	Introduced	Bug in the circuit was introduced	<i>Shorts a resistor to itself when building the circuit</i>
	Fixed	Bug fixed or removed from the HW setup	<i>Takes apart circuit with shorted resistor</i>
SW	Introduced	Bug in software was introduced and uploaded to the Arduino	<i>Forgets to include a pinMode for one of the outputs and uploads code to the Arduino</i>
	Fixed	Bug fixed or removed from the HW setup	<i>Includes a previously forgotten pinMode for one of their pins</i>
HW+SW	Introduced	Bug in HW+SW was introduced to the setup	<i>Forgets to plug in Arduino when they press the button to upload the code</i>
	Fixed	Bug in HW+SW was introduced to the setup	<i>Realizes they forgot to plug in the Arduino and plugs it in and uploads code</i>

6.6.4 Overview of the Obstacles, Breakdowns, and Bugs Data

The data reveals trends between the obstacles, breakdowns, and bugs. There were similarities between the data in the think-aloud and non-think-aloud sections so they were combined in the numbers presented to convey the difficulties participants encountered with

the various tools. We will first outline the circuits and code that each activity required the participants to build, then provide an overview of the prevalence of the obstacles, breakdowns, and bugs across activities and groups, and then end with a comprehensive break down of the thematic trends we found across these codes diving into the specifics of the participants' issues in the hardware, software and hardware+software.

6.6.4.1 Activity Overview

The goals of each task are important for contextualizing the information provided by the numbers and types of bugs, breakdowns and obstacles experienced by the participants. Therefore, I will briefly review the circuits and code that could be built to complete each task.

Blinky LED: The introductory task walked participants through creating the code and circuit necessary to blink an LED forever. They were provided step by step instructions to create the circuit and code with each of the steps explained from a conceptual stand point. For example, the significance of initializing the pins as inputs or outputs. The circuit and code created by the participants for this activity are below in Figure 47.

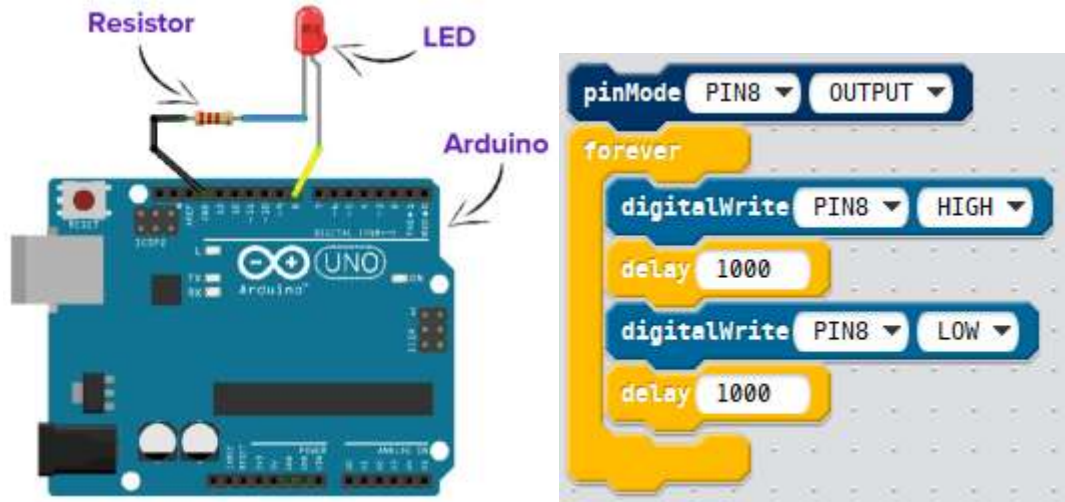


Figure 47 – The circuit and code created by participants in the Blink LED activity

Task #1: The first task required the participants to create a slightly more complex hardware and software setup. To build the hardware for the first task the participants had to replicate the circuit they created in the Blinky LED activity twice using different pins. To create the software, the participants had to figure out how to initialize a second LED and properly integrate code for blinking two LEDs into the same forever loop. Figure 48 has a diagram of one version of a circuit and code that participants could make in order to complete Task #1. The participants had a choice in which pins they used and could get the circuit to work properly even if the pinModes were not placed at the beginning of the code.

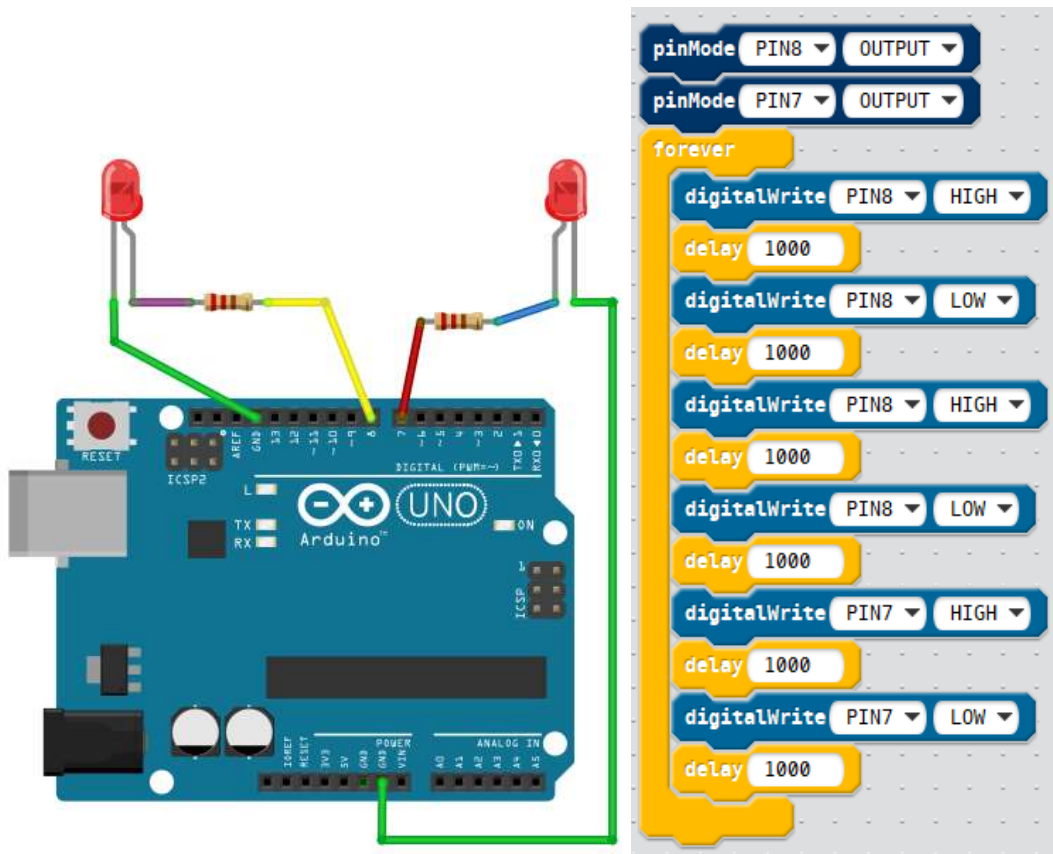


Figure 48 – Task #1 example circuit and code

Task #2: The second task required the participants to switch prototyping tools and figure out how to use it while adding another LED to the circuit they used in the Blinking LED activity. The software component required the participants to replicate the same code as was used in the Blinking LED activity. Figure 49 shows one version of the circuit and code that will complete the task. The participants were required to use PIN3 in the hardware, but the code could have variations (ex. number of digitalWrites used) as long as it would blink the LEDs in their circuit.

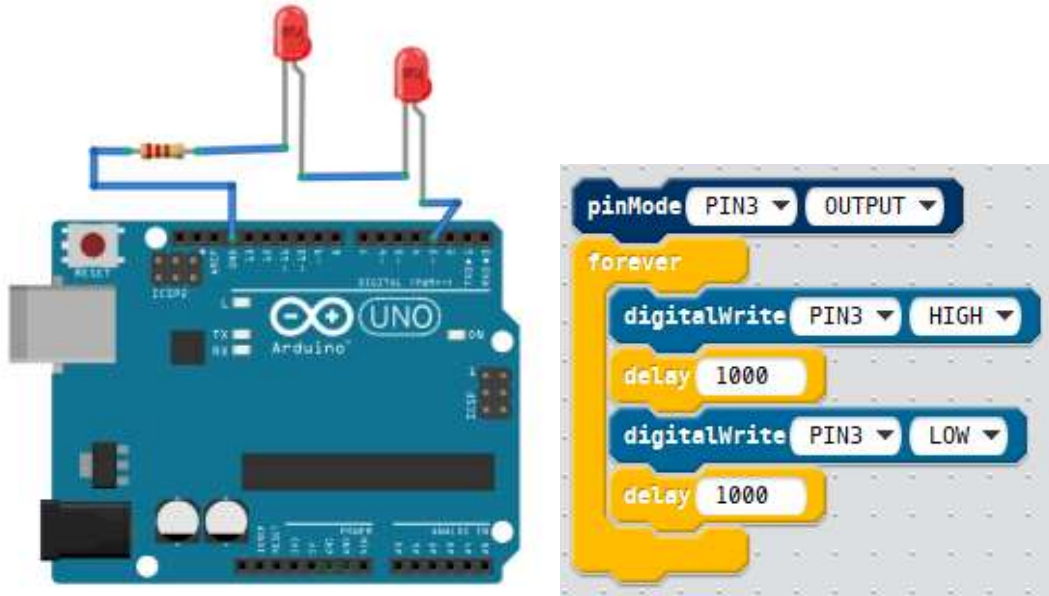


Figure 49 – Task #2 example circuit and code

As the data below will reveal, based on the number of bugs, breakdowns, and obstacles the Blinky LED activity was the easiest, while the first task was most difficult. From a software standpoint, organizing the code to blink two LEDs consecutively in the first task, posed conceptual hurdles for organizing the code resulting in more issues with the software than the hardware. In the second task, switching prototyping tools resulted in more hardware issues than software issues. We will now provide an overview of the number of obstacles, bugs, and breakdowns faced by each group in the various activities.

6.6.4.2 Obstacles Encountered by the Think-Aloud and Non-Think-Aloud Section

The data from the obstacles presented in Figure 50 and Figure 51 show where participants had questions and confusions. The researchers coded more obstacles in the data from the think-aloud participants (204 obstacles) than in the data from the non-think-aloud section (148 obstacles). Although there were fewer participants in the think-aloud

section, it provided more insight into the obstacles because the participants would articulate their questions and explain if and why something was confusing. In this section, we also coded the data from when the participants were initially learning about the LED circuit and Arduino because the participants were voicing their questions as they traversed the material for the first time. This is reflected in the high number of hardware obstacles identified in the Learning Activities. In the non-think-aloud section, we relied mostly on when the participants went back in the slides to seek information or on the rare occasion that they stated their problems out loud either seeking help from the researcher or out of frustration. In the non-think-aloud section, we only coded the Blinky LED activity and Tasks because there was little to be observed in the initial learning activities.

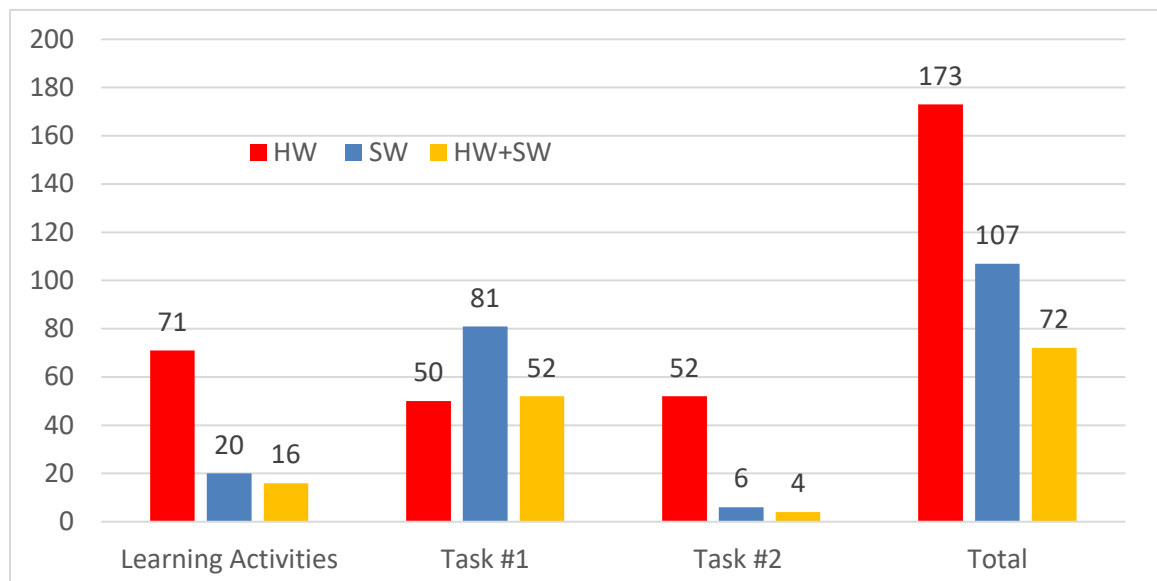


Figure 50 – Obstacles in Think-Aloud and Non-Think-Aloud Section by Activity

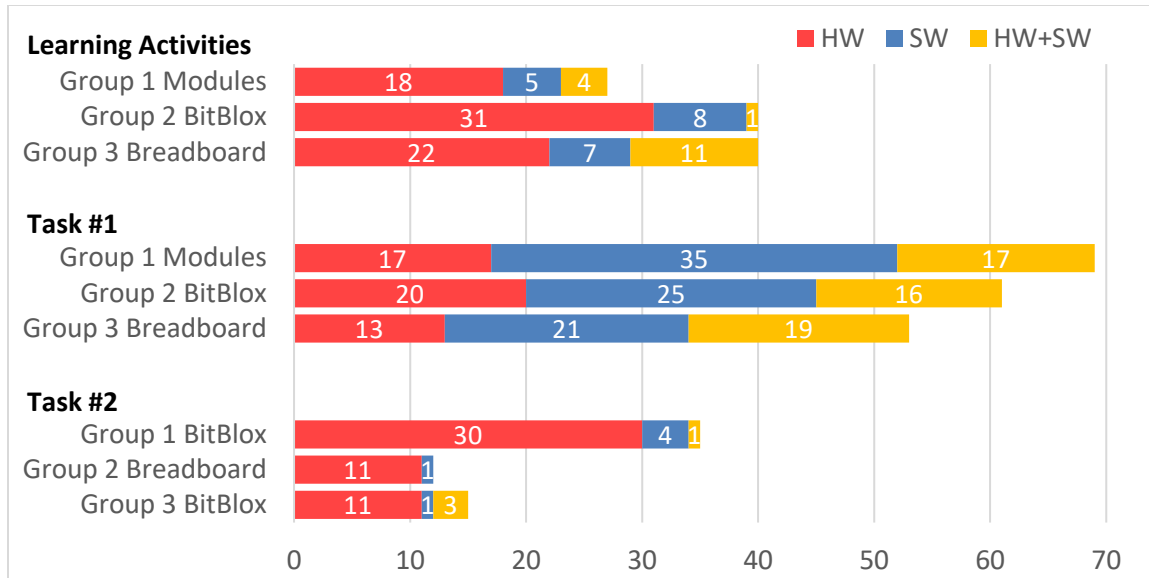


Figure 51 – Obstacles in Think-Aloud and Non-Think-Aloud Section by Location and Group

Overall, the participants had a greater number of questions and confusions when it came to the hardware than the software or hardware+software. We expected the hardware to pose more confusion since the code was blocks-based and the hardware was not perceived to be as beginner friendly based on previous analyses (Blikstein, 2015; DesPortes & DiSalvo, 2017); however, the activity impacted the types and location of the questions and confusions. While drawing conclusions solely based on the differences in number of obstacles between the groups could lead to false conclusions (since the number of obstacles was often dependent on whether or the participants were voicing their questions and confusions), the trends in this data are similar to the ones found in the bugs and breakdowns.

6.6.4.3 Bugs Introduced by Think-Aloud and Non-Think-Aloud Section

The number of bugs provides a good starting point to discuss the differences between the groups, which is complimented with data from the breakdowns. Overall, there were about as many hardware bugs as software bugs, with far fewer hardware+software bugs (see Figure 52 and Figure 53). The Blinky LED activity had fewer bugs introduced than either of the other two activities, which is expected since the participants were walked through this activity step by step. When errors did occur, they were more likely to be in the hardware—26 bugs—than in the software—10 bugs. In Task #1, there were more software bugs than hardware bugs—185 compared to 100—with the hardware+software only contributing to 24. In Task #2, the hardware component resulted in more bugs than the software component—90 compared to 35.

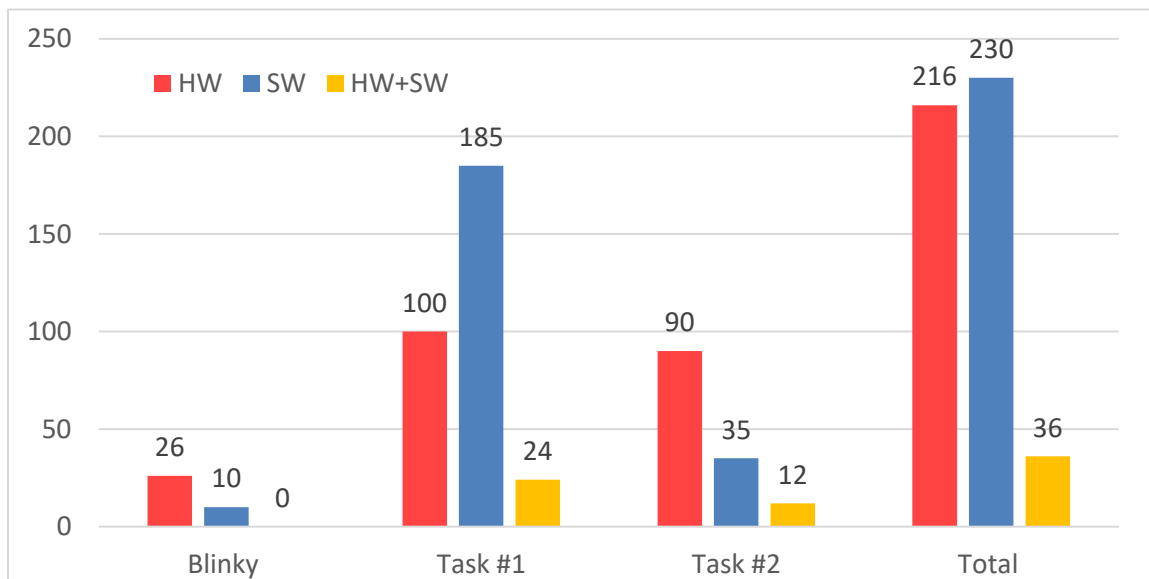


Figure 52 – Bugs in Think-Aloud and Non-Think-Aloud Section by Activity

Examining the bugs broken down by activity and group (*see* Figure 53) provides us with a more comprehensive picture of where the challenges were when participants were using the different tools.

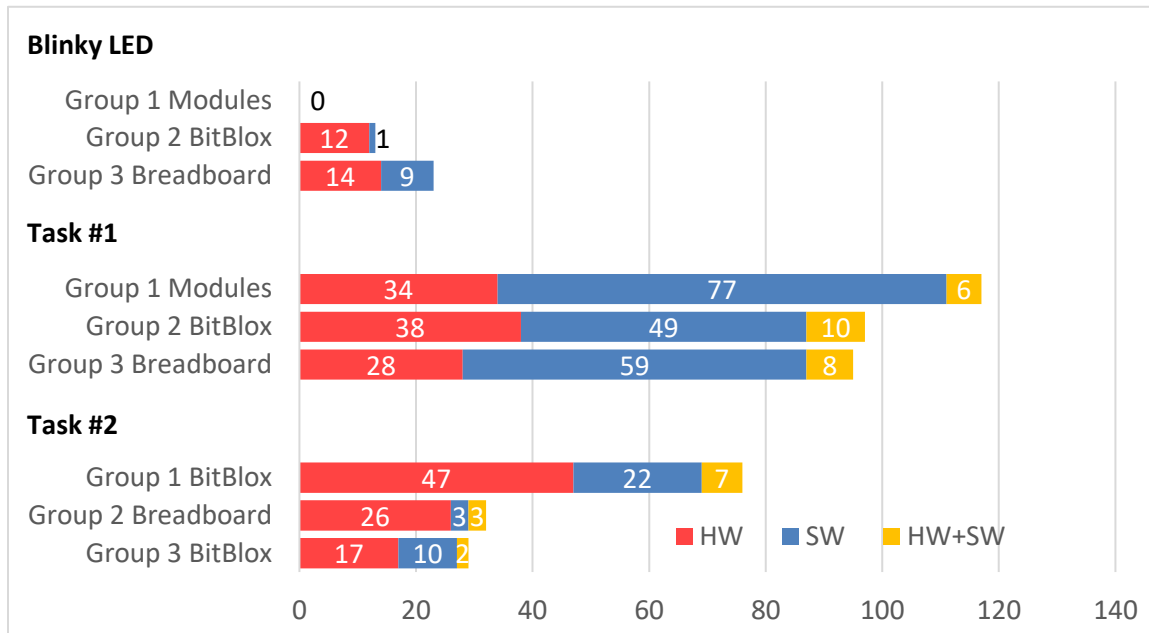


Figure 53 – Bugs in Think-Aloud Section and Non-Think-Aloud by Location and Group

In the Blinky LED task we see that the Modules [Group 1] had no bugs since there were far fewer opportunities for them to make mistakes. In the participants using the BitBlox [Group 2] and Breadboard [Group 3], we do see 13 and 23 bugs respectively with a majority of hardware bugs. These numbers represent that initial difficulty of working with the tools even when being walked through the Blinky LED circuit step by step. As participants struggled with the prototyping tools in the BitBlox and Breadboard group, they also made more errors in the software.

The groups with the highest and lowest numbers switch when we get to Task #1. The Breadboard [Group 3] and BitBlox [Group 2] participants introduced about the same with 95 bugs and 97 bugs respectively, and the Modules [Group 1] introduced the most with 117 bugs. This was unexpected given that we predicted the Modules would continue to have the fewest bugs. Although they had a slight change in their hardware, it was still intended to integrate more scaffolding than the BitBlox or Breadboard would have. The hardware bugs between the groups turned out to be about the same with slightly fewer in the participants using the Breadboard who introduced 28 compared to the Modules [Group 1] who introduced 34 and the BitBlox [Group 2] who introduced 38. Furthermore, the software bugs introduced by the participants using the Modules [Group 1] was higher with a count of 77 software bugs while the Breadboard [Group 3] introduced 59 and the BitBlox [Group 2] introduced 49 software bugs. As we discuss in section 6.6.5.3, faulty hardware issues in the modules could have impacted these numbers.

Task #2 required the participants to switch prototyping tools. The participants in Group 1 who transitioned from the Modules to the BitBlox had over twice as many bugs introduced than Group 2 and 3 with 76 bugs introduced compared to 32 and 29. The task of transferring to the open-ended prototyping tool from the modules proved more difficult than transferring between the two open-ended tools. Furthermore, the participants transitioning from the Modules to the BitBlox group not only introduced more hardware bugs (47 compared to the BitBlox participant's 26 bugs and the Breadboard participant's 17), they also introduced a substantial number of software bugs in comparison to the other two groups (22 software bugs compared to BitBlox participant's 3 and Breadboard participant's 10). Understanding the types of breakdowns in the sections that follow

provide a better understanding of where the misconceptions and mistakes that participants had were coming from that led to these differences.

6.6.4.4 Breakdowns Faced by the Think-Aloud and Non-Think-Aloud Participants

The breakdowns offer a picture of what and how participants were incorrectly assessing and interpreting information as they worked through the activities. While the bugs indicated when an error was introduced in the setup, a breakdown indicated whenever there was any type of misconception, mistake, or error even if it did not lead to the introduction of a bug—all bugs were tagged with at least one breakdown, but not all breakdowns were bugs. Overall, we see more breakdowns than bugs because, while all bugs were linked to a breakdown, not all breakdowns introduced bugs. Figure 54 and Figure 55 provide an overview of the breakdowns across the groups and activities.

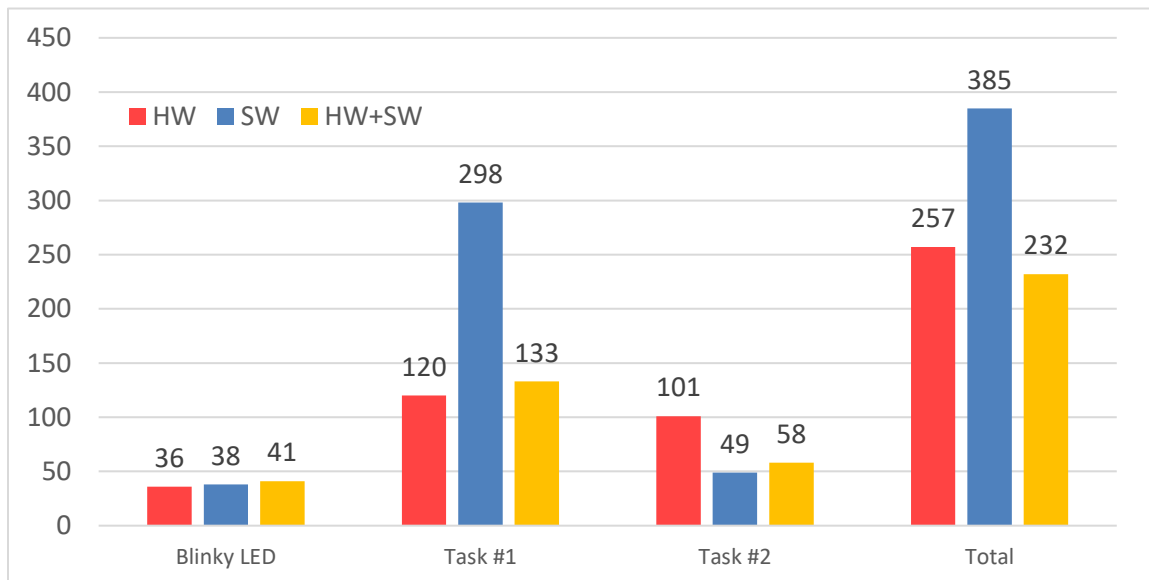


Figure 54 – Breakdowns in Think-Aloud and Non-Think-Aloud Section by Activity

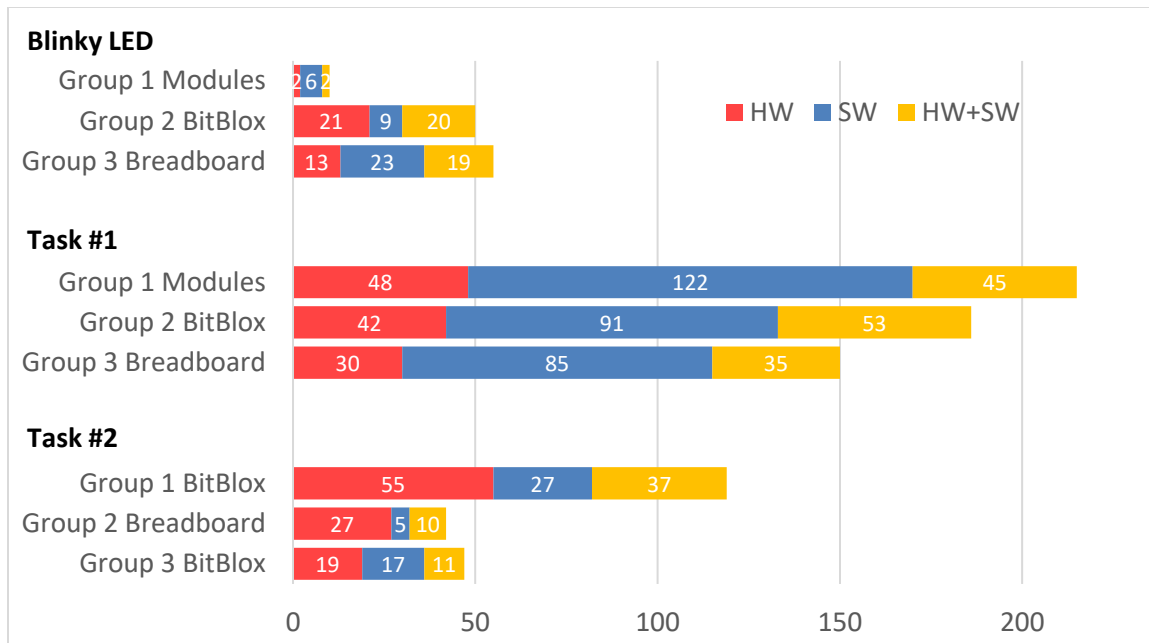


Figure 55 – Breakdowns in Think-Aloud and Non-Think-Aloud Section by Group

The prevalence of breakdowns appears similar to the bugs, in that software breakdowns predominated Task #1 and hardware breakdowns predominated Task #2; however, there was a higher prevalence of software breakdowns and hardware+software breakdowns than we saw in the bugs. The results indicate the participants in the Breadboard [Group 3] and BitBlox [Group 2] groups faced more breakdowns in the Blinky LED activity while the group that started with the Modules [Group 1] faced more breakdowns in Task #1. This group also had the most bugs as they transitioned to the BitBlox in Task #2. The following sections examine what these breakdowns actually were across the hardware and software using the codebook in Table 33, while contextualizing them with the bugs and obstacles participants faced.

6.6.5 Hardware Bugs, Breakdowns and Obstacles

Examining the data confirms that the participant’s experiences with the hardware varied based on the activity and the prototyping tool they used. The bugs, breakdowns and obstacles in Figure 56 show a combined view of how the prototyping tool impacted their hardware experiences. Despite the step by step instructions in Blinky LED, participants using the BitBlox and Breadboard had difficulties starting up with their tool. In contrast, the participants using the static modules had few difficulties in the Blinky LED activity, but when they transitioned to the variable modules they ended up having a harder or equally challenging time during Task #1. Furthermore, changing prototyping tools proved the most difficult for participants transitioning between the Modules to the BitBlox, with the easiest transition occurring for the group switching between the Breadboard to the BitBlox.

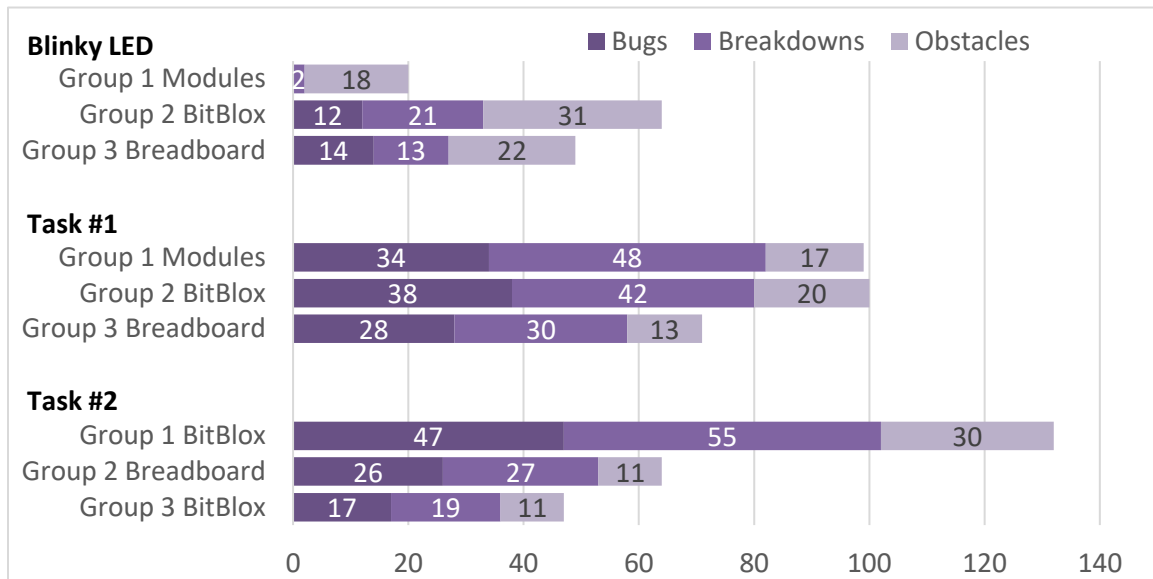


Figure 56 – Hardware Bugs, Breakdowns, and Obstacles by Group and Activity

The types of hardware breakdowns provide insight into the errors that participants were actually hitting when working with the tools. There were six consistent breakdowns

that emerged from the data: using the wrong Arduino pin, creating a short circuit, creating an open circuit, assessing the LED directionality incorrectly, leaving out a component in their circuit, and forming their circuit incorrectly so there were components either in parallel or series when they should not be. The *Other* category was used as a bucket for the *long-tail* or the errors that were not prevalent enough to warrant their own category. Figure 57 illustrates the number and type of breakdowns within each group.

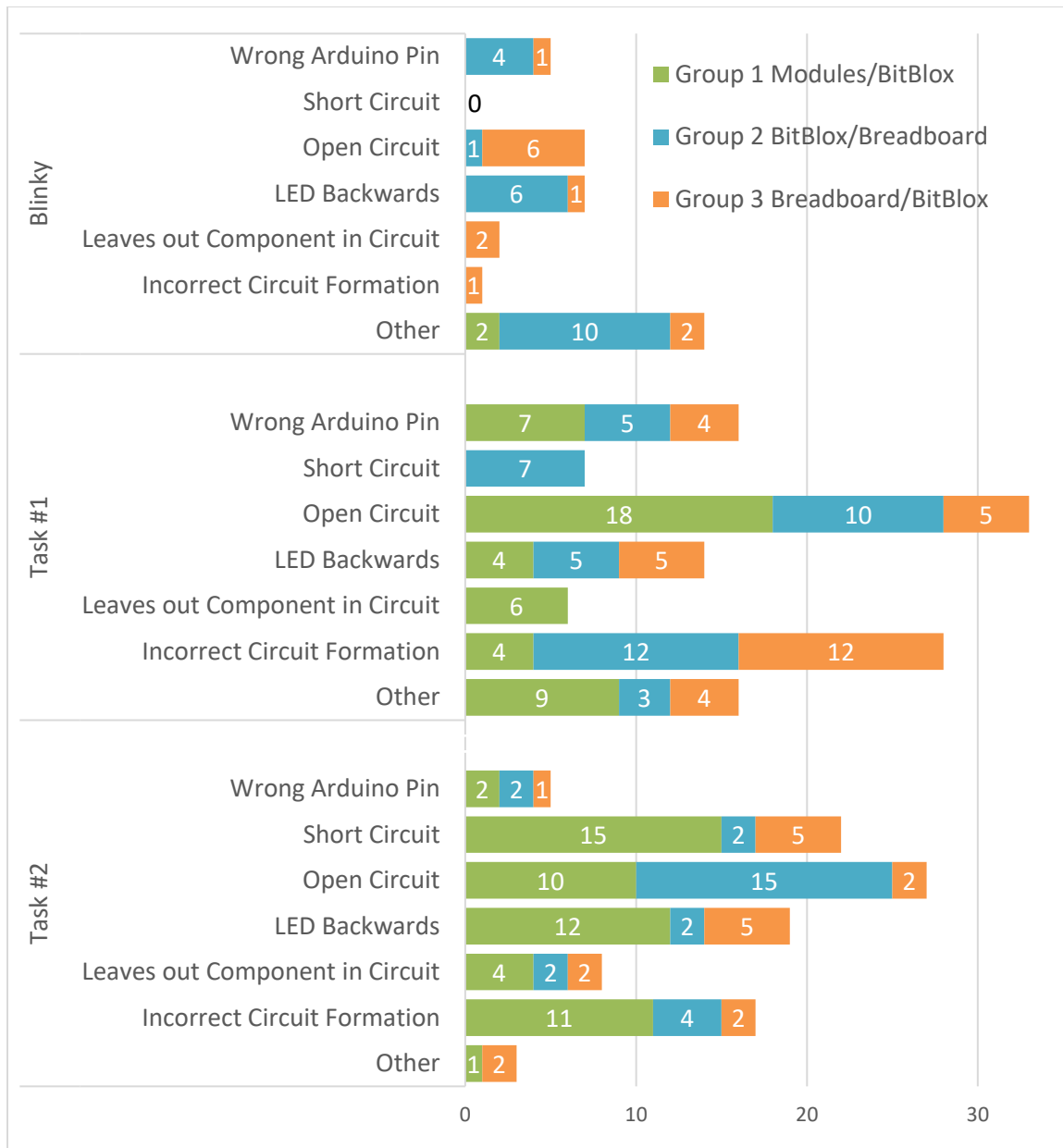


Figure 57 – Hardware Breakdowns from Think-Aloud and Non-Think-Aloud Section

6.6.5.1 Wrong Arduino Pin and LED Backwards

The breakdowns of *Wrong Arduino Pin* and *LED Backwards* was observed in all of the groups, and usually led to the introduction of bugs within the participant’s setup. The small size of the components contributed to the difficulties participants had with these

breakdowns. For example, one participant looking for the positive side of the LED said: *“The positive end is the longer side and the curved side. That one is longer...the whole thing is curved, so that’s not really a good representation...[continuing to look at the LED]...oh! I see it. That side is straight.”* While the majority of participants were eventually able to figure out which side was which, it took added time and effort to do so and on occasion they ended up getting it wrong after taking the time to analyze it.

The other reason participants introduced this breakdown into their circuit was just to check to make sure they had it right. For example, a participant in the think-aloud section had an open circuit in his hardware setup and made a change in his hardware. The researcher asked him what he changed and he said, *“Oh ok, so ok, when I switched it I turned [he picks up LED and examines it] positive side and negative side just to see if it would do anything different, but it didn’t, so I’m just going to switch it back to what I had before cause I’m pretty sure that’s what I had before, but I just wanted to see if it did anything different”.*

The tiny size of the Arduino also posed its own usability issues with participants having to spend time trying to identify certain pins or making mistakes in identifying the pins they were using. For example, one participant who was trying to plug in a ground wire, picks up the Arduino and stares at it: *“I can’t tell..okay...8, 9, 10, 11, 12, 13...ground?”* He was able to find the correct pin after counting it out. Upon inspecting the data, half of the *Wrong Arduino Pin* breakdowns were from the think-aloud section (i.e. only 15 of the 45 participants) who were using an Arduino that did not have numbering on the sides of the pins (see Figure 58 left). Because of the difficulty participants were having, we transitioned to one with the numbering (see Figure 58 right), which made the error less

common across the participants in the non-think-aloud section. Furthermore, when we did see it occur it was more likely to be because of a conceptual difficulty than a slip in which they mistook one pin for another. For example, a participant consciously choosing to put a jumper cable into the 5v pin instead of a digital pin.



Figure 58 – Arduino without labeling on the pin headers (left) and with labeling (right)

6.6.5.2 Short Circuit

The Short Circuit breakdown led to bugs and was mostly encountered with participants using the BitBlox. The participants would hit this error when they interpreted how to use the tool incorrectly and purposefully plugged both ends of one component into the same section of a BitBlox module, therefore shorting it to itself. This misconception was also found in participants who used the BitBlox in the comparative classroom study reported on in CHAPTER 5 (DesPortes, Anupam, et al., 2016).

6.6.5.3 Open Circuit

The open circuit bug was found to persist across all the tools. In Blinky LED, the participants were learning to use their prototyping tool and those using the Breadboard often made a slip in how they were using the tool leading to an open circuit bug by plugging components into holes that were next to one another but not electrically connected. Booth et al.'s study noted similar issues with miswiring in their study of programmers using the Arduino (Booth et al., 2016). Surprisingly the open circuit bug became more prevalent in the groups using the BitBlox and Modules in Task #1. The video data revealed that participants using the BitBlox usually introduced open circuit bugs for other reasons. For example, some of these errors were caused from the BitBlox that were not snapped together that spread apart and pulled the components out of the blocks as they moved. Other open circuits were caused because participants consciously set up their circuits incorrectly. For example, connecting a resistor with only one leg attached to the rest of the circuit (ex. Figure 59).

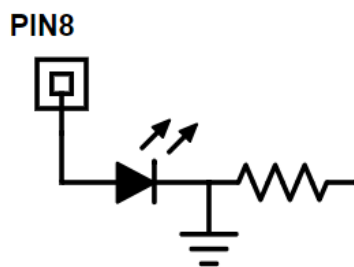


Figure 59 – Example of open circuit created by participants who did not connect components into their circuits properly

The prevalence of the open circuit bug in the participants using the variable modules ran counter to our initial hypotheses, which would have predicted this would

rarely happen since there are only four things to plug in (i.e. LED, resistor, power, and ground). If the components are plugged in correctly, they should connect to one another. However, the header pins caused unexpected issues both in their usability and because the pin headers became loose over time making inconsistent connections between components (ie. bugs caused by faulty hardware instead of usability issues). The faultiness of the hardware was not caught by the researcher until the last two participants in the section. Of the 15 participants who worked with the modules, 9 experienced an open circuit bug from connections using the pin headers in the modules. Within 2 of these instances (both in the think-aloud section), it was clear the participant had not pushed their components into the module all of the way. For example, one participant noticed her LED coming out of the pin header saying, “*and that’s not connected*”, which was subsequently fixed when she saw the issue. Out of the 7 participants who might have had an error because of the faulty hardware, the video reveals the last 3 participants pushing down the components down and not getting a secure connection (i.e. the LED flickered as they pushed the component into the module). In the 4 remaining cases, 3 participants fixed their issue the first time they attempted to push their components down all the way; and one was unable to complete the task with this as a remaining bug. In these cases, it was unclear if it was caused by a usability error (i.e. they didn’t have all of their components pushed down into the pin headers) or caused by the faulty hardware (i.e. their components were pushed down, but not making a solid connection because of the pin header).

Insight for supporting participants in these situations, comes from their processes for overcoming the issue. The time it took participants to check their hardware connections ranged from never attempting it (the case of the participant who did not complete the task)

to taking anywhere between 30sec to 25min (avg. ~8min). Furthermore, this was sometimes only attempted after the participant was prompted by the hardware debugging hints. The longer it took for the participants to debug this issue the more iterations they went through in their code. In some cases, the participants had the correct code that was disassembled when they did not see the intended change in their hardware. This was similar to what happened in other instances when the location of a bug was misdiagnosed (more on this in section 6.6.7.1).

6.6.5.4 Leaving Out Components in Circuit

Participants using the variable modules were the only participants to leave out a component in the first task. The participants in this group were using a static module in the Blinky LED activity, which led some participants to forget they needed to plug in components in the variable modules when they first started using them. Participants generally identified and fixed this error quickly. In the second task, we see a greater prevalence of this error across tools caused by participants forgetting to include the resistor in the circuit with the two LEDs they had connected in series.

6.6.5.5 Incorrect Circuit Formation

The participants using the BitBlox and Breadboard, often created incorrect circuits placing components in series or parallel when they should not be. For example, participants in the first task who should have been creating the circuit demonstrated in Figure 60 (left), combined both LEDs into one circuit (Figure 60 (right)). This rarely happened with the participants using the modules since each module had all the components needed for one circuit.

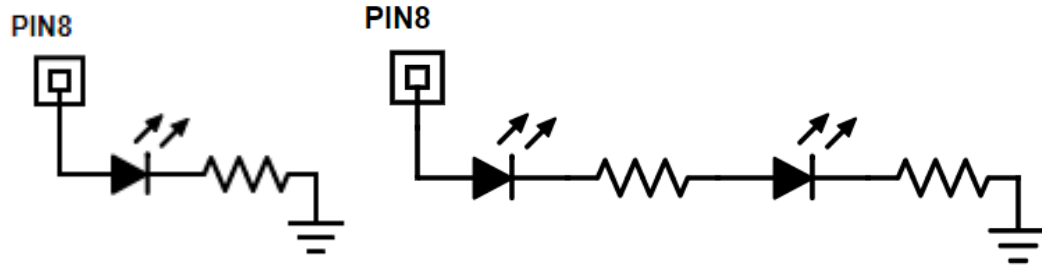


Figure 60 – Task #1 correct circuit formation (left) and Task #1 common incorrect circuit formation (right)

Participants in the second task often combined the LEDs and/or resistor in parallel instead of series in the second task. The participants in Group 1 who transitioned to the BitBlox from the Modules often translated the circuit incorrectly as they built it out using the BitBlox. Figure 61 shows what the circuit should have looked like and Figure 62 shows the three most common incorrect circuits that were built.

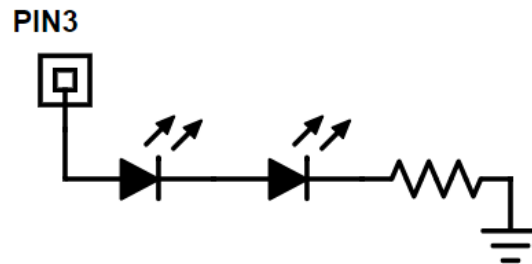


Figure 61 – Task #2 Correct Circuit Formation

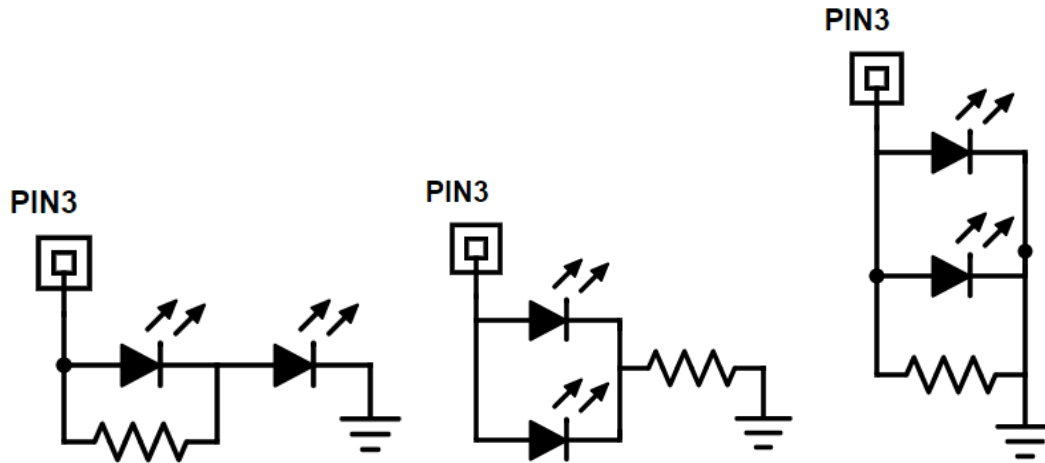


Figure 62 – Task #2 Common Incorrect Circuit Formations

6.6.5.6 Other Hardware Findings

Outside of the obstacles and breakdowns reviewed in the above sections, there were a wide range of other hurdles and questions the participants encountered in relation to the hardware; from questions about how the Arduino is *actually* able to supply 5 volts, to asking whether it was necessary to extend the USB cord all the way for it to work. Other important findings involve participants’ problems using ground pins, their initial confusion using the prototyping tools, their struggle to correctly interpret visual information, and the difficulties they faced transitioning between the tools.

Creating a Common Ground

One consistent conceptual obstacle was the concept of ground and whether or not the participants could use different ground connections on the board. Although the guide indicated they could, the participants were still uncertain when it came to actually using them in the tasks. One participant demonstrates this as she looks for a second ground to connect a second LED circuit to the Arduino: “*I don’t see like another pin that says*

ground...hmm...I see it but on the other side, like underneath power, but I don't know if that will also work. Or maybe the uh, these three that are next to it, maybe they are also ground." Even though she sees the correct pin her uncertainty leads her to introduce a bug by plugging it into a pin adjacent to the ground pin she is using. This is interesting because the participants wanted to create a common ground between the circuits even though we did not discuss the importance of this concept. A common ground is important because voltage is the potential difference between two points. A common ground ensures that the bottom or base voltage (i.e. 0 volts) from which that difference is measured is the same. The grounds on the Arduino board are connected, but the participants still faced an obstacle when deciding whether they could or should connect their circuits to different ground pins. Furthermore, while the participants in the Breadboard and BitBlox group had the tools to connect their circuit to a common ground they did not usually use their tool when they were stuck on this obstacle. In the comparative classroom study, we noted the conceptual hurdle for participants to link one point in their circuit to another location in their prototyping tool when they ran out of holes for their ground pins.

Purpose of the Prototyping Tools

Data from the obstacles made it apparent that the concept of the prototyping tool was foreign to the majority of the participants as they first began to work with either the Breadboard or BitBlox. Participants, therefore, did not necessarily understand their purpose; as one participant asked, *"so why is this breadboard actually used to connect things instead of a circuit?"*. Within the video footage, participants can be seen trying to figure out a way to attach components together after going through the slides describing how you put components into the tools. For example, a few participants tried stacking the

components on top of one another on the table in order to get them to connect as they neglected their prototyping tool.

Correctly Interpreting Relevant Visual Information

Participants also hit obstacles trying to correctly interpret the relevance of certain visual cues. For example, not knowing whether or not the wire colors mattered, wanting to know if the “~” next to the digital pins (indicative of pulse width modulation capabilities) mattered, interpreting the pin numbers as voltages, or mistaking the blinking of the Rx and Tx LEDs on the Arduino (which blink when code is uploaded) for changes they created using their code. This also happened with the coloring between the different BitBlox. Participants sometimes thought they needed to find two separate BitBlox with the same colored section so that they could *properly* connect their components. One participant asked after reading the directions on BitBlox, “*does it matter what color they are?*” The researcher told the participant they couldn’t answer any questions and the participant later asked, “*is there a red one we can use?*” as he tried to match the colors between the blocks.

The BitBlox also had the ability to connect the various blocks which was visible from the edges of the tool, but while participants noticed you could do this they did not necessarily know how to do this. One participant, when trying to push two together horizontally said, “*It’s like LEGOs, but it doesn’t fit. It’s not right*”. These obstacles indicate that the participants are particularly attuned to the various visual cues in the hardware, but that the information can be misinterpreted and lead them astray.

In the directions for using BitBlox, we did not explicitly cover that participants could snap them together, although the pictures in the directions did have the BitBlox

connected. Out of the 15 participants who used the BitBlox only two did not end up connecting any of the modules, but they did not necessarily connect all of the blocks they were using. This could lead to open circuits as discussed in section 6.6.5.3. The video footage also revealed that connecting the blocks did not fit seamlessly into their work processes. Participants often tried connecting the BitBlox together at the same time they were trying to insert the component in them (contrasted to connecting two BitBlox first then sticking the component into the blocks). The parts usually had to be readjusted before they were all connected properly. This might have been why participants did not connect all of their BitBlox.

Transitioning Between Tools

The transition between the prototyping tools also caused issues dependent on which tools the participants were transitioning between. While the participants were given instructions on how to use the new tool they introduced to, they were not walked through an activity with it. As the data from the obstacles, breakdowns, and bugs revealed, the transition posed a greater challenge for the participants in Group 1 switching from the Modules to the BitBlox compared to other two groups. These participants had a high prevalence of breakdowns such as: creating short circuits, open circuits, inserting the LED backwards, and connecting components in parallel that should have been in series, which we did not see in the other two groups. Given that the Breadboard and BitBlox are isomorphic representations of one another with different connection schemes, switching from the modules to the BitBlox presents a larger conceptual gap. While participants gained some knowledge using the modules, the data demonstrates it did not encompass

information that transferred well to using the BitBlox. As one participant reflected, “*yeah, this is more confusing*”.

The participants who transitioned to the BitBlox from the Breadboard had the fewest breakdowns and bugs. Since we hypothesized that using the Breadboard was the hardest task, it aligned with our expectations that the BitBlox would be an easy transition for them. The participants who transitioned from the BitBlox to the Breadboard had an easier time than the modules group, but some still had some difficulty interpreting the connection scheme. One participant in this group, for example, who switched to the Breadboard after using the BitBlox said, “*so I’m just kinda confused. It just, it just looks completely different to me. I mean, so if I plug this in here [pointing to a spot on the breadboard], I mean how will I know if it’s actually going to do anything? It’s just, it’s just weird.*” It was common for participants in this group make slips with the breadboard creating open circuits by incorrectly putting components in separate columns or rows that should have been connected.

6.6.6 Software Bugs, Breakdowns and Obstacles

The greatest number of breakdowns and bugs came from how the participants were thinking about and working with the software. Similar to the hardware breakdowns the most software breakdowns were in Task #2, followed by Task #1 and the fewest in the Blinky LED activity. Figure 63 shows the software bugs, breakdowns, and obstacles by group and activity.

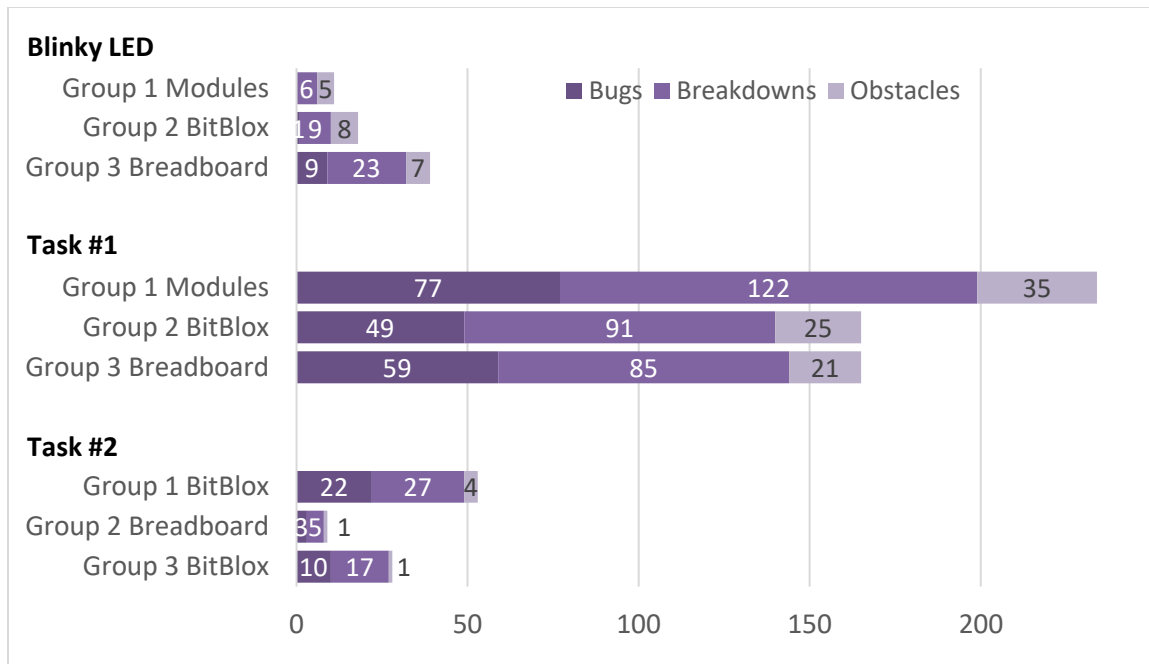


Figure 63 – Software Bugs, Breakdowns, and Obstacles by Group and Activity

In Blinky LED, the group using the Breadboard had the most software issues followed by the participants using BitBlox and then those using the Modules. In Task #1, the group using the modules had the most issues, with a similar amount of issues found in the participants using the BitBlox and Breadboard. Task #2, revealed similar difficulties for those transitioning from the Modules to the BitBlox, followed by those transitioning from the BitBlox to the Breadboard. The group transitioning from the Breadboard to the BitBlox had the fewest issues.

The types of breakdowns provide insight into the types of problems the participants were facing with the software (see Figure 64). The six most common types of breakdowns were, referencing the wrong Arduino pin, misunderstanding how code would execute sequentially, incorrectly using (or not using) a pinMode to initialize a pin as an *input* or *output*, incorrectly sending signals to the pins as *high* or *low*, incorrectly using or

interpreting the delay block, and incorrectly connecting blocks in the IDE. Similar to the hardware, we used an *Other* category for all the errors participants made when incorrectly assessing or interacting with the software. Figure 64 shows the counts for each of the breakdowns.

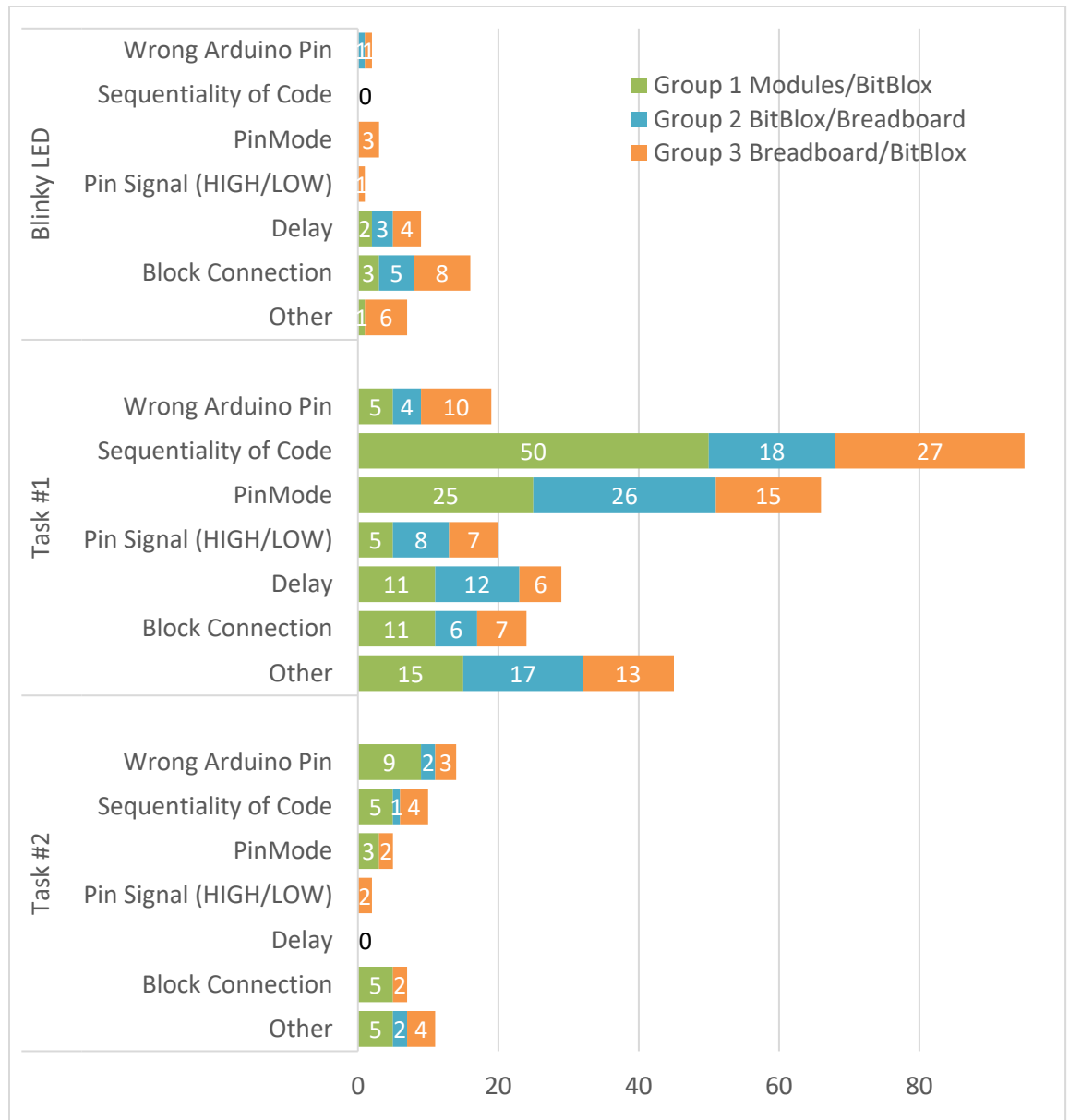


Figure 64 – Software Breakdowns from Think-Aloud and Non-Think-Aloud Section

6.6.6.1 Sequentiality of Code

The largest number of breakdowns came from participants incorrectly understanding the order of execution within their code, or what we are calling *sequentiality of code*. This error predominated Task #1 and accounted for much of the difference that

we saw between the Modules group and the other two groups, as they made almost twice as many errors as either the participants using BitBlox or those using the Breadboard. Two main mistakes made up the majority of the breakdowns in this category across all the groups: (1) creating two chunks of code with two forever loops (ex. Figure 65), and (2) attaching code after a forever loop (ex. Figure 66).

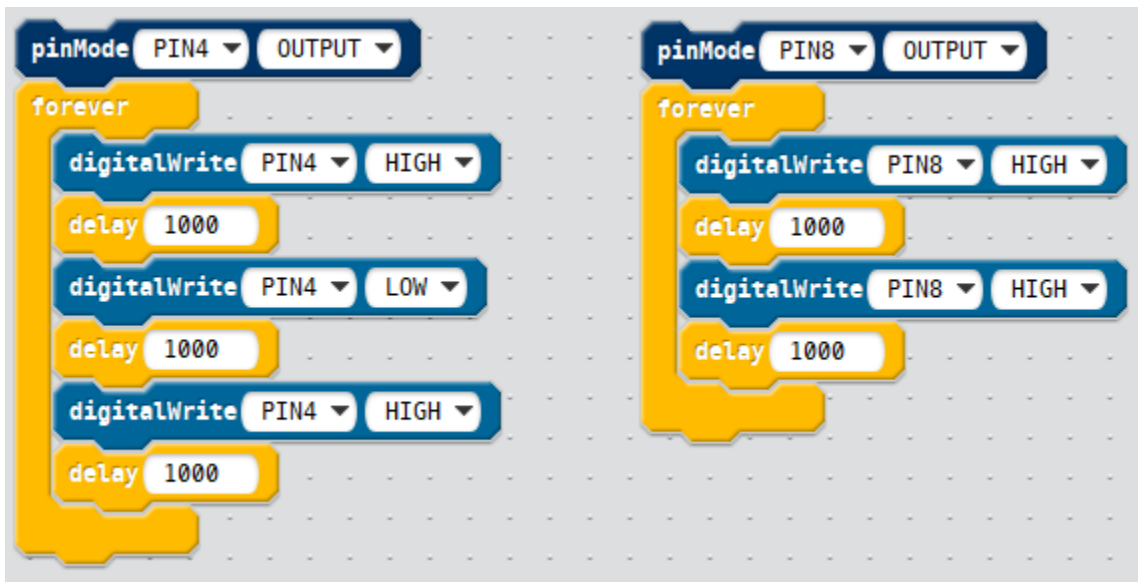


Figure 65 – Incorrect code initializing two forever loops

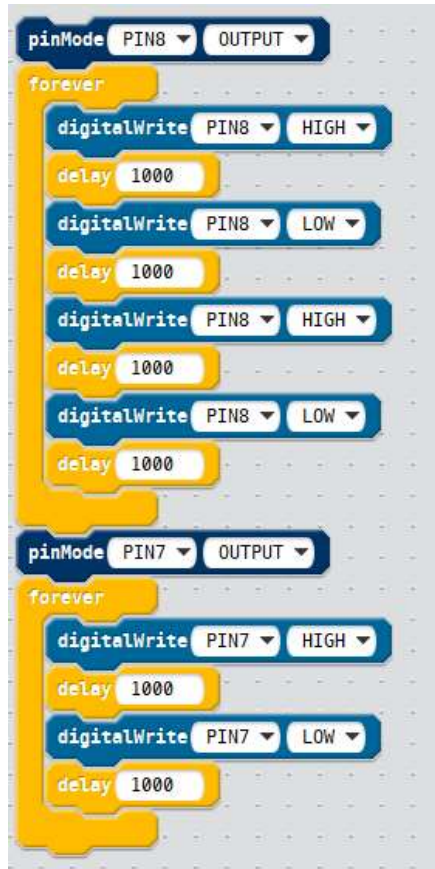


Figure 66 – Incorrect code which connects code after a forever loop

When Modkit compiles the code for two separate forever loops it does not run them at the same time, since the Arduino cannot run parallel processes or support multithreading. Instead, it runs the code that was last *touched* by the user. Participants sometimes ran into obstacles when something that was working would stop working because another chunk of code was *activated* by a change they made (see example below). Outside of the misconception of how the computer interprets code, the thought process for creating these two chunks of code is inherently incorrect. If the two blocks ran in parallel the LEDs would blink at the same time not sequentially as the task instructed (i.e. Blink LED 1 → Blink LED 1 → Blink LED 2 [repeat forever]). The second most common error the participants made, is in thinking that code placed after a forever loop would run. Instead, the Arduino

runs the code attached before the forever loop then stays within the first forever loop until power is reset, in which case it restarts execution from the top of the code.

Participants were generally unsure of how to correctly add the code for a second LED and often times switched in between several incorrect configurations. Participants also nested forever loops and brought in new code blocks to help them accomplish the task. The following example illustrates the sequence of configurations from a participant in Group 1 who had already constructed and connected her modules to the Arduino correctly. One module was connected to PIN8 and one to PIN6. She starts with the following code uploaded to her Arduino, which blinks her LED at PIN8 (Figure 67).

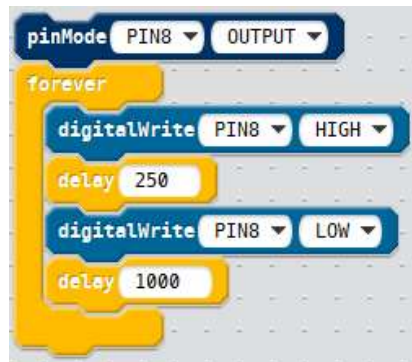


Figure 67 – Sequentiality Example Step 1

She then creates a second separate block of code for her second LED at PIN6 and uploads it (see Figure 68).

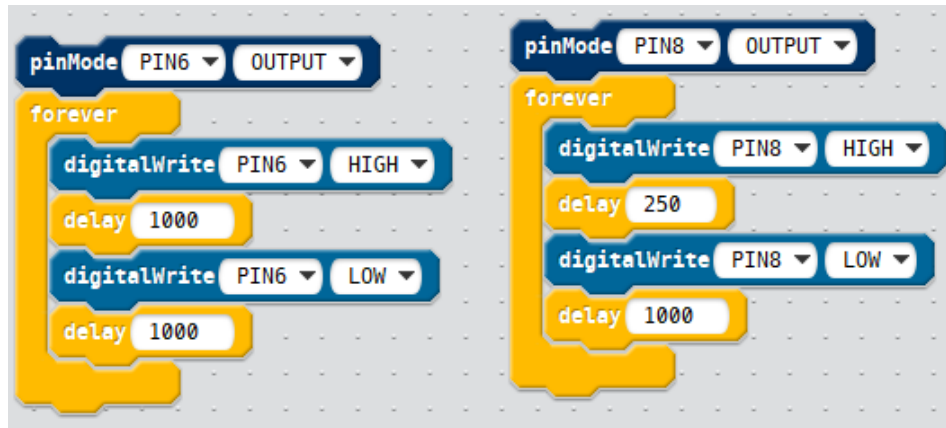


Figure 68 – Sequentiality Example Step 2

With this change, the LED that was blinking switches from PIN8 to PIN6 since she last touched that code chunk. She says, *“hmm now that one’s blinking and that one stopped”*. Her next solution involves using a while-loop block, connecting everything together in one chunk of code as shown in Figure 69.

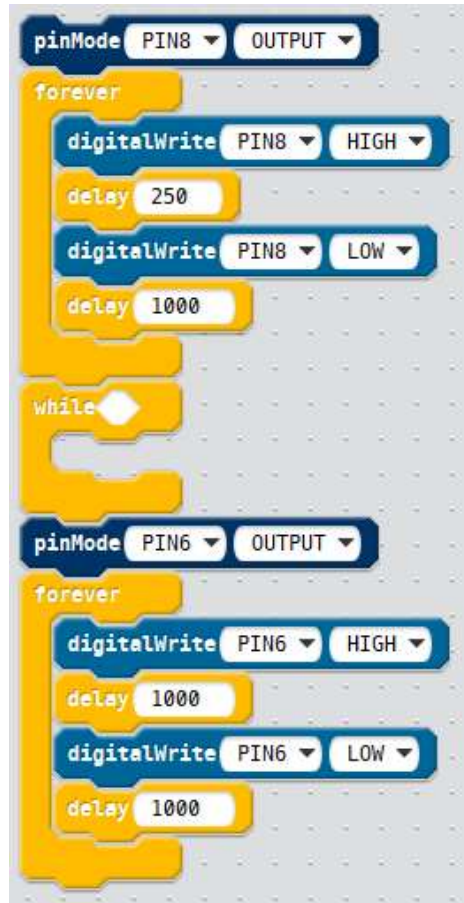


Figure 69 – Sequentiality Example Step 3

Uploading this code switches the code back to blinking the LED at PIN8 since it executes before the other code. Upon seeing this she exclaimed, “No? Why don’t you want to blink?!?!...Hmmm, ok”. She then separates all the loops (as seen in Figure 70) touching the PIN6 code last before uploading it.

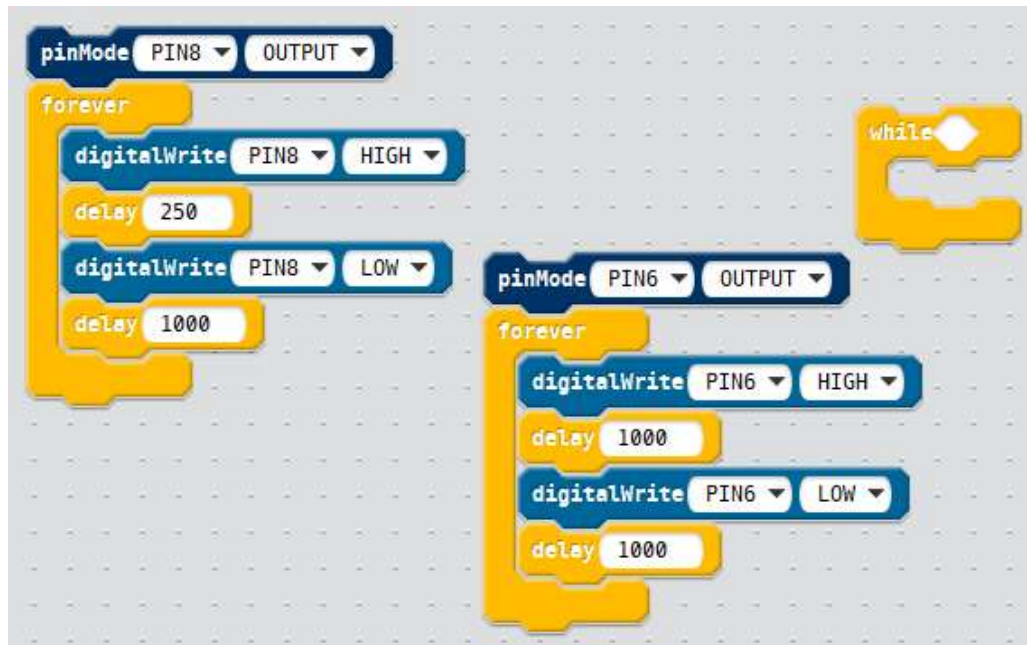


Figure 70 – Sequentiality Example Step 4

The LED blinking switches again, “*ok now that one’s blinking [referring to PIN6] and then before, this one was blinking [picking up the PIN8 code].*” She then clicks upload, with PIN8 being touched last, which then switches the blinking LED back to PIN8. She then states, “*yeah so they are two separate things...[scrolling through the blocks on Modkit]...so maybe to make them blink at the same time you don’t do while maybe you do and? [she picks up the and block from the menu but does not see a way to attach it] no that doesn’t work. You can’t do that.*” The participant eventually received a hint that tells her to use one forever loop, and she creates the code in Figure 71.

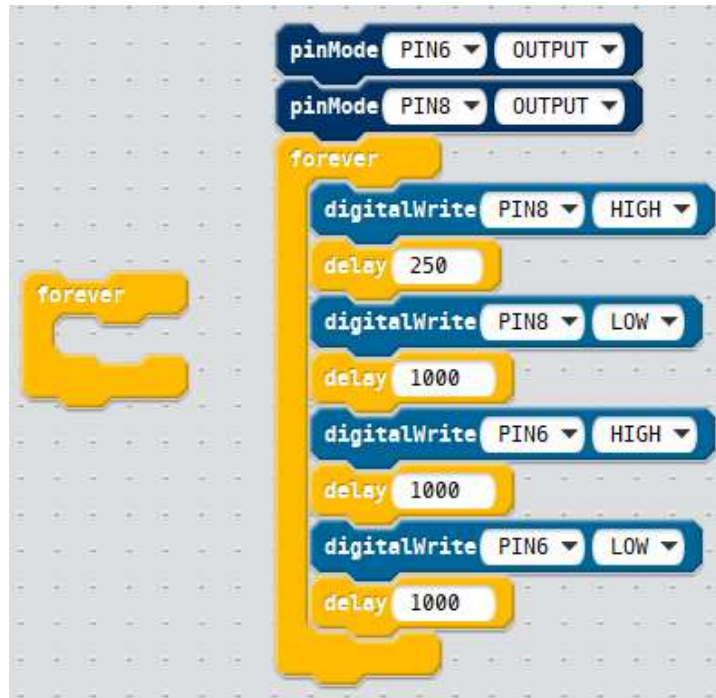


Figure 71 – Sequentiality Example Step 5

The participant last interacted with the code on the right, so it successfully blinked her LEDs in succession, while the forever loop on the left was *inactive*. She deleted the forever loop and added the code needed to make the LED at PIN8 blink twice and completed the task.

The example above demonstrates how participants would experiment with multiple iterations of code as they reasoned through how to blink their LEDs. As discussed in the next section, part of this misconception might have stemmed from confusion surrounding where they should be putting the pinMode to initialize the second LED. Since the Blinky LED activity involved putting a pinMode block above a forever loop, this code arrangement was readily accessible for many participants. Only 26 of the 46 participants started with putting the code for both LEDs within one forever loop on the first try, and some reverted from this method because of misdiagnosing other errors they encountered.

In CS education research, the work on plan composition has identified that novices often start with a *schema*, or chunk of code for a specific purpose, that they know in order to begin solving a problem (Rist, 1989, 1991). This indicates that participants might have been doing this here with the pinMode+forever loop schema they had used for the Blinky LED task; they had two LEDs to initialize so they created their schema two times.

6.6.6.2 Pin Mode Breakdowns for Initializing Hardware

The pinMode breakdown was the second most prevalent software breakdown that participants encountered. These breakdowns involved how participants were thinking about or using the *pinMode* block to initialize the Arduino pins as *input* or *output*. The two most common errors were: (1) not using a pinMode to initialize one of their pins as an output, or (2) using a pinMode and leaving it as an input instead of changing it to an output. The first issue could have been because participants did not know where to put it, or simply because they forgot it was necessary. For example, one participant was trying to figure out where she should put a second pinMode and asked, “*can one pinMode go under another pinMode though? Maybe?* [drags one pinMode under another] *Oh it can...that’s interesting.*”. Other participants put the pinMode into the into the forever loop unsure of whether this would work, “*but [pinModes] don’t normally go in the bracket [forever loop] ...but I’m going to try it anyways*”. If participants did not think they could put the pinMode either underneath another pinMode or in the forever loop, it might have led a decision to create two chunks of code. Only once did a participant upload code without having any pinModes in their program.

The second issue with not switching the pinMode to *output* could have either been a slip since the block initialized as an *input*, or because they thought it should be an *input*. While the video footage indicated that majority were probably due to slips, some participants did struggle understanding the concept of input and output, *“yeah I still don’t understand this whole input/output thing but I trust that it says output”*. When this participant first learned about setting input and output she said that it would make more sense to her if they switched what they were calling inputs and outputs. Other participants could have had similar interpretations.

6.6.6.3 Delay Code Block Breakdowns

The delay code block was also a source of confusion, more so in the think-aloud section than in the non-think-aloud section. The wording in the initial guide for the think-aloud section caused many participants to think that the delay caused the blinking of the LED. As one participant reflected, *“hmm [reading from the guide] ‘we will now add a second digitalWrite but this time make it so we turn the LED off’ see like that’s where I’m kind of confused because if we want it to blink forever then why are we turning it off? cause like the delay is what makes it blink not off. Off turns it off, so if you want it to keep blinking why would you turn it off?”* This misconception was also seen as participants created incorrect code. For example, one participant put the delay in a repeat block by itself in order to get one of her LEDs to blink twice (see Figure 72).

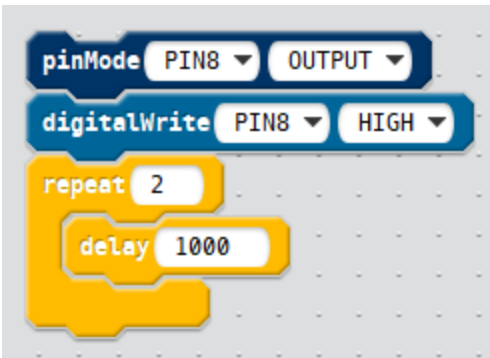


Figure 72 – Participant who had a misconception about the delay block thinking that it caused the LEDs to blink

The wording in the guide was modified to describe the blink differently, using switching on and off a light switch as an analogy, and removing the wording that led participants to believe the delay caused the blinking. After this change, the misconception surrounding the purpose of the delay was less prevalent in the non-think-aloud section.

The other common breakdown participants would make with the delay function, was thinking it would fix other issues in their setup. For example, the video captured participants changing the delay in order to: get one of their LEDs to turn on, change the order in which their LEDs were blinking, and to make an LED brighter when they had not initialized a pin as an output. Often times, when a participant would change the delay, it was indicative that the participant was stuck on an issue, and this was something they could do when they were at a loss for what else to do. As we will show in an example in section 6.6.7.1 even when acknowledging the flaw in their logic they sometimes would still change the delay to fix their problems.

6.6.6.4 Wrong Arduino Pin and Pin Signal High/Low

Setting the wrong Arduino pin and sending the wrong pin signal were two of the breakdowns that were usually caused by slips rather than conscious decisions. These slips introduced bugs into the code. Participants would often forget to change the pin that was being referenced, the signal that a digitalWrite was sending, or they would try and set it using the drop-down menu and accidentally choose the wrong selection. This was evidenced by situations such as a participant setting all the other code blocks to reference PIN8 and selecting one of the digitalWrites to reference PIN7 when nothing was hooked up to PIN7 (ex. Figure 73)

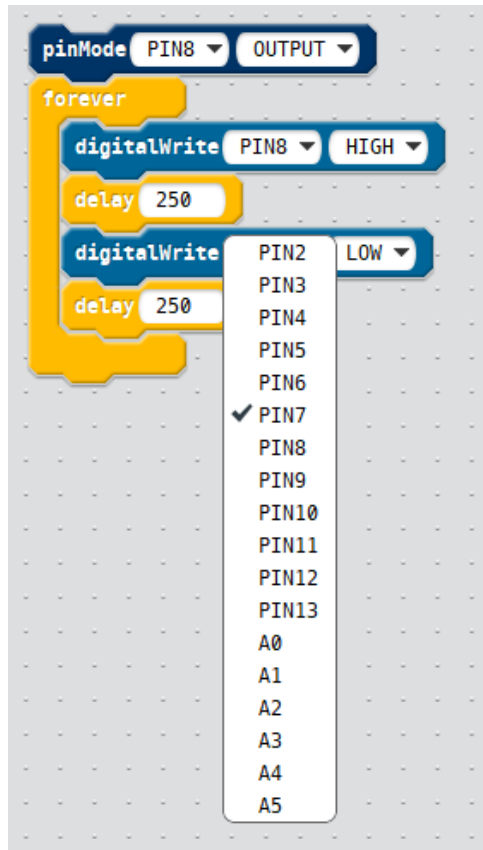


Figure 73 – Example of selecting the wrong Arduino pin from the drop-down menu

Participants, such as in the example below, would be thinking about their code correctly but then be more concerned with using the correct code blocks and introduce this error. For example, one participant had the code in Figure 74 below.

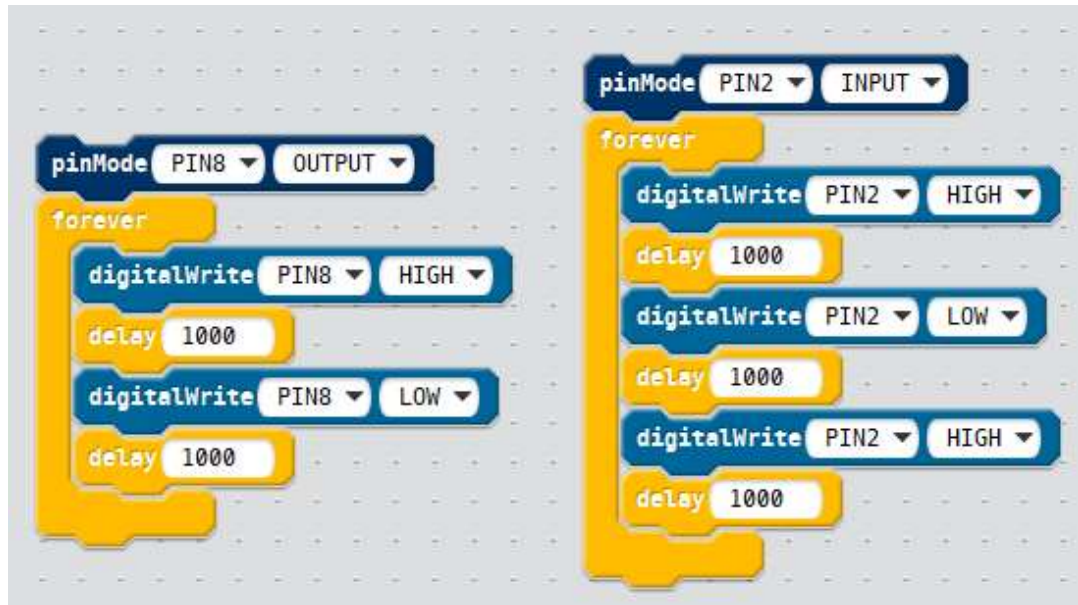


Figure 74 – Example Pin Signal Breakdown Step 1

She traces her code on the right with the mouse and says, “Turn on, turn off, turn on and turn off again”. As she places another digitalWrite and delay for PIN2 she says, “Turn off then delay again.” She neglects to set the digitalWrite LOW resulting in the code in Figure 75 below.

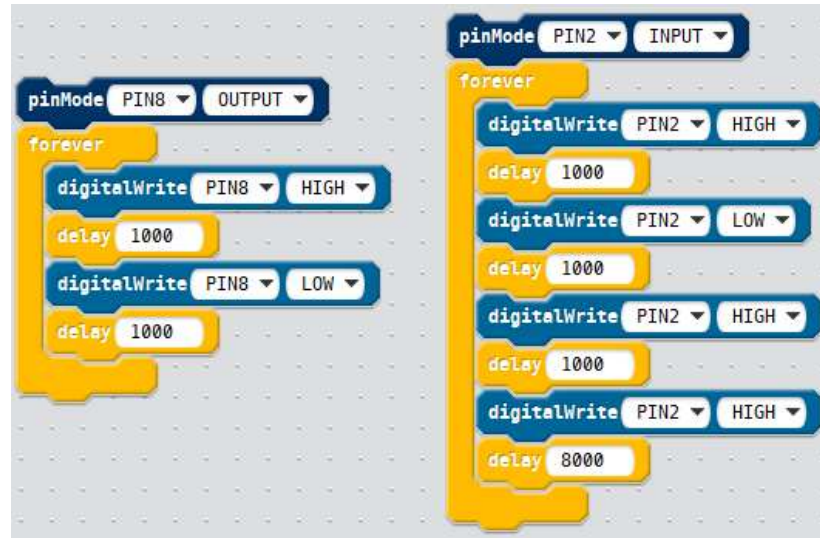


Figure 75 – Example Pin Signal Breakdown Step 2

The participant goes back through the code again, “*This would be turn on, turn off, turn on, turn off, delay then loop again.*” She uploads it with the pin signal set incorrectly. While this sometimes happened, usually if participants were seen reviewing the code, they would be able to pick up these mistakes.

6.6.6.5 Block Connection

The last software breakdown that was prevalent amongst the participant was a breakdown in the usability of the IDE for connecting code blocks. While participants had a few other obstacles using the IDE, this was the only one that consistently led to bugs in the code; furthermore, these bugs were often difficult to discern. The obstacle connecting blocks is illustrated well by one participant who was trying to connect the blocks for the first time in the guide’s tutorial for using Modkit. She dragged two blocks out and was attempting to connect them using the block above (Figure 76 (a)). The connection only works if you move the block below. After several attempts moving the block above the

participant says, “Ok that looks like a shadow [it wasn’t] ...That is not connected... [drags block under other block and sees shadow (Figure 76 (b))] ...ooooh! I think maybe that worked. I don’t know what I did [trying to connect from above again and it’s not working so she picks up the bottom block to connect them] oooooh that works.” The participants often faced difficulties connecting blocks in the activities following the tutorial. The fact that there are no visible indicators to show you when blocks are connected, led to participants thinking they had connected blocks when they had not.

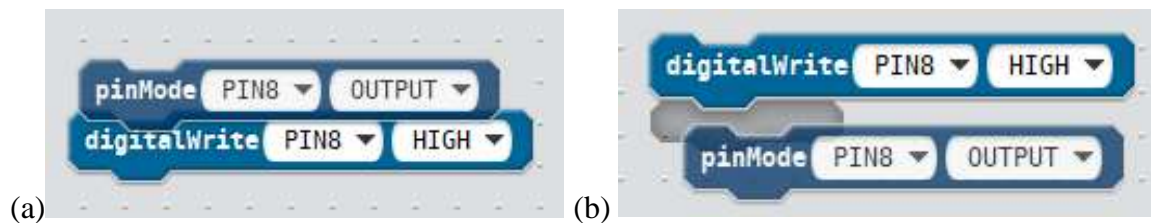


Figure 76 – Participant attempting to connect blocks (a) unsuccessfully and (b) successfully

While the software has an indicator when you are about to connect blocks (i.e. the shadow under the block as shown in Figure 76(b)), there is no indicator showing which blocks are actually connected. In Figure 77, the pinMode is connected to the forever loop in the left chunk of code whereas the one on the right is not connected. While it is possible to notice the slight offset of the blocks, it’s difficult to see and was often missed by participants reviewing their code for bugs. One participant who was debugging this exact issue eventually moved her blocks around fixing the issue, but she had no idea how, “something changed. I don’t know what I did...I don’t think I changed anything.” These findings were consistent with the barriers Boothe’s study of participants using the same software interface (Booth & Stumpf, 2013).

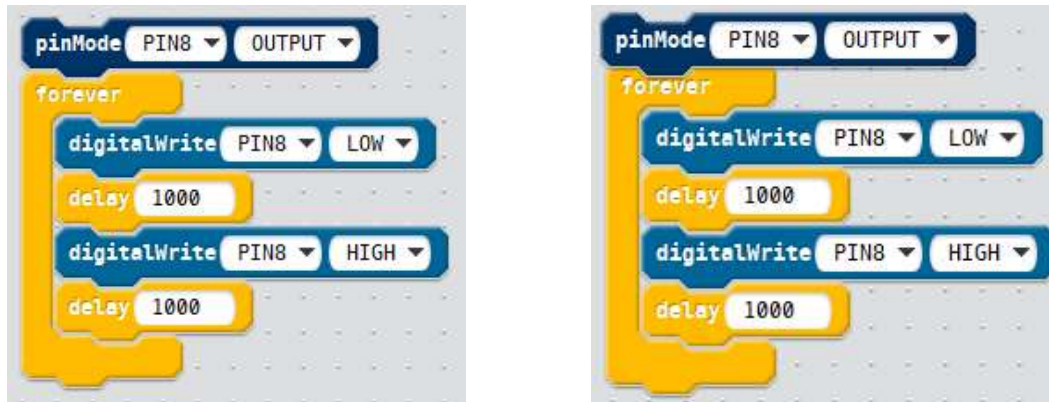


Figure 77 – Two pieces of code that demonstrate a connection between the pinMode block and the forever block (left) and code that demonstrates unconnected blocks between the pinMode and the forever block (right)

6.6.6.6 Exploring New Code Blocks

On top of the findings discussed above, a fair number of participants had varying success as they experimented with new code blocks during the tasks. In the guided activities, we only covered the usage of four blocks—pinMode, forever, digitalWrite, and delay—which were the only blocks needed to complete the activities. However, upon seeing other blocks that they might be able to use, participants often experimented integrating them into their code. A number of participants identified and correctly used the repeat block to get the first LED in Task #1 to blink twice. The fact that the block used language that was directly linked to its usage and it did not require any inputs made it so participants were generally able to integrate it correctly. This is similar to Weintrop and Wilensky’s finding that the repeat block was easy for their participants to understand (Weintrop & Wilensky, 2015c). The repeat block was the only new block that participants used correctly without introducing bugs into their code.

When participants tried integrating other blocks, such as the *while* and *and* blocks explored by the participant in the sequentiality example (section 6.6.6.1), there were breakdowns in how they were interpreting the blocks. In Kaczmarczyk et al.'s work they found that students would apply “real-world semantic understanding to variable declarations” (Kaczmarczyk et al., 2010). Unsurprisingly, our work suggests that this is also how they are initially determining the code blocks' functionality. Characteristics, such as the size and shape enabled participants to quickly rule out certain blocks, such as the *and* block which physically did not fit with the other blocks the participants were using. Blocks that were more similar, like the *while* and *break*, were more likely to find their way into the code participants uploaded to their Arduinos (see Figure 78 (left)). Slips were also made by participants who would accidentally pick up an `analogWrite` or the `read` blocks instead of a `digitalWrite` block (see Figure 78 (right)). Similar to the *and* block the `analogRead` and `digitalRead` blocks were often deleted from their code before uploading, but participants usually took a bit longer to notice that they had an `analogWrite` in their code.

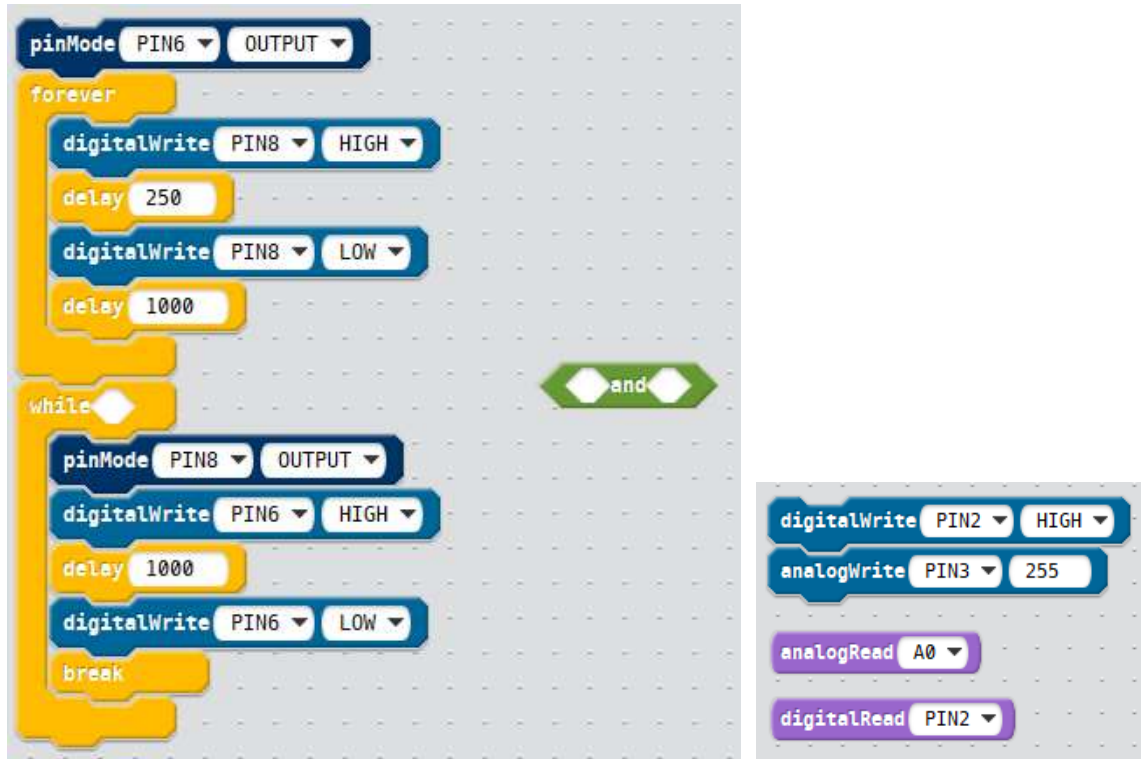


Figure 78 – (Left) Code demonstrating the shape differences between the *while*, *break* and *and* code blocks leading participants to integrate or rule out their use. (Right) Similarities between *digitalWrite* and *analogWrite* blocks.

6.6.7 Hardware+Software Bugs, Breakdowns and Obstacles

The hardware+software Bugs, Breakdowns, and Obstacles were intended to capture mistakes and misconceptions about the interaction between the hardware and software. The bugs resulting from this category were lower than the number of breakdowns because there was only one breakdown that was directly linked to the introduction of a bug. However, the mistakes participants made did reveal deficits that are important for understanding how a novice could be supported. Figure 79, outlines the hardware+software bugs, breakdowns and obstacles identified across the groups and activities. Similar to the hardware and software, the Blinky LED activity had a higher number of issues found in the participants using the Breadboard and BitBlox when it came to the hardware+software interactions.

While the participants using the BitBlox had a higher number of issues in Task #1, followed by the group using modules and then the Breadboard. Task #2 continued to demonstrate the difficulties faced by the participants transitioning from the Modules to the BitBlox.

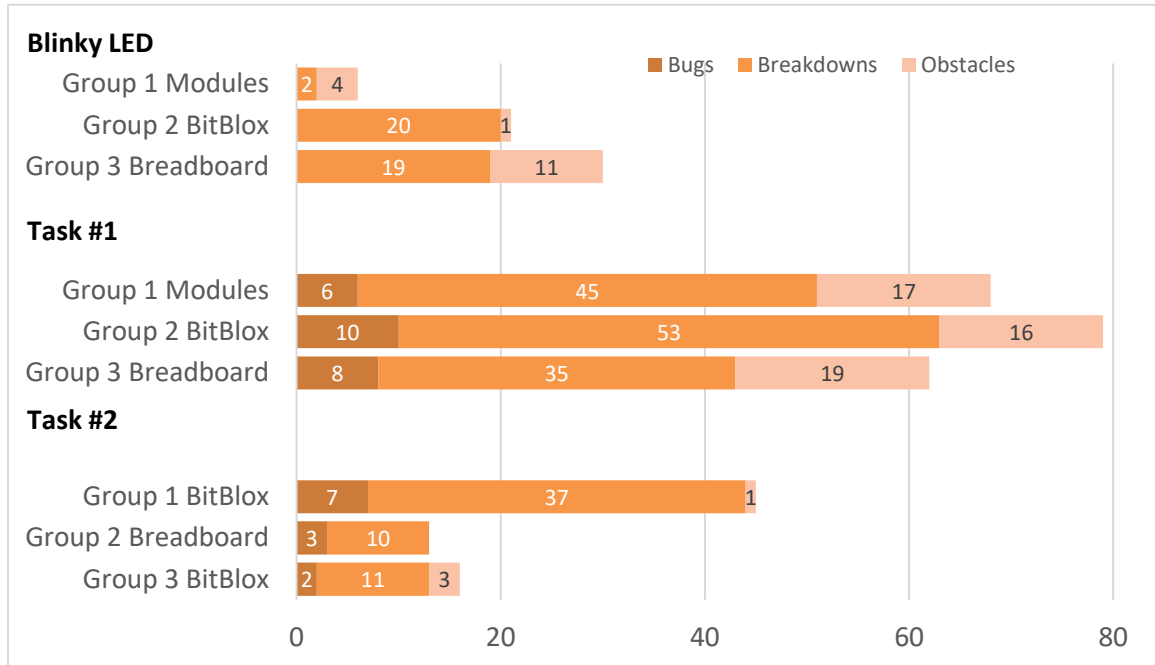


Figure 79 – Hardware+Software Bugs, Breakdowns, and Obstacles

There were only four breakdowns in this category since most of the errors participants made were predominantly in either the hardware or software. The four categories of prevalent breakdowns were: incorrectly assessing where a bug was in their setup, reprogramming the Arduino with the same code that was already uploaded to it, programming code to the Arduino without the Arduino plugged in, and incorrectly thinking a problem was solved when there was still a bug (see Figure 80).

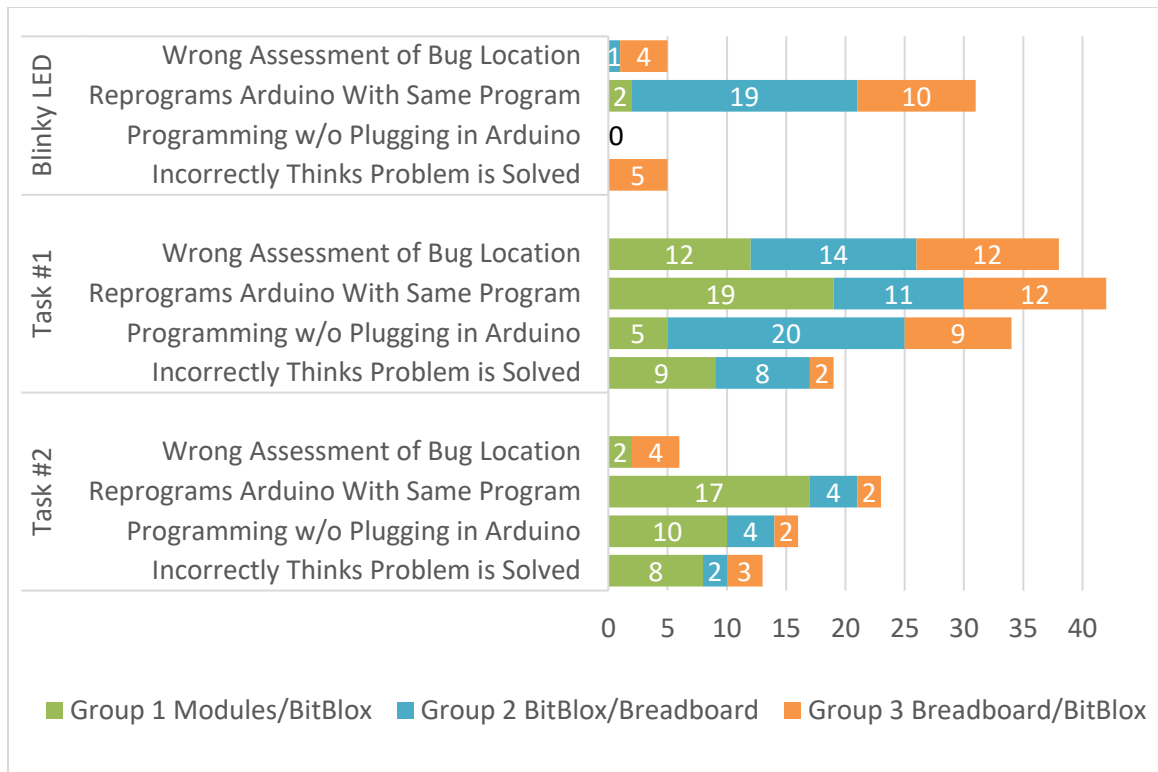


Figure 80 – Hardware+Software Breakdowns from Think-Aloud and Non-Think-Aloud Section

6.6.7.1 Wrong Assessment of Bug Location

The breakdown associated with incorrectly assessing the location of a bug did not introduce a bug but led to the persistence of bugs in participants’ setup. The issue usually occurred when the participant had a software issue and thought it was in their circuit. For example, when participants created two forever loops or placed one forever loop after another, it led to a program that would only light up one of the participants LED circuits. Not realizing their code was wrong, participants often assumed the issue they were having was in their circuit rather than with their code leading participants to make changes in their hardware. These changes were sometimes harmless, such as rebuilding their circuits correctly or changing the pins their circuits were connected to; however, sometimes the participants

introduced bugs such as reconstructing their circuit with the wrong circuit formation or changing the directionality of their LED just to make sure they had it correct. Participants often would guess and check certain configurations which relied on buggy code or circuits, rather than narrowing down their problem based on what was working.

One participant provided a clear articulation of his process debugging a pin that was not initialized properly. The example was chosen because the processes and mistakes he made were similar to the themes identified across the participants. The participant started with the code in Figure 81, which did not initialize PIN12 as an output, and was faced with debugging why the LED at PIN12 was dim. Both of his LED circuits were built correctly.

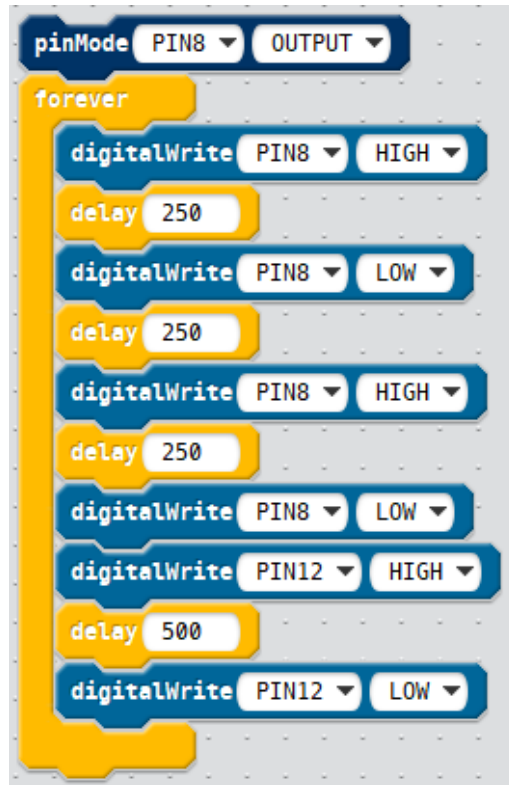


Figure 81 – Example Incorrect Assessment of Bug Location Step 1

As he begins to debug his setup, he goes straight for the circuit at PIN12: *“Let’s take a closer look [takes out the LED in his circuit] let’s see if this is wired correctly. Obviously if it weren’t wired correctly it wouldn’t work, but [continues inspecting his LED]...so flat end this definitely seems to be positive to negative like I said, so positive ground [plugging the LED back in correctly]. It’s wired like before. It still has the resistor and then the wire. Let’s try moving the ground over to see if that makes a difference. I’m not sure it will but let’s just test it, and see what we got [switches ground pin he is using, which wouldn’t make a difference], so let’s plug it back in. Ok, let’s give it a shot [he says uploading the code]. Ok so no change. This one’s still bright and this one’s still dim, so why don’t we try switching the LEDs and seeing if that makes a difference...ok [he unplugs the Arduino and presses stop to upload blank code]. So we take this out...[he switches the LEDs between*

the circuit at PIN12 and PIN8]. *Hmmm* [picking up circuit to look at it] *so there definitely seems to be an issue either with the programming or the wiring because this one, this one now that it switches but is going really brightly compared to the other one before when it was being very dull* [he unplugs the Arduino and stops the code], *so let's try to redo this portion of the circuitry and see if we can see what's wrong.* [takes apart the dim circuit] *ok just to make changes let's try using this one instead* [he changes the BitBlox he is using] *just because, so let's start by navigating...*[connects the circuit correctly with new BitBlox]...*Attach this* [plugging in the USB cable and uploads the code]. *Ok* [looks at the circuit that's still dim].” The researcher provides a hint that the problem is in his code. “*Ok, let's take a look so writes HIGH writes LOW. We have the PIN8 components are working properly, and then this is plugged into PIN12* [verifying in his circuit] *so the issue is not there. Let's take a look at this more closely* [pointing at PIN12 code], *so we have a write to PIN12 HIGH, wait, write to PIN12 LOW...hmmm* [picks up delay block from menu] *let's see* [places a delay between turning PIN8 low and PIN12 high], *so we've got a little bit of a gap in between* [changes delay to 500ms] *although that shouldn't change anything...write HIGH, write LOW, and then after that the delay just means, no delay just means it seamlessly goes back into flashing twice before repeating. It might be possible that the delay between the two is affecting the brightness, so let's change it to 250 as well and see if this affects things* [changes delay and uploads code to Figure 82].

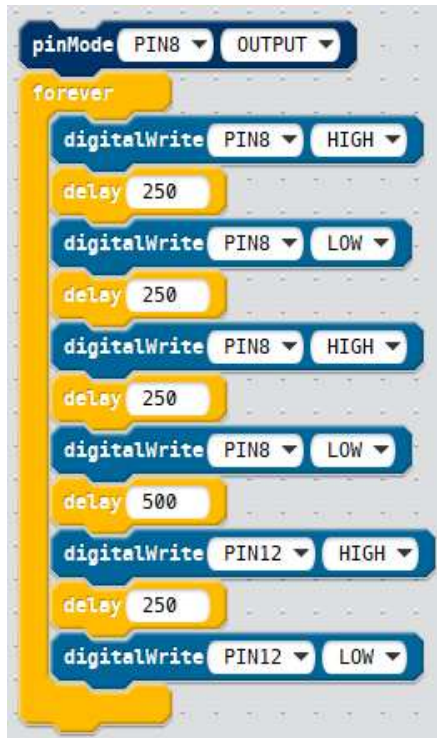


Figure 82 – Example Incorrect Assessment of Bug Location Step 2

Hmm alright whelp...this doesn't seem to have helped, so stop this [presses stop]. Let's see why could that be the case?" The researcher gives him another hint that he is doing something with pin 8 that he is not doing with pin 12. He then introduces a bug by adding another digitalWrite high for PIN12 which will make the LED turn on the whole time the LED at PIN8 is blinking (see Figure 83).

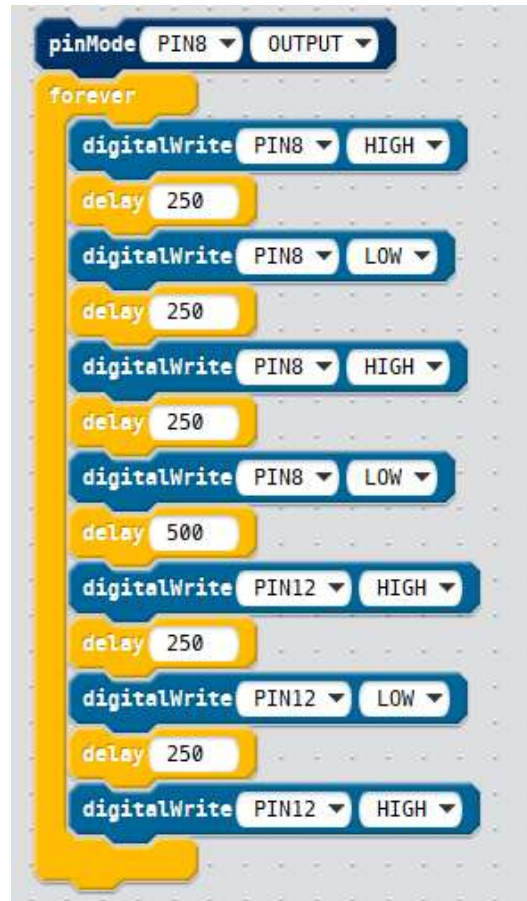


Figure 83 – Example Incorrect Assessment of Bug Location Step 3

When it doesn't work he goes back to debugging his circuit despite being told that his error is with his software. The researcher eventually identifies his specific problem by telling him to look at his first block of code. He then realizes his mistake and fixes his bugs to complete the task.

The example above demonstrates several themes we saw in how participants would debug their code. First, the participant tried things that ran counter to his beliefs of how the circuit and code worked, such as checking the LED orientation (which he stated had to be correct because the circuit was blinking) and changing the delay (which he stated should not make a difference). The participant tried harmless fixes in the hardware such as

rebuilding his circuit using different BitBlox and ground pins. Other researchers have noted these debugging strategies as *Just in Case* in which students make unnecessary changes just to check, and *tinkering* making “fairly random and usually unproductive changes” (Murphy, Lewandowski, & McCauley, n.d.). He also introduced bugs, such as adding another digitalWrite high to his code. Furthermore, he was fixated on the hardware as the problem space. Even after the researcher told him the error was not in his hardware, he did not know what else to do in his software, so eventually returned to debugging his hardware after introducing an error in his code. Not all of these behaviors were inherently bad; for example, testing your assumptions is important for debugging. However, the lack of a systematic and directed debugging process made it inefficient and unsuccessful.

While the participant above was clear in his direction for debugging the hardware, other participants were faced with instances where they discussed their confusion over where the issue was. One participant in the non-think-aloud section who had an issue with her pinMode not being attached to her forever loop wanted a hint to solve the problem. When asked what type of hint she wanted, she said, “*software....it doesn't matter haha any one I don't really know.*”

Other participants were confused if the circuits were responding the way they intended with their code. For example, if the participant mixed up how they were turning the LEDs high and low so that one was turning on before the second one was off it became a difficult problem to reason about especially because the code was in a loop. Fast blinking LEDs also exacerbated participants issues discerning what was happening. The following exchange happened, when a participant had the code in Figure 84 with short delays of 50ms and 100ms long:

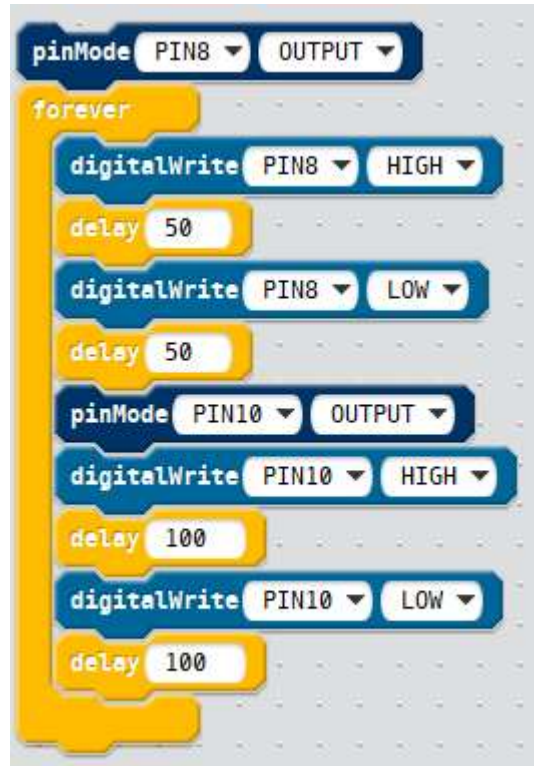


Figure 84 – Code from participant who could not tell if her code was blinking one of the LEDs once or twice because it was changing so rapidly

[Participant] *They are both blinking but I don't think that it's doing the right thing...*

[Researcher] *How come?*

[Participant] *Well I mean, I thought. Let's see here. I thought this was [going back to the task question in the guide] one of the LEDs should be blinking twice and the other one says it should. Then the other one should blink once, but I can't tell if I'm doing that right though.*

Her code was going too fast for her to know if the first LED was blinking twice or once before the second LED. While the answer could be found in her code, she was trying to use the feedback from her circuits to understand what was going on in her code.

6.6.7.2 Reprograms Arduino with Same Program

Reprogramming the Arduino was a harmless breakdown and perhaps avoided bugs that could have occurred had participants not been as diligent in uploading code to their Arduino. However, it became clear that participants did not realize the Arduino stored the code on the board after it was uploaded. Several of the participants used the same pin that was used in Blinky LED for completing Task #1 (i.e. PIN8). When they plugged in the Arduino and their LED started to blink, several indicated their surprised; as one participant said, “*oh! That one’s already blinking...cool, okay now I just have to* [turning her attention to the code] *oh it’s because I didn’t remove this* [pointing at the code on the screen with her mouse] *that’s why*”. While the guide did mention that the Arduino stores the code, it is not something that would be reinforced since the participants had the Arduino tethered to the computer throughout the study.

Reprogramming the Arduino consecutively without changing anything in the circuit or code, was often seen in participants who were stuck and churning on an issue. Reuploading the was often a last-ditch attempt just to see if it would work when they were not getting the results they wanted. One participant uploaded the code three times in a row within a few seconds because his circuit was not lighting up as expected. Other investigations in CS education found that novice debuggers would recompile code even though they had not changed anything (Murphy et al., n.d.).

6.6.7.3 Programming without Plugging in the Arduino

The third breakdown participants experienced between the hardware+software was a breakdown programming the Arduino without plugging it into the computer. This was

the one breakdown that was marked as introducing a bug and could be relatively harmful to the participants success if it was not identified quickly. While the Modkit software does indicate there was an error programming (see Figure 85) this was not always seen by participants and if it was, it was sometimes interpreted incorrectly. Participants sometimes thought the code was wrong, rather than realizing the need to plug in the Arduino. Participants that were not able to catch the error quickly iterated through several versions of their code not knowing what was correct or incorrect. Not getting the expected feedback from their circuit skewed participants' interpretation of what would work in their code. One participant went through six different iterations of her code without uploading any of them properly to the Arduino. At one point during these iterations she had the correct code, but never realized it and ended up not being able to complete the task.



Figure 85 – Error participants would receive if they did not plug in their Arduino when programming it

Some participants, however, added to the number of hardware+software breakdowns by pressing the *stop* button (which uploads blank code) without waiting for the program to finish uploading. This accounted for 8 of the breakdowns that were subsequently fixed when they plugged in the Arduino and uploaded the code.

6.6.7.4 Incorrectly Thinks Problem is Solved

The last category of hardware+software breakdowns was incorrectly thinking the problem or task was solved when it was not. This generally was because participants would forget to initialize one of their pins as an output. They would see the LED dimly blinking and think they had completed the task. They were told that they still had an error because the LED should not be that dim. This often resulted in them debugging the circuit instead of their code because they could see the error in their circuit resulting in a *wrong assessment of bug location*.

6.6.8 *Non-Think-Aloud Section Extra Interview Questions*

The data from the short post-study interview served to complement the information gathered during the study. Because the participants in the non-think-aloud section did not complete a think-aloud protocol, they answered two extra interview questions that could help provide an understanding of how they were thinking about the material. The first question, was the Button Blink Question and the second was the BitBlox Circuit Debug Question. I'll describe the questions in more detail in the sections below and report on how the participants did.

6.6.8.1 Button Blink Question

The Button Blink Question is as follows:

If you were given a button, an LED and resistors, how do you think you would create the circuit and code so that an LED blinks twice when you push a button?

You can explain it at a high level.

Overall, the participants struggled with the question. It was hard for the participants to articulate their thoughts as they struggled with the vocabulary for all of the parts:

I'm assuming the programming would be similar except you'd take um you'd have to plug the button into one of the um one of the um, sorry I'm forgetting names.

You'd have to plug the button into one of the um digital slots.

The participants in the comparative classroom study also struggled with applying the correct terminology with the amount new vocabulary they were presented with when using the Arduino for the first time (see section 5.3.4). All of the participants who worked through the study did so by tinkering with the Arduino to get it to work properly. Asking them to simply talk about it without being able to tinker was inauthentic (Shaffer & Resnick, 1999) in comparison to how they had learned. Participants commented on the need to actually be able to play with it in order to figure it out:

I'm more of like a doer and player than like explainer.

Some participants were not able to answer the question at all or stayed at such a high level that it was unclear what they knew. For example, one participant said:

I have to first connect the circuit and connect to the computer and the USB to the software. And make a code and upload it and it will work.

When the researcher probed the participant for more information they were unable to say anything more concrete. Those who could give a more comprehensive answer often still struggled to determine that the button should not be directly connected to the LED. Out of the 20 who answered the question, only 6 identified the need to separate the button from the LED circuit. As one participant put it,

I would try and connect the Arduino to the led and maybe try and connect that to the button and then hopefully it would, I don't know be able to control it by the button.

Out of the 6 who did identify the button as needing to be separate 5 were able to correctly identify it as an input rather than an output; one swapped the term *analog* for *input*.

About half of the participants (9 out of 20) identified the need for conditional logic without being taught about it in this session. A couple participants recalled seeing an *if-block* in the software. For example, one participant said,

If you press the button the light would blink so um the light...it'd be on high then a delay and then low and then another delay, but that's only if the button's pressed so I guess, I don't know, I saw an 'if' thing so maybe that would work?

Most participants talked more vaguely about the need for conditional logic, such as this participant who said,

If you put the pinMode as like an input so when it's like triggered or something then it goes on the cycle of referencing um pin whichever over here on the digital side to use a blink twice so reference it to the HIGH input delay and then same pin LOW input and then delay and then that would be it. Yeah 'cause it wouldn't be going forever

A couple participants had fairly comprehensive and specific answers such as this participant,

So I'd like create two circuits one with the button in it and one with the LED in it...then I'd have to make the button circuit I think an input in the code so it'd be like whatever pin that's connected to would have to be an input, and then I'm not sure, I'd probably have to play around with it to make it work exactly, but I think I'd like have...I'd have to find some sort of if-then relationships in the in the code, so that like if when the button's pressed then I like do this same thing that I had setup before but like a twotime blink instead of a forever blink.

Although the participants realized that the LED should no longer be blinking forever, no one identified the need for a forever loop in order to check for the button press.

6.6.8.2 BitBlox Circuit Debug Question

The BitBlox Circuit Debug Question was designed to provide the participants with the opportunity to identify both code and circuit bugs to see how they approached the problem. The participants were given a sheet of paper with the following instructions and code that the researcher walked the participant through:

What's Wrong with The Code and Circuit?

GOAL: The code and circuit are supposed to blink the three LEDs in this order forever:

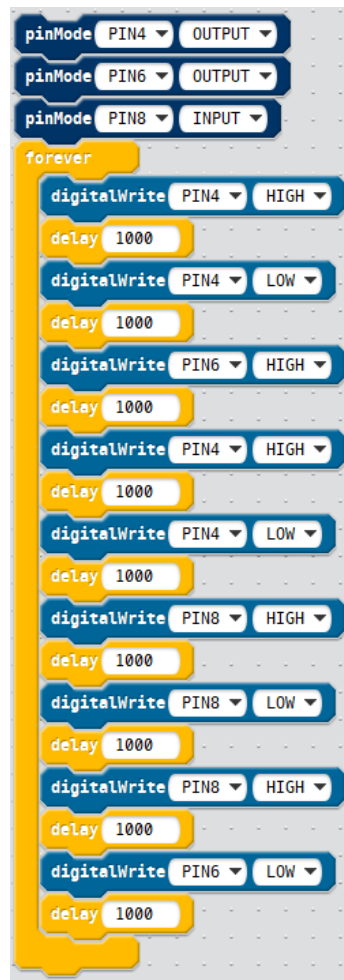
LED **PIN4** -> LED **PIN6** -> LED **PIN4** -> LED **PIN8** -> LED **PIN8** ->

LED **PIN4** -> LED **PIN6** -> LED **PIN4** -> LED **PIN8** -> LED **PIN8** ->

(continue repeating forever)

QUESTION: Can you tell me what the problems are in the circuit and/or the code?

CODE:



```
pinMode(PIN4, OUTPUT);
pinMode(PIN6, OUTPUT);
pinMode(PIN8, INPUT);

forever {
  digitalWrite(PIN4, HIGH);
  delay(1000);
  digitalWrite(PIN4, LOW);
  delay(1000);
  digitalWrite(PIN6, HIGH);
  delay(1000);
  digitalWrite(PIN4, HIGH);
  delay(1000);
  digitalWrite(PIN4, LOW);
  delay(1000);
  digitalWrite(PIN8, HIGH);
  delay(1000);
  digitalWrite(PIN8, LOW);
  delay(1000);
  digitalWrite(PIN8, HIGH);
  delay(1000);
  digitalWrite(PIN6, LOW);
  delay(1000);
}
```

Figure 86 – Code in Circuit Debug Question in post-study debrief with non-think-aloud section

The participants were then given the following Arduino circuit, which they could plug into the computer:

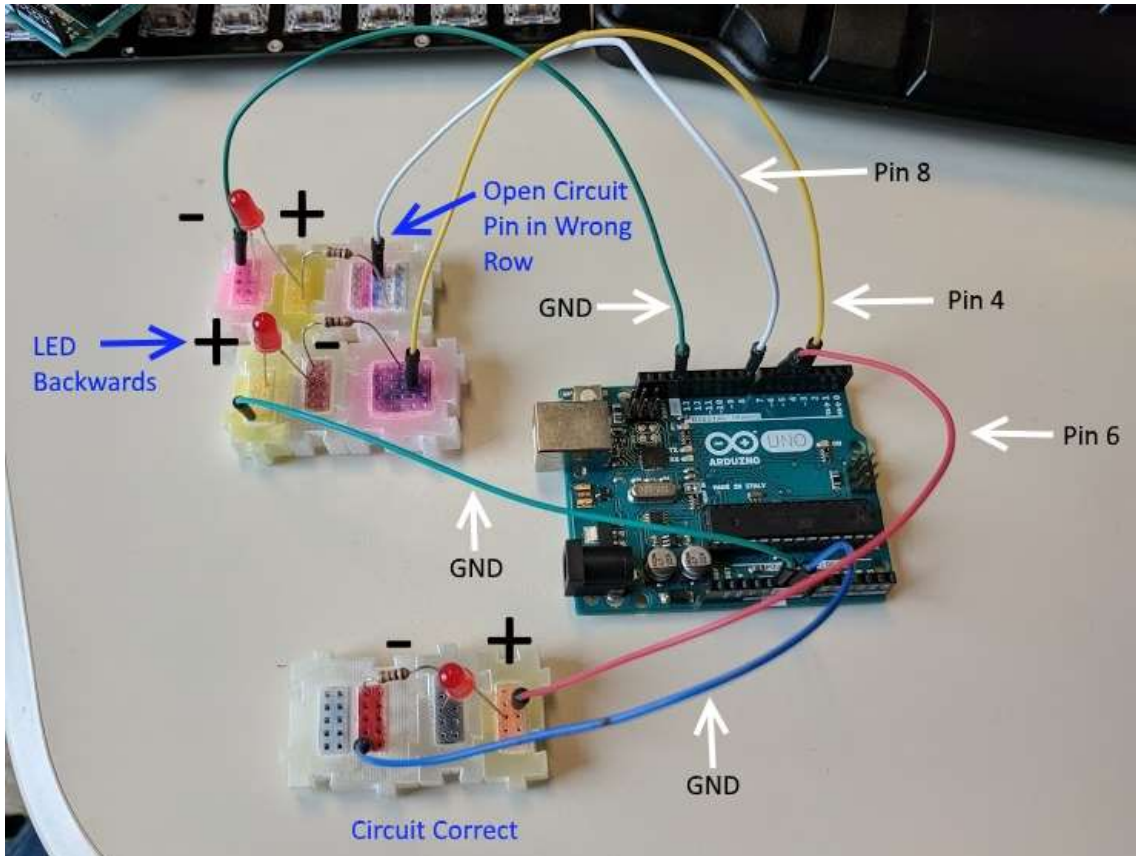


Figure 87 – BitBlox Arduino Circuit in Circuit Debug Question in post-study debrief with non-think-aloud section

The setup had three code bugs and two circuit bugs:

Code Bugs:

- (1) Pin 8 is initialized as an INPUT instead of an OUTPUT
- (2) Pin 6 is missing the digitalWrite LOW
- (3) Pin 8 is missing the digitalWrite LOW which is incorrectly assigned to Pin 6

Circuit Bugs:

- (1) LED circuit at Pin 8 has an Open Circuit
- (2) LED circuit at Pin 4 has a Backwards LED

Overall the participants were able to catch the majority of the bugs. Table 35 reports on the bugs identified by the participants in each group.

Table 35 – Bugs Identified by Participants in BitBlox Debug Circuit Question

	<i>LED Backwards</i>	<i>Open Circuit</i>	<i>pinMode set as INPUT</i>	<i>Missing Pin 6 LOW</i>	<i>Missing Pin 8 LOW</i>	<i>Total %</i>
<i>Group 1 (N=5)</i>	3	3	4	5	4	76%
<i>Group 2 (N=7)</i>	5	5	5	7	6	80%
<i>Group 3 (N=8)</i>	2	6	8	7	7	75%
<i>Total % (N=20)</i>	50%	70%	85%	95%	85%	

All groups performed about the same on the debugging task. Group 2 who started with the BitBlox, performing slightly better than the other two groups catching an average of 80% of the bugs compared to 76% (Group 1) and 75% (Group 3). The hardware bugs were the least spotted bugs with the LED backwards only being caught by 50% of the participants, and the open circuit only caught by 70% of the participants. The bug that was caught the most was the need to set pin 6 LOW in the code, with only one participant missing this bug—95% of the participants caught it.

Aside from correctly identifying bugs, participants also falsely identified bugs. In the code, participants often thought that the pinMode needed to be inside the forever loop before each digitalWrite. After one of the participants was provided with the correct answer he reflected,

I didn't realize they were settings I thought they were orders, so I thought that if I needed to switch from the current output being pin 4 then I would have to put it in to say like 'ok, pin 4 was the output now pin 6 is the output'. I didn't know multiple things could be [set] but that makes sense now.

In the circuit, participants falsely identified bugs because of sequentially reasoning about the circuit thinking that the LED needed to be next to where the circuit was receiving power instead of the resistor. When they *fixed* this bug they often also fixed the Open Circuit bug and the LED Backwards bug so therefore thought their assessment was correct. In one case, the researcher asked the participant to try switching it back to see if it actually mattered, as the participant did this she flipped the LED backwards. The researcher then identified the participant's error. This example demonstrates how a misconception can propagate when wrong assumptions are made about what matters and what doesn't as a participant is debugging a circuit.

Participants in Group 1 and 3 often falsely identified open circuits in the hardware based on thinking that the row/column of the BitBlox mattered, even when they were in the same section. The participants would switch the components around in the circuit in order to ensure there were no open circuits even when the LEDs were blinking. While their reasoning was incorrect, it ensured they correctly identified the one open circuit that did exist. This error was not made by anyone in Group 2, who started with and thus had more experience with the BitBlox.

6.6.9 *Reflection on The Experience*

In the think-aloud and non-think-aloud section, the post-study interview provided an opportunity for the participants to reflect on the experience. Some questions were open ended while others required the participants to reflect specifically on the tools they were using. Findings spanned the participants' perceptions of three main categories (1)

electronics and programming, (2) the hardware tasks and tools, and (3) the software tasks and tools. I will go through each of these breaking it up by group where it is appropriate.

6.6.9.1 Participant Perspectives on Electronics and Programming

In the interview, a persistent theme across all the participants was how they viewed this experience as existing outside their comfort zone. The participants described coming into the study viewing electronics and programming as, *nerve wracking, scary, daunting, intimidating, imposing, challenging* and *stressful*. Furthermore, most did not self-identify with the content, saying things like “*I’m someone who’s always considered myself to be really bad with technology*”, “*comp sci, it’s just not me at all*”, and “*it’s just that biology is more my thing*”. These findings aligned with an observation from the researcher, who had to remind the participants that they were supposed to be beginners and that it was okay if they did not know the answers on the pre-test, which often elicited negative feelings, as one participant put it:

It was just a flashback to junior year of physics and the very first thing when I had to draw the [circuit] I was like ‘uh Mr. [Palmer] is gonna be so disappointed in me’.

One participant even excused herself from the study when she realized she had no idea how to draw a circuit in the pretest. She stated:

*I worry that later even with the instructions I might not [be] able to solve the problems, so I worry that I will embarrass myself even more...maybe my husband can do this, but I can't do this. I'm sorry.*²

This participant was a graduate student at a top tier college and was still afraid that she was not going to be able to learn about electronics. Another participant had such a strong aversion to the material that she did not want to try building anything as she went through the guide. The participant went through the guide without putting together the alligator clip circuit or her code or circuit for the Blinky LED activity (even though she was reminded that the parts were there for her to put together the circuits). When the researcher informed her that she had to complete the Blinky LED activity she was visibly reluctant to do so. She proceeded to build the circuit and then clicked past all of the slides that showed her how to make the code. As the participant tried to move on, the researcher told her, “*You have to actually make it blink*” to which she replied, “*like this?! [pointing at her circuit in disbelief] Oh...would it actually blink?*”. She went back to through the slides and ended up completing the Blinky LED but was unable to get through the first Task. She was one of the few participants who had a strong negative reaction to the experience throughout. She scored a 1 on the posttest knowledge test and was 2.5 standard deviations below all other participants in her group in both the knowledge and self-efficacy scores. Debriefing on the experience she said it was *awful* and *stressful* because she is “*not inclined towards anything math or physics related*”. When asked why she did not want to put the circuits together she said,

² Quotation used with the permission of the participant

I just like have no interest in it...it's just my dad's an engineer and an electrical engineer and he used to make our science projects with these things and I never enjoyed it and I've been very like...like I'm graduating with a counseling and business degree

While those two participants completely rejected working with the content, the majority of the participants commented on how their perspectives and feelings changed:

I've seen one before but I was just too scared to get near it so I never really learned anything. I never got to do anything with it, but you know I would do it [now].

Another participant remarked on his experience:

I thought it was cool I mean I never really used or made my own circuits or used a board like that or anything so I mean it's not as hard as I thought it would be I guess, so it makes me feel more confident in the future.

While the participants felt more comfortable with the material, they also still had some hesitations and felt they had a long way to go before being able to work with the Arduino on their own. One participant reflected on the knowledge they gained:

I felt stupid at first. I mean I did learn some stuff though like I actually learned something, which was nice and felt less stupid afterwards, but yeah this isn't the most intuitive thing for me.

Participants talked about how they would only be able to work with the Arduino again if they knew they would have the proper support:

If it's guided then it's ok...and like guided properly. Like the guide has to be beginner friendly you know?

Having a desire to want to work with the Arduino was not only tied to whether they would have the proper scaffolding, but also if they found they could do something relevant, useful or interesting. For example, one participant stated:

I thought it was really awesome like you can use it to monitor the garden water cause I'm in organic and sustainable crop production, so the thought of like being able to like monitor the moisture in the soil through something like this is really cool to me, so like maybe I could try and do that in the future.

Others did not find any immediate relevance; as one participant said,

It's pretty fun... I would do it and I wouldn't be worried about it. Like I don't know if I would like do it for fun you know...I wasn't necessarily...like 'oh look at this amazing thing I've done.' I've made a thing blink three times.

6.6.9.2 Participant Perspectives on Hardware

We were also able to understand the participants' perspectives on the hardware. The participants often found that working with the hardware could be a tedious task; as one mentioned,

I mean it's just pretty uh meticulous really like slow when you have to make sure you're plugging everything into the right place. Putting everything correctly on

there you have to like check every little thing so I mean that's just kind of hard cause there is a lot of stuff going on"

Furthermore, they were often surprised at the small size of the components, which made them difficult to see and deal with:

"At first I was surprised about how small [the components] were and how like accurate you have to be with the wires and with the little ports and it was really tiny.

The LED was often specifically called out for its lack of usability:

Like I had no way of knowing...the longer side is positive or that it mattered. Especially cause like with the resistor that's the weird thing. The resistors look like the ones that matter 'cause they have the colors on them like these stripes so I literally was assuming oh one of these sides is specifically you have to go through it because of these stripes that are not in the middle, but like on one side and then [the LED] looks the same almost on both sides. Like this is ridiculous to tell...there's a slight flat part that's insane

On top of understanding the similarities between the participants' overall perspective on the hardware, the various groups enabled us to understand the similarities and differences on the prototyping tools they used. The order that the participants were introduced to the tools impacted how they felt about them, with the majority preferring the one they were introduced to first. Group 1, for example, used the Modules then transitioned

to the BitBlox and these participants generally preferred the Modules over the BitBlox. Participants commented on the usefulness of the labeling in the Modules:

Yeah [the modules] were like pretty self-explanatory cause it kinda like shows you where everything goes.

The participants consequentially found BitBlox more difficult in terms of figuring out where the circuit components should go and how to create connections between them. As one participant remarked:

“the most confusing one was [the BitBlox] haha um yeah cause like the [variable modules] was...connecting directly to something else so it was a lot easier for me to like, I guess, like do than doing it this way (pointing at BitBlox) where you had to kinda connect the dots and then make sure that one wasn't in the other one and like trust that if they were in the same pile that they were going to receive the same energy from each other and like respond to the same um power or voltage or whatever at the same time like that was a little like confusing to setup so once it did work I was happy that it actually worked but like the whole time I was just like I don't even know if this is going to work.”

The participants did however comment on some of the usability issues that they faced with the Modules. The biggest complaint was surrounding the difficulty of plugging in the wires to the pin headers and making sure they were pushed down. One participant stated:

[The modules] make life easier...Yeah until you get the thin wires on the resistors. One of them kept bending and I was like 'nooo!'

The participants in Groups 2 and 3 were able to compare their experience using BitBlox and the Breadboard. The participants in Group 2 generally preferred the BitBlox over the Breadboard. They felt that it was easier to understand because the connections were clearer. As one participant put it:

I mean I think [BitBlox] kind of shows like the connections clearer because of like, I dunno, just like the separate pieces versus [the Breadboard] where I think it's a lot easier to get confused cause they're just like all in a line and then remembering like that these are rows and these are columns and I guess not really knowing why you would set it up like that but that it is.

Participants also remarked on the usefulness of the BitBlox spreading out the circuit,

I liked working with these [BitBlox] more than the [Breadboard] because like I just have big fingers. It's a lot easier to plug it into these than to stick it into the little holes on the [Breadboard].

However, some participants had mixed feelings because the individual pieces of the BitBlox posed their own usability problems. For example, one participant stated,

I think the Breadboard was like simpler. Um just because well [the BitBlox] kind [of] just kept falling apart...I dunno the color [in the BitBlox] is helpful because in the breadboard it doesn't have any color, but I just felt like it's just kind of unnecessary to have all of these like small little pieces that you're piecing together.

In contrast, the participants in Group 3 preferred the Breadboard over the BitBlox. Similar to some of the acknowledgements from Group 2, they found the Breadboard had a

cleaner design because it had fewer pieces making it easier to work with and understand.

One participant stated,

“I felt like [the Breadboard], I don’t know, was easier to understand. [The Bitblox] has so many colors and blocks and things I wasn’t sure if I was doing it right, or if I like misread the instructions for that ‘cause I sort of just jumped into it.”

Some of the participants using the Breadboard still had difficulty with the connection scheme, such as this participant who said,

the Breadboard is a little more confusing because with...um BitBlox you connect one and this is one and you connect another one it’ll connect another. It’s relatively straightforward, but [the Breadboard] you have to keep the actual columns and rows and, I don’t know, like those columns and rows actually differ. It’s a little more confusing.

While the participants in Group 3 definitely had their preferences, they were more inclined to suggest different use cases in which either the Breadboard or BitBlox would be better suited. For example, one participant stated,

I liked [the Breadboard] better but I think it only works for like simple tasks because...obviously you could do it for a billion different things, but for doing this circuit (the one used for the debugging task) I would have to work really hard to make sure that everything is connected right [in the Breadboard]. I might accidentally bump one and knock it off to the wrong spot and I’d be like doing wire

over wire because there'd be like you know like twelve or some fifteen components like stuck in [the Breadboard] and it would just be difficult.

6.6.9.3 Participant Perspectives on Software

The participants across all groups had varying opinions on the difficulty of programming the software. The participants who found it difficult, thought it was challenging to apply what they learned to new situations and to figure out how to get the code to do what they wanted. One participant talked about the difficulty of transferring what she learned to a new problem,

like in [Blinky LED] we were only like taught to do a specific circumstance so it was kind hard being not familiar with programming to figure out like what uh applied in other situations. Like if the delay was put before it how would that change it, or if it would.

Another participant discussed the difficulty of having to understand what is going on in the code in order to get his LEDs to blink,

the most challenging part was for me was the code part...cause when you use the code and have to actually know what each part does..I have to learn like...how the code is setup and how it correlates with what cut on first and what cut on second, so I feel like that was the hardest part.

Although, a set of the participants found the programming hard, they still appreciated the blocks-based language. One participant stated,

I thought it was going to be something a lot more complicated, but it's just as simple as dragging and dropping icons and then selecting parameters for each icon that reference what you made.

Furthermore, while several participants mentioned that it looked like a tool for kids, unlike what (Weintrop & Wilensky, 2015a) found in their study, the participants who remarked on it did not regard it as a bad thing,

The coding software was really cool. I was expecting to...type up tons of lines, but this...felt like one of those educational games that I give to kids in elementary school to like teach them letters and stuff except its teaching you how to code. It's cute, it's like a cute little format, and it makes sense how the blocks work and you can move them.

However, they did view it as a starting point,

If it was my goal to learn how to code I would want to start translating pretty quickly from this to like actual programming languages.

6.7 Discussion

The lab study findings presented above demonstrate insights into the tools and experiences of novice participants working with physical computing for the first time, contributing to our understanding of the following research questions:

RQ2. *What are novice students' experiences when working with the Arduino for the first time?*

2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?

- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?
- 2.4 What are novice students' reflections on working with the Arduino?

RQ3. How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?

- 3.1 What characteristics of the hardware and software tools impact their usability in individual and group settings?
- 3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?

Within this discussion I will first cover how our findings of the obstacles, bugs and breakdowns integrated into the findings from previous studies. Next, I will discuss some of the processes of solving errors and correcting misunderstandings. Last, I will talk about the implications of our findings in relation to the design of the tools for physical computing.

6.7.1 Findings Situated with Prior Literature

In section 4.6, we introduced literature that could be useful for understanding how participants would be thinking about the Arduino in our laboratory study and specifically the misconceptions and errors they might encounter. I will go through the findings concerning the obstacles, breakdowns and bugs participants encountered in the study and discuss how our work is situated within the prior work answering RQ2.1 and RQ2.2.

6.7.1.1 Summary of Hardware Errors

The hardware errors were made up of six main categories: using the wrong Arduino pin, creating a short circuit, creating an open circuit, assessing the LED directionality incorrectly, leaving out a component in their circuit, and forming their circuit incorrectly so there were components either in parallel or series when they should not be. We found

that using the wrong Arduino pin and using the LED backwards was often the result of the small components causing a slip in the participants actions, or in the case of the LED causing participants to forget that orientation mattered. Similarly, our comparative classroom study (DesPortes, Anupam, et al., 2016) and the work of Sadler et al. (Sadler et al., 2017) and Booth et al. (Booth et al., 2016) demonstrated how small components caused similar issues. Sadler et al.'s study with high school students also found that their participants would flip the LED backwards, and Booth et al.'s study of intermediate adults, identified miswiring the temperature sensors and LEDs to the wrong Arduino pins as the most common errors (Booth et al., 2016). In Booth et al.'s case, these errors were often *fatal*, inhibiting participants to complete the task. Our results from the debugging task showed that catching hardware bugs with the small components were some of the harder issues for participants to spot, with only half finding the backwards LED. However, our results also show that small changes to the tools, such as using an Arduino with better labeling on the pins, could increase usability and decrease participants' errors.

Other hardware breakdowns we identified could be related to the prototyping tool design and conceptual misunderstandings participants harbored. Some participants using the BitBlox interpreted the colors as meaning everything should be plugged into one connection section, which then shorted everything together. This short circuit issue with the BitBlox was one that we identified in the prior comparative classroom study (DesPortes, Anupam, et al., 2016). In contrast, the design of the Breadboard increased the likelihood that participants would make slips causing open circuits because they did not plug the component into the right column or row. The usability issue was prevalent in the comparative classroom study (DesPortes, Anupam, et al., 2016) and occurred within Sadler

et al.'s study (Sadler et al., 2017), demonstrating the persistence of this error. Participants also struggled conceptually in their construction of the circuits, or as Sadler et al. identified it, having *theoretical misunderstandings* (Sadler et al., 2017). While Sadler's work only mentions one mistake—placing too many LEDs in parallel—we found participants would leave components only partially connected to the circuit (creating open circuits), place two things in parallel that should be in series, or place two things in series that should be in parallel (creating incorrect circuit formations). Participants also sometimes left out components in their circuits, but this usually seemed to be because of a slip rather than an intentional choice (i.e. not a conceptual issue). Booth et al.'s study demonstrated how detrimental this could be. The lack of a resistor in their participants' LED circuit caused it to source too much current, leading the Arduino to drop its supply voltage to a temperature sensor, which in turn began providing non-sensical data (Booth et al., 2016).

One other error that was not as visible until the non-think-aloud participants were debugging the circuit in the post-study was sequentially reasoning about the circuits. Some participants thought that the LED had to be “closest” to the power source in order for it to work. A few participants in the think-aloud group were noted with this error, but it was unclear how many participants in the non-think-aloud group had this misconception when they were working since they could just hook it up the “*correct*” way. This error is noted within the literature to be one of the most common misconceptions of novices working with electronics, as well as one of the more difficult misconceptions to correct (Beheshti et al., 2014; Chi, 2005; Cohen, 1983; Grotzer & Sudbury, 2000; Küçüközer & Kocakulah, 2007).

6.7.1.2 Summary of Software Errors

The software errors were made up of six main breakdowns: referencing the wrong Arduino pin, setting the pin signal incorrectly (i.e. high/low), failing to connect their code blocks, errors interpreting the sequentiality of code, misunderstanding or misusing the delay block, and misusing or misunderstanding initialization with the pinMode block. Similar to the hardware errors the software errors had errors caused by slips and conceptual difficulties. Referencing the wrong Arduino pin, setting the pin signal incorrectly, and failing to connect blocks properly in the code were all generally caused by slips. The debugging task demonstrated that the participants were generally successful at finding and correcting wrong Arduino pin references, as well as incorrect pin signals. This finding supports Weintrop and Wilensky's finding that the blocks-based language provides code that is easily readable for users. In contrast, identifying blocks that were not connected and should be was much more difficult a problem. When participants went through the guided section teaching them how to connect blocks we identified the difficulty they were having, which Booth et al. also identified as a barrier of the interface (Booth & Stumpf, 2013). They did not however report on the errors caused by participants not being able to see or understand the consequences from missed block connections, which we found to be an insidious problem for participants to find and fix.

The most common software breakdown in our study was a conceptual error in which participants misinterpreted the sequentiality of code. This misunderstanding led participants to create two forever loops, which they intended to run at the same time or two forever loops, which they intended to run one after the other. This error is similar to the parallelism error found in novice programmers who thought code outside of loops would

be “*known*” when the code was inside the loop (Pea, 1984). In our case, the participants thought that code in both loops would be “*known*” at the same time. While two separate forever loops would work in multi-threaded systems (ex. Scratch enables multiple forever loops), this is not the case for this context and did not get the participants the desired effect. Difficulties with this misconception often led to an introduction of other bugs because the code had unintended consequences (ex. only blinking one of their LEDs), which often hindered them from sorting through the issue.

The delay block breakdowns were also generally caused by conceptual issues, but often did not lead to the introduction of bugs. Participants sometimes extended the functionality of the delay block beyond its actual purpose—ex. thinking it could make an LED brighter. However, we found that these mistakes were generally an indicator that a participant was stuck and did not know what else to do at the moment.

Errors with initialization of the pins using the pinMode block spanned from confusion over where they should initialize the pins, if they needed to initialize the pins, and slips in setting it to *input* incorrectly. Reflections from one of the participants indicated that premise of initialization could have been misinterpreted as a setting that needed to be changed between pins as they referenced different pins in their other code blocks. When participants forgot or incorrectly initialized the pinMode, the error resulted in a dim LED which was not usually interpreted as an issue, although participants did notice that it was dimmer than usual. There is a question of whether or not the compiler should assume whether it is an input or output based on whether the user is reading or writing to it in the code (i.e. similar to how Python assigns a type to objects). However, this gets back to the question of whether it should be black-boxed and automatically set or should be glass-

boxed providing information about how it should be set such that participants can learn the difference in how the initialization changes the capabilities of the pins.

6.7.1.3 Summary of Hardware + Software

The hardware + software breakdowns provided us with insight of four mistakes participants made in their understanding of the programming and electronics: trying to program the Arduino without plugging it in, reprogramming the Arduino with the same code that is already uploaded, incorrectly thinking they had solved a problem when they had not, and incorrectly assessing where a bug was in their setup. When participants did not plug in the Arduino to program it, it was because they forgot to rather than because they thought it was not necessary. Catching this error early proved essential for the participants to understand what was and was not working in their setup.

Reprogramming the Arduino was identified as both a conceptual misunderstanding and an indicator that participants were stuck. Our findings revealed that participants did not always understand what it meant for code to be uploaded to the Arduino, with many articulating their surprise when code continued to run after being unplugged and plugged back in. This conceptual misunderstanding could probably be rectified by integrating activities that require participants to question their assumptions—ex. having them power and use the Arduino through a wall or battery pack after code is uploaded.

The last two errors were both indicative of a failure to properly understand how to debug. Thinking the problem was solved, was a mistake caused by not double checking the setup if they were getting reasonable responses from the code and circuit. Incorrectly assessing the bug location caused the bugs in the setup to persist without getting the

participant closer to figuring out the problem. Booth et al. found similar difficulties with the intermediate programmers in her study who had worked with the Arduino before (Booth et al., 2016). We will talk more about the work processes of our participants in the section below.

6.7.2 Working Through the Issues – Novices Strategies for Solving Problems

As participants hit issues they had different ways of handling the errors they were faced with. We will discuss these strategies in the following two sub-sections, examining RQ2.3. First, we will discuss examples of how visibility played a role in both the success and failure of participants sorting through errors and misconceptions from these errors. Next, we will talk more specifically about debugging and the knowledge that would have helped participants problem solve.

6.7.2.1 Impact of Visibility into Errors on Participants Debugging and Learning

Conceptual errors were either ameliorated or exacerbated based on the participants' visibility into their misconceptions. One example of this in the hardware was how participants struggled to understand what should and should not be electrically connected leading to open circuits, short circuits, leaving out a components and incorrect circuit formations. In the first task, participants had to figure out how to create the hardware to blink two LEDs separately. To do this correctly the participants had to create two separate LED circuits and hook them up to two different pins such that they could reference them individually. In the process of solving this task, several participants connected everything in series using one pin in order to get both LEDs blinking. They were soon faced with the constraint of needing to make the LEDs blink separately. When participants went to edit

their code in order to do this, they usually realized that their series circuit would not allow them to reference the LEDs separately. They then had to adapt their circuit schema in order to solve the problem. The physical constraints of the hardware and the visibility into the mistake led to their ability to fix it.

In contrast, when participants made errors but were still able to get the intended blinking effect, they had difficulty understanding how their setup was wrong. In the first task, participants could get away with putting the LED and resistor for each circuit in parallel instead of series or creating a closed circuit with the LED while not incorporating the resistor properly and still get the LEDs to light up using their code. Similarly, in the second task, participants could have several circuit arrangements that enabled them to blink all of their LEDs while still having bugs in solving the task. The limited visual feedback consisted of the physical connections they could trace, and the different brightness levels of the LEDs. However, the brightness of the LEDs was not feedback that was readily seen or interpreted as an issue by participants. They were able to get the basic intended effect (i.e. blinking), and thus did not question the brightness levels. It often was not until they were told they had not completed the activity or asked for a hint for something else that they were able to catch and fix their mistakes. Other research has demonstrated that novices struggle identifying incomplete or short circuits in both circuit diagrams (Engelhardt & Beichner, 2004) and soldered circuits (DesPortes, Pathak, et al., 2016). Without relevant feedback, this error also seems to persist in circuits built with prototyping tools. The participants' processes indicate the need for more relevant information on top of the LED brightness (either from the learning materials or the tools) to be able to understand and correct these mistakes.

Visibility of information was also important for correcting misconceptions surrounding the sequentiality of code. The participants had low visibility into how the computer was compiling and interpreting the code, which created a complex situation for them to fix their errors and misconceptions. Similar to Meerbaum-Salant et al.'s investigation of habits formed in blocks-based environments (2011), when participants were constructing their code for Task #1, we found that participants were often implementing a bottom-up programming process in order to get both of their LEDs to blink sequentially. The participants took the schema (Rist, 1989, 1991) that they had from the Blinky LED task—i.e. the `pinMode` and `forever` loop code to blink one LED—and implemented it twice. The participants in this situation had to work through three issues: the first concerns their thought process about the problem—i.e. incorrectly using parallel processes to get the LEDs to blink one after another; the second concerns interpreting what the IDE is capable of—i.e. the IDE cannot compile parallel processes; and the third concerns interpreting what the code is doing when it does not compile parallel processes—i.e. understanding what code will actually be executed. The second and third are related, but the participants had to deal with the fact that their code wasn't running properly, and then deal with what it was actually doing. Some participants (like the example in section 6.6.6.1) were able to use the feedback from their two working LED circuits to decipher how their code interactions were linked to the output (i.e. understanding that the last `forever` loop touched was active), but this did not necessarily lead to an understanding of how to correct their issues or an understanding of why they were getting certain behavior. Not knowing what they could or should do led participants to repeatedly try failed solutions and focus on innocuous changes such as modifying the delay or uploading the code

multiple times. Visual indicators from the compiler, such as greying out inactive code or highlighting active code (i.e. what Scratch does (M. Resnick et al., 2009)), could help participants think through both how the code is executing and how it should be executing to accomplish the task at hand.

When participants did have visibility into how the computer was interpreting code, it made it easier for participants to sort through their misunderstandings. For example, the IDE provided insights into how code blocks should be used based on their shape and color. These indicators helped participants understand if the blocks would work in the way they intended. For example, when participants thought the *and* block would help them do two things at the same time (i.e. blink this LED *and* that LED), they were able to see that the code block would not fit with any of the code blocks they were currently using. When a code block's color or shape did not rule it out, participants were more likely to use an incorrect block introducing bugs into their setup. Other studies have also identified the usefulness of the colors and shapes in guiding students on their use (Weintrop & Wilensky, 2015a). As we continue to develop both learning materials and tools for physical computing it is important that we determine ways to provide information and activities that can help learners develop their conceptual models of the hardware, software, and programming tasks.

6.7.2.2 Debugging the Hardware and Software

The processes implemented by participants as they tried to identify and fix the bugs in their setup, demonstrated the conceptual hurdle posed by debugging. The participants in the various groups showed that as more problems were introduced in the hardware it also

increased the number of errors that group made with the software. For example, the participants using the static module in the Blinky LED activity had lowest number of issues with their hardware and also had the fewest software errors; similarly, as these same participants had the most difficulty in Task #2, they correspondingly had the most software errors. This demonstrates how errors could compound as participants tried to work through them.

Debugging a physical computing setup was difficult for participants because they had to deal with the possibility of bugs being in both the hardware and software. This complexity makes it important that a systematic debugging process is used to understand where the issues are coming from. However, similar to other findings with novice programmers, this was not something the participants were able to do without guidance (Ahmadzadeh, Elliman, & Higgins, 2005; Edwards, 2004; Murphy et al., n.d.) causing them to incorrectly identify software bugs as hardware and vice versa. The findings suggest that participants lacked methods to check parts of their setup, lacked knowledge of the types of errors they should check for, and did not realize the importance of thoroughly checking their assumptions. Furthermore, the findings provided insight into the common actions participants would implement when they were stuck.

Throughout the activities there were instances when participants could use the working parts of their setup to debug the parts that weren't working, in order to isolate the problem. For example, a common case participants were faced with was when they were able to get one of their LED circuits working but not the second one. The participants could have changed the hardware or software to provide better insight into where the issue was coming from—ex. use the 5V pin to statically check the circuit, switch the non-working

circuit to the pins the working circuit was using, or switch the working code to reference the non-working circuit. These strategies were rarely attempted, and when participants did try them they were not always implemented correctly or in a way that was comprehensive. For example, in the debug example we walked through in section 6.6.7.1, the participant switched the LEDs between circuits and switched the grounds between circuits, but he never checked switching the whole circuit between the two pins he was using.

Part of the difficulty with debugging also stems from knowing the types of bugs that they should check for in their setup. While the participants did not have a list of errors that they knew to check for (unless they received the debugging hints), they did consistently check for a backwards LED. We attributed this to the fact that the guide consistently reminded participants that flipping the LED backwards is a common mistake. Thus, participants would often check for this when they were having difficulties with an LED lighting up. However, their repertoire for checking errors was low, so bugs that could be quickly fixed, such as pushing down the components into the prototyping tool, sometimes lasted a long time before they were diagnosed.

Even when participants were given the debugging hints they did not always go through and physically check all the issues. If they thought something was correct (like their block connections), they would not perform the check of trying to lift up the top block to see if they were connected (this process was outlined in the guide and debugging hints). Even in Booth et al.'s investigation of programmers using the Arduino, they demonstrated that the programmers had a hard time translating their knowledge of debugging software to debugging an Arduino circuit (Booth et al., 2016) emphasizing the importance of this line of research is explored specifically in physical computing.

The participants' strategies for debugging provided us with insight into characteristic actions they would take when they were stuck. The three main indicators we identified were: continuously uploading the same code to the Arduino without making changes, changing the delays multiple times, and taking apart and rebuilding their circuit. Rebuilding the circuit was a strategy also noted in the comparative classroom study (DesPortes, Anupam, et al., 2016). Information on what “getting stuck” looks like can inform designs that are built to offer hints or suggestions when it seems the user is having difficulties.

6.7.3 Design of Physical Computing Tools

In the following section we are going to discuss implications of our findings on tool design for physical computing (RQ3.1, RQ3.2).

6.7.3.1 The Arduino and Electronic Components

In the Arduino and electronic components, we saw how the small components contributed to slips as participants plugged in components to the wrong Arduino pin and plugged the LED in backwards. Further, the small size required more time for participants to decipher information in these components as they tried to avoid introducing bugs—ex. time spent examining the Arduino for the correct pin or the LED's directionality. While it is unlikely that participants learned anything from a slip using the wrong Arduino pin, it is possible that the difficulty the participants had deciphering the directionality of the LED, led to improved learning outcomes based on the increased time they spent using the component. Similar to the concept of seamful design, in which breakdowns or uncertainties in the technology which are usually hidden (i.e. seamless) are can be used in an

opportunistic way to exploit uncertainty (Chalmers, 2003), integrating design choices that make participants spend time on specific concepts might improve learning outcomes. However, Barkhuus integrates seamful design in ways that are intended to increase the enjoyment of the gameplay (Barkhuus, 2005), which stands in contrast to the participants' dislike of the small components. Moving forward in this research direction would require an examination of seamful designs that exploit participants' uncertainty in ways that prove conceptually and motivationally beneficial.

Outside of the slips participants made, we also found that participants were particularly receptive to noticing and identifying visual information that was larger, but with the number of visual cues, the challenge became sorting through the relevant and irrelevant information correctly. As one participant put it, "*there is a lot of stuff going on*", which becomes hard for participants to keep straight. The Arduino has small text on the board and a number of functions and information built into the visual design of the board. Participants articulated many questions associated with this information as they wondered what everything meant. While they did learn to ignore certain information, much of it was initially misinterpreted.

The findings suggest that the design of these boards could benefit from pairing down the amount and types of information presented, while expanding the design to provide more visible and comprehensive identification of the relevant information. While many beginner electronic tools make information more visible (ex. Snap Circuits), they often also come with other restrictions such as tying components to a baseboard. As discussed in the post-study interviews, the ability to integrate the Arduino into the things participants cared about was important to their desire to want to work with it again, making

the restrictions posed by these tools problematic. However, inspiration from the design of these tools, can be integrated into the microcontroller designs. For example, boards such as the Micro:Bit (Figure 88 (left)) that pair down the aspects of the design that are visible and labelled for the user. Other boards such as the Adafruit Trinket (Figure 88 (right)), which integrate a microcontroller with less functionality, could increase the size of the board design and offer more information to the users without overwhelming them.

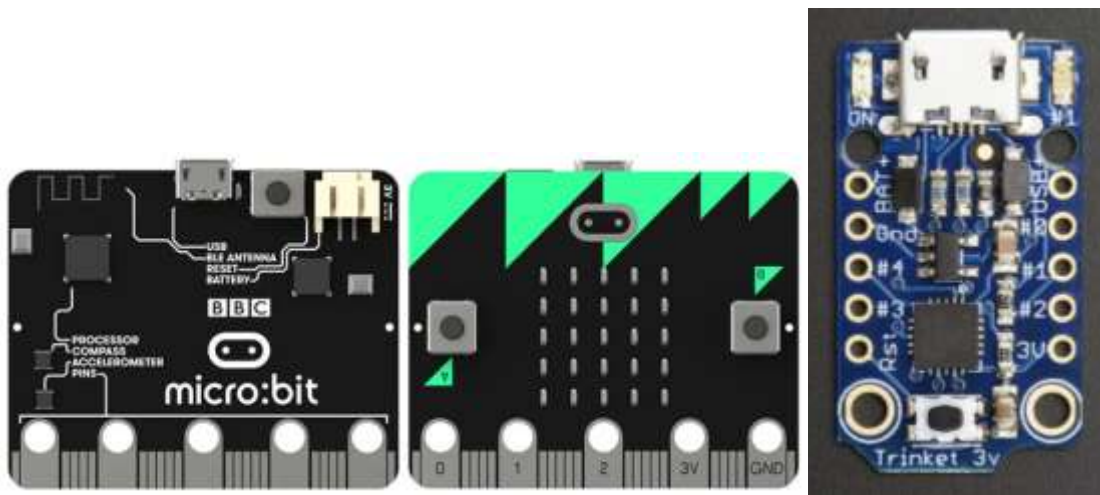


Figure 88 – Micro:Bit example of paired down microcontroller (left) and Adafruit Trinket Mini example of a less complex microcontroller (right)

6.7.3.2 Circuit Prototyping Tools

The findings from the participants usage and success with the different circuit prototyping tools provides an understanding of the benefits and drawbacks within the designs of each of the tools. The static and variable modules were easy for the participants to figure out how to use, indicated by the participants' ability to get up and running with the static modules more quickly and with fewer errors compared to the other groups. When they transitioned to the variable modules and realized they had to plug in the components,

the participants did not have difficulties figuring out where to put the components. Furthermore, the modules demonstrated that having a separate module for each circuit guided participants to hook them up individually to the Arduino rather than in series (i.e. they rarely created an incorrect circuit formation in the first task). Thus, they avoided the conceptual difficulty other participants hit in the first task, but also might not have learned the same things since they did not have to sort through this error. The participants using the modules did face usability and faulty hardware issues leading to a number of open circuit bugs identified in Task #1. The problems did not seem to contribute to an elevated conceptual understanding; however, the participants who did identify and fix this issue often learned to adapt pushing down the components as a debugging strategy. When it was not identified quickly however, it compounded participants' mistakes and misunderstandings of what was going on in their software.

While it is difficult to understand what the participants' experience would have been on the first task had these issues not existed, the transition from the modules to the BitBlox indicates the differences in how the tools forced participants to think about the circuit. In presenting the spectrum of modularity, we hypothesized that learning is related to the concepts that are made visible based on the interactions they require from the participants. In this case, the need to translate the picture of the circuit to create connections between the components was offloaded to the module although still physically visible. When they transitioned to the BitBlox, accomplishing this translation was their greatest difficulty, causing them more open circuit, short circuit, LED backwards, and incorrect circuit formation breakdowns than the other groups. Those that had already used the BitBlox and Breadboard were already forced to make these translations in the use of the

first tool. Because of the difficult transition the first group had to make between Modules to BitBlox, most did not enjoy working with the BitBlox. The findings suggest that students could be more reluctant to transition away from the tools with more integrated scaffolding if the transition to the new tool presents too many difficulties.

The BitBlox also had some advantages and disadvantages to its design. The coloring and separation of connections successfully eliminated the majority of slips caused by participants plugging components into the wrong area by accident. While participants reflected on the benefit of the coloring in their post-study debriefs, the data from the study indicated that the color was also sometimes misinterpreted by participants who created shorts in their circuits and those who were looking to match colors between different blocks to create connections. We identified the same interpretive errors in the comparative classroom study with the BitBlox, in which students created shorts in their circuit because they misunderstood how to use the tool (DesPortes, Anupam, et al., 2016).

One finding that ran contrary to our expectations based on the previous study, was in the difficulties associated with snapping the blocks together. We hypothesized this capability would make it easier for participants to organize their circuits. However, the separate blocks presented participants with more things they had to deal with in an already complex environment. We did note decision paralysis in the comparative classroom study caused by students not knowing which block they should use (DesPortes, Anupam, et al., 2016) but did not recognize the extent of the difficulties the participants were having. In addition, snapping the blocks together did not fit fluidly into how participants were integrating their components into the blocks. When this happened, this led to either a less organized workspace that could lead to open circuits because blocks were left unconnected,

or it could cause barriers in the participants' work processes as they rearranged components to get everything plugged in correctly.

Participants often contrasted using the BitBlox to the simpler and cleaner design of the Breadboard. The Breadboard encompassed all the needed connections on one board and required less interactions to use. One drawback of the Breadboard was that it has a more difficult connection scheme that led to bugs being introduced even when participants were walked through step by step instructions for how to construct the circuit in the Blinky LED activity. This aligns with the difficulties we identified in the comparative classroom study (DesPortes, Anupam, et al., 2016). In the laboratory setting, the participants had a more comprehensive guide to creating the circuit, and were able learn the connection scheme, leading to fewer issues in the later activities. This contrasted to the persistence of the issue in the participants in the comparative classroom study of high school students. This could be indicative of a few things. For one, it could be highlighting a limitation of the laboratory environment for accurately predicting outcomes in a classroom environment. One participant acknowledged the differences, "*In a classroom I would have totally like not, I would have started skipping slides much quicker*". The differences in the persistence of the error could also have been because the participants in the laboratory study were older, more educated, had more materials to reference, and more practice using reference materials.

On top of taking more time for the participants to learn, the second drawback of the Breadboard's connection scheme was that it led to slips from participants unintentionally plugging components into the wrong place. This seemed to be a reflection of the compactness of the breadboard and the lack of indicators showing the separations between

columns and rows. Slips in tool usage were more common in participants using the breadboard than in the other groups.

The transition between prototyping tools presented the participants with a task in translating between connection schemes. The transition between the BitBlox and Breadboard groups was easier than the transition to BitBlox made by the modules group. The BitBlox and Breadboard are isomorphic representations of one another, which is not the case for the modules and BitBlox. The similarities between the tools were recognized by the participants as evidenced in their reflections in the post-study debrief. Enabling ways for participants to become proficient in translating between representational forms is important whether the goal is to provide pathways for an individual to work on hobbyist electronics, or so they can eventually work as an engineer. This skill is needed for working with the majority of prototyping tools as a person experiments with new electronic components, which have circuit diagrams, circuit layouts, and physical circuits that one might need to understand.

6.7.3.3 Blocks-Based Software and Modkit IDE

The software was shown to embody many of the benefits of approachability and ease of use found in prior work investigating blocks-based environments with older students (Weintrop & Wilensky, 2015a). The software enabled the participants to easily start creating code for the first time, and participants reflected on the simplicity of the environment and the ease with which they could pick up the coding. When students did critique the environment they noted possible issues with slow authoring and stated that they would like to advance to more challenging coding environments, which was similar to the

drawbacks High School CS students in Weintrop and Wilensky's study identified (Weintrop & Wilensky, 2015a). However, while Weintrop and Wilensky found that the blocks-based environments could be *potentially damaging* for older students (Weintrop & Wilensky, 2015a), the inauthenticity and kid-like interface was not negatively perceived by our participants, who were also older but had high levels of apprehension when it came to working with CS and electronics. Our study suggests that the type of simplicity you build into the interface should depend on the level of proficiency and self-efficacy of the learner rather than just the age. Furthermore, when combined with the lower perceived cognitive load and higher self-efficacy identified by Booth et al. (Booth & Stumpf, 2013) there is a strong argument to be made for blocks-based languages for novices of all ages.

Outside of the benefits provided by the blocks-based interface, the participants had to deal with errors introduced in their code because of usability mistakes. The breakdowns related to slips in using the software were: accidentally referencing the wrong Arduino pin in the code blocks, setting the pin signal incorrectly, accidentally setting the pinMode to be an input instead of an output, and not connecting blocks properly. Some of the slips were caused because participants brought out a block and simply forgot to change any of the settings (ex. forgot to set the pin reference). Other times, the bugs were introduced because participants tried to select a different setting but missed the selection on the drop-down menu. While these slips did happen, we found that participants were pretty astute at identifying everything except their block connections. It was generally not a conceptual issue, but the visibility into how the computer was interpreting the code would have been useful for identifying this error.

Another slip that was between the hardware and software, but visible in software was when participants forgot to plug in their Arduino when uploading code. When this happened, the compiler identified an *error programming*, but the lack of specificity led some participants to misinterpret it, not realizing why the compiler would have an error programming. This was marked as an understanding barrier by Booth et al. (Booth & Stumpf, 2013). Researchers in CS education have attempted to support students in various ways, from personifying the compiler (M. J. Lee & Ko, n.d.) to aggregating useful suggestions based on other programmers actions (Hartmann, MacDougall, Brandt, & Klemmer, n.d.). Creating better methods to help novices fix compilation errors continues to be actively explored in the CS education literature. Unsurprisingly, our research confirms that this is also an important consideration in physical computing environments. Some researchers such as Jung et al. have begun to investigate useful characteristics of an animated agent for guiding novices through using the Arduino offering insight into how to make the agent more *likeable*, *socially present*, able to reduce *task stressfulness*, and ability to stimulate reflective conversations with the participant (Jung et al., 2014). While the work is in its infancy (i.e. the agent was tested using a Wizard-Of-Oz implementation) the results offer a promising direction for providing information to the user.

6.7.3.4 Other Reflections from Participants

We covered some of the reflections participants had in the prior sections, we focus this section of the discussion on reflections that would further inform the design of tools and materials for novice students (RQ2.4). The intimidation novice participants were faced with when working with the electronics and code for the first time was more extensive than expected. Even with the filtering provided by participants self-selecting into the study,

many had aversion to the material. The negative emotional responses that were elicited by the pre-test caused participants to have flashbacks to other failed experiences, and in short, made many participants feel incompetent. From a design perspective, the need for designs that convey ease of use and approachability are essential for this demographic of older participants who are novices. When it came to the software, we found that the blocks-based interface was intuitive, helped participants pick up programming relatively quickly, and had aesthetics that decreased their aversions and intimidation to programming. In the hardware, several of the participants mentioned that they had seen the Arduino before from friends or on the internet but felt it would be too difficult for them to use. Even when these participants had relatively successful experiences with the material in the study, many were still apprehensive about working with it again, emphasizing the need for beginner friendly materials and tools that guide students through their learning trajectory. More research is needed to understand what types of design choices in a microcontroller and electronic components embody ease of use and approachability as perceived by someone who is intimidated by these tools.

On top of feeling comfortable working with the Arduino again based on their self-efficacy, the participants' desire to work with it in the future was tied the relevance that they found with how it fit into their lives. Even if the participants were able to traverse the material seamlessly with the tools and technology, it was still meaningless unless they were able to connect it to something that they valued. This finding supports the need to bring attention not only to the design of tools for supporting learning but also emphasizes the need for design that considers how these tools can be situated within the lives and values of the learners.

6.8 Contributions

The research study presented in this chapter was intended to provide a close analysis of novice participants working with physical computing using the Arduino for the first time. Within this I sought to contribute to our knowledge of two of the high-level research questions for my thesis:

***RQ2.** What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?
- 2.4 What are novice students' reflections on working with the Arduino?

***RQ3.** How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability in individual and group settings?
- 3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?

The data and analysis of this study led to five main contributions. First the data resulted in development of a codebook of the types of novice breakdowns within the hardware and software (RQ2.1, RQ2.2). Second, the study provided an analysis of the types and prevalence of obstacles, breakdowns and bugs within the hardware and software (RQ2.1, RQ2.2, RQ3). Third, the data offered insight into how the tools and their design characteristics contributed to the various issues and difficulties across tools and transitioning between tools (RQ3). Fourth, the analysis examined novice work processes and strategies for debugging errors with the Arduino (RQ2.2, RQ2.3 RQ3.1). Last, the

analysis of participant reflections gives an account of how these types of experiences and tools are interpreted by novices who are intimidated by CS and electronics (RQ2.4).

6.9 Limitations

The limitations of this study include the context of the lab environment, the study design for statistical analysis, and the faults found in the hardware for the modules. The laboratory context offers one perspective to analyze these research questions; however, it is not the same as a user working on physical computing activities at home or in a classroom setting. The data from this study provided findings that would have been difficult to capture in a real-world context; however, they should still be used to triangulate with data gathered in real-world environments to understand how they are impacted by contextual considerations.

While I originally intended to gather enough data for the statistical analysis across the three groups to complement our understanding of (RQ3.2), the findings were inconclusive from this perspective. First, the faulty hardware presented issues for being able to make assumptions from the findings (although it provided other insights into the debugging practices of the participants). Second, the low number of participants in each group made it difficult to see any statistically significant results between the groups. Furthermore, the study design could be improved to more authentically align with the statistical tests run. For one, the assumptions of a MANOVA require that each participant was not exposed to each tool (i.e. the modules group had an added level not used by the other groups and did not use the breadboard); furthermore, the ANCOVA assume that the participants' groups are completely separate was not true given that the tools used

overlapped across participants. The future work, in the last chapter, outlines an improved study design that will target RQ3.2 using a Multivariate Analysis of Covariance (MANCOVA).

CHAPTER 7. DISCUSSION AND CONTRIBUTIONS

7.1 Introduction

The research presented in this dissertation focused on contributing to our understanding of equity and inclusivity in physical computing learning environments. The problem domain was explored through a lens focused on values in order to conceptualize personally meaningful experiences across a diversity of students. At the beginning of this dissertation, I presented the following thesis statement:

I theorize that exploring the design and analysis of the learning environment through a lens focused on values can provide us with an understanding of how to empower students to create personally meaningful experiences with computing. Further, to enable these types of experiences, physical computing tools should be designed to address usability, learning and self-efficacy.

To explore this thesis statement there were five studies conducted that shed light on three research questions:

RQ1. *How does design of physical computing learning environments affect students' ability to integrate their diverse values into their learning experience?*

- 1.1 How can we conceptualize the design of the learning environment from the perspective of students' diverse values?
- 1.2 How can we analyze the learning environment from the perspective of students' diverse values?
- 1.3 What characteristics of the learning environment in physical computing activities impact students' ability to integrate their diverse values into their learning experience?

RQ2. *What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?
- 2.4 What are novice students' reflections on working with the Arduino?

***RQ3.** How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability in individual and group settings?
- 3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?

In this discussion, I will describe how the studies presented in this dissertation answer these questions as I review the findings from the investigations.

7.2 RQ1: Culture, Values, and Physical Computing Learning Environments

7.2.1 Summary of Contributions: RQ1

The MoveLab Workshop and the Day of the Dead Summer Camp mainly focused on the first research question. The studies contributed to the development and exploration of values-based learning that was used to provide a way to conceptualize the design and analysis of learning experiences to support personally meaningful computing education. While the work surrounding values-based learning is still in its infancy, the studies provide initial exploration of the construct. In this work, we posed infrastructure as a way to deconstruct the design choices within the learning environment from the perspective of values while examining participatory design methods for scaffolding reflection on values (RQ1.1). We integrated the constructs of the self-concept, boundary objects, and value-domains, in order to analyze the learning environment for how values manifested in the environment (RQ1.2). Analysis using these methods in the MoveLab workshop and Day

of the Dead Puppet camp provided insights into how characteristics within the infrastructure both supported and neglected students' values (RQ1.3).

7.2.2 *Conceptualizing the Design of the Learning Environment*

I define and advocate for a values-based learning approach for the design and analysis of learning environments that can promote equity and inclusivity in computing education. Values-based learning is a concept that is still in development but was cultivated from two investigations presented in this dissertation: the MoveLab workshop and the Day of the Dead Puppet camp. I define values using a definition from Graeber, "*conceptions of what is ultimately good, proper, or desirable in human life*" (Graeber, 2001), and situate values-based learning as both a design approach and a lens for analysis. As a design approach, values-based learning seeks to create inclusivity and equity in a learning experience through providing opportunities for students to reflect and integrate their values into the learning environment. As a lens for analysis, it provides insight into the ways in which values manifested in the environment. By using a framing that takes both culture and values into consideration, the goal of values-based learning is to understand where the learning environment was inclusive and where it was neglectful of different students.

In order to deconstruct the design of the learning environment from the perspective of values-based learning, I incorporate the ideas from the literature on infrastructure. *Infrastructure* was initially defined by Star as *something* that facilitates interaction between people and exists in a form that is ready to be appropriated for local practices to serve individual needs (Star & Griesemer, 1989b). Infrastructure was chosen because it draws your attention to where the learning environment integrates constraints as well as flexibility

for students to navigate through the activities. Upon investigating infrastructure, we realized that the design of the MoveLab was integrating *infrastructuring strategies* (P Ehn, 2008) in various ways as we scaffolded participants through the learning experience. The following four strategies outlined by Ehn (2008), enabled us to tease apart the learning environment from the perspective of values-based learning:

1. **Formats**—*pre-defined solutions with the important characteristics highlighted. These characteristics can then be flexibly applied to new situations based on the user's knowledge of the process to appropriately modify characteristics.*
2. **Component Strategy**—*LEGO block approach in which the user can build solutions for specific problems they encounter using the components provided*
3. **Design cases**—*design examples that can serve as cases for the students to reason about and appropriate to new situations (Maher & Gomez de Silva Garza, 1997). They are described in terms of context, problematic situation, and proposed solution (Pelle Ehn, 2008)*
4. **Protocols**—*within a social context are the defined procedural agreements for completing activities and/or communicating*

While these strategies might not be the only useful strategies for structuring values-based learning with computing, our studies demonstrated that they provided a start for providing the basic structure to scaffold students into creating personally meaningful computational artifacts with physical computing.

7.2.3 Analyzing the Learning Environment from the Perspective of Values

In the first analysis of the MoveLab, we found the constructs of the self-concept and boundary objects to be useful in understanding our results. The self-concept is a set of self-beliefs that are an internalization of one's culture, values and how one views herself (Barbara M Byrne & Shavelson, n.d.; Shavelson et al., 1976). In our case, we examined the students' self-concepts with respect to the disciplines of dance and computing. By

examining how participants navigated the learning environment and negotiated views of themselves with computing, we began to understand the importance of their values in creating opportunities to find alignments between the disciplines and how they viewed themselves.

We identified the construct of boundary objects to describe how we observed the participants (both leaders and students) using the dances to negotiate and integrate their diverse values into the learning environment. Boundary objects, as originally defined by Star, are things that emerged in work processes that facilitated collaboration between different groups of people with different needs. The objects are weakly structured between different groups of people that enable them to productively collaborate in the face of differences. Within specific groups, they become more strongly structured based on needs and contexts in which they are implemented. In our work, boundary objects became a way to understand how the participants in the environment were able to create personally meaningful learning experiences even when they had diverse values. Furthermore, we saw how disagreements could be mediated through the abstract themes of the dances.

The work on the MoveLab prompted a more specific analysis from the perspective of values in order to understand how we can create dynamic learning environments that can be personally meaningful. Through iterating on the design of the MoveLab from the perspective of infrastructure to support student values, we conducted the Day of the Dead Puppet Summer Camp. To identify values across the two learning environments we used Graeber's definition—“*conceptions of what is ultimately good, proper, or desirable in human life*” (Graeber, 2001)—and Schwartz et al.'s definitions of value-domains to provide consistent language as we identified and analyzed participants differing values and

analyzed how the learning environment tied into them. We identified the construct of value-domains—power, achievement, hedonism, stimulation, self-direction, universalism, benevolence, tradition, conformity, and security—as useful for talking about values because it has shown that these categories exist within and across most cultures (Bardi & Schwartz, 2003; S. Schwartz, 1999; S. H. Schwartz, 1992; S. H. Schwartz & Bilsky, 1987).

7.2.4 Impact of the Learning Environment from the Perspective of Value-Based Learning

7.2.4.1 Findings from the Self-Concept and Boundary Objects in the MoveLab

Applying the theoretical constructs described in the previous two sections for the design and analysis of learning environments provided us with an understanding of how the characteristics within the learning environments impacted students from the perspective of their values. In the MoveLab, we found that characteristics within the learning environment were influencing the students' transformation of their self-concepts in relation to computing and dance. The participants' previous pre-dispositions combined with the experiences they had in the learning environment, highlighted the different ways the participants were negotiating how they viewed themselves in relation to the disciplines. We found instances in which participants were forming congruence between their self-beliefs and the disciplines, as well as instances in which they were forming dissension between their self-beliefs and the disciplines. Three characteristics within the MoveLab supported students in forming congruence: (1) multiple roles for participation, (2) a socially supportive community, and (3) opportunities for students to integrate their values within the themes. Throughout the workshop, students also developed dissension between their perspective of dance and technology and their self-concept. Three characteristics that

cultivated this dissension were: (1) their previous perspectives of the disciplines (2) fear of failure, and (3) working in a group environment.

Within the MoveLab we also found that the dances emerged as boundary objects enabling us to understand how the participants, with varying backgrounds, interests and expertise, were facilitated in navigating the learning environment in ways that were representative of their values. The dances and computing artifacts were a form of artistic expression that enabled the participants to collaborate using abstractions while embedding different concrete individual values. From learning about one another's backgrounds to learning about one another's expertise, the boundary objects served as educational tools for the entire community of learners. Furthermore, they highlighted various values and interests of the students and educators and helped to identify the learning opportunities between the diversity of participants.

7.2.4.2 Impact of the Infrastructure: Findings from the Analysis of Student Values

In both the MoveLab Workshop and the Day of the Dead Puppet Summer Camp, we found characteristics within the infrastructure that facilitated participants to integrate their values into the learning environment, as well as characteristics that either conflicted with or inhibiting students' values. We break up the findings from the infrastructure into the infrastructure for reflecting and sharing values, for building knowledge, and for organizing their work processes.

Infrastructure for Reflecting on and Sharing Values

The *protocol* of the driving questions provided an accessible topic that participants—including the disciplinary leaders and community members—could contribute to, bringing in expertise from their lives. The MoveLab, which required participants to reflect on an issue, stimulated discussions about participants' values their lives, their relationships, their place in their community/society, and things that they wanted to change in their community/society. In contrast, the Day of the Dead project, required participants to reflect on death, mourning and loss, which stimulated conversations around cultural responses and expressions as participants engaged in discussions surrounding their values. We found that conflicts could arise with participants' values when they felt another culture was being imposed upon them either by the driving question, the topics covered, or the materials.

In both of the learning environments, we scaffolded participatory design activities using *formats* in which leaders served as cognitive models (Collins et al., 1989) demonstrating how one might reflect on their own lives, experiences, and values. The infrastructure proved successful in breaking down the driving question into smaller questions that facilitated participants to engage in self-reflection, while then guiding the participants into linking these reflections to the technology, materials and other mediums for expression (i.e. dance and design of interactive objects).

The MoveLab workshop and Day of the Dead Puppet Camp had different instantiations of *protocols* for engaging with a community outside of the learning environment in order to share the participants' work. We found that the type of

interdisciplinary environment that computing is coupled with can impact the types of *protocols* that participants are comfortable with. In the MoveLab, students rejected the *protocol* for sharing their work in a public performance. A public performance as a non-dancer or non-performer scared many of the students and created an environment where students were concerned about failure rather than feeling pride or self-efficacy over what they created, thus, conflicting with their value of *achievement* and *social power*. In the Day of the Dead Puppet Camp, we found that publicly sharing the interactive objects in an open house did not carry the same concern or apprehension as the dance performance. The design of infrastructure needs to be considerate of how to integrate appropriate flexibility and scaffolding for participants to navigate the *protocols* for sharing their work.

Infrastructure for Building Knowledge

The *components* and *design cases* were used in both environments to build a basis of knowledge for participants to use as they built their final designs. We found that focusing the unit of analysis on the level of the *components* in the environment provides an understanding of the types of concepts, opportunities, and values the learning environment prioritizes. The analysis revealed the issues surrounding learnability and usability of the Arduino and other electronic components. These issues placed constraints on what and how the participants were able to build. However, we did find that the design cases the participants constructed provided a scaffolded way to interact with the components, similar to the “problem- to project-based” learning environments (Brigid J.S. Barron et al., 1998). The cases sometimes proved to be more difficult than intended, but we found benefits in cases that: (1) introduced participants to issues they might hit in their final designs, and (2) could serve as starting points for participants to remix as they created their final designs.

Furthermore, design cases that used computing as an expressive medium, stimulated engagement and motivation.

Outside of constructed design cases, we also had design cases from professional artists and technologists. Our work suggests that professional design cases can serve as a good tool for getting students thinking of forms of expression as well as the affordances and limitations of the technology and medium the professionals are working in. The professional cases might have contributed to the high bar for success that participants in the MoveLab set for themselves. In order to set more realistic expectations, it might prove useful to talk about the constraints and capabilities of the technology components in the environment in relation to what the professional cases are accomplishing.

Infrastructure for Organizing Work Processes

The learning experiences integrated informal and formal *protocols* to engage participants in organizing and sharing their ideas and work with others. In the MoveLab, we found success when the *protocols* forced discussions around work allocation and provided the groups with a shared artifact to reference as they negotiated between roles and design decisions. The organizational *protocols* in the MoveLab for sharing work were rejected in activities when the end result needed to be performed (ex. dance sequences). This demonstrated the need for low barriers for success, while building trust in the learning environment between participants.

In contrast, SCRUMs and peer critiques in the Day of the Dead Puppet camp served as formal *protocols* that facilitated engagement between participants. They were successful because they: (1) enabled opportunities for all participants to engage with each project, (2)

provided a way for the participants to interact in ways they were comfortable with, and (3) stimulated ideas that could be built on by one another. The success of these discussions was also attributed to the fact that they were led by a leader would solicit feedback from the rest of the participants. Furthermore, the accessibility of the problem spaces enabled the discussion to get to a point where it was easy for other participants, including the community leaders, to contribute. These *protocols* offered a low bar for participants to engage in discussions while creating a space for shared reflection on cultural expression.

Summary of Infrastructure

We found characteristics that improved participants ability to create values-based experiences within the learning environment. I will briefly list some of the findings that our work suggests, which are not to be interpreted as concrete or exhaustive, but to begin a dialogue around design of infrastructure as the work on values-based learning continues:

Formats:

- Can enable the educator to serve as a cognitive model
- Can facilitate the breakdown driving questions
- Can guide students to create connections between their ideas, values, and design of computational artifacts
- Should enable flexibility for participants to navigate conflicts with driving question

Protocols:

- Can build community through engaging participants in one another's work
- Can facilitate shared dialogue that values all participants' contributions
- Can create or use shared artifacts to stimulate dialogue around work and values
- Should enable flexibility in their requirements of participation for participants to navigate conflicts

Component Strategies:

- Can bring visibility to the concepts, opportunities, and values the learning environment prioritizes

- Can offer multiple ways for participants to express themselves, their ideas, and their values
- Should promote self-efficacy, learning, and agency
- Should facilitate integration into participants' design
- Should facilitate expression of participants' values

Design Cases:

- Can scaffold learning of components
- Can enable remixing to produce more complex designs
- Can provide multiple examples for reasoning about components and design
- Should offer a range of entry levels to build proficiency and knowledge

7.3 RQ2: Novices Experiences with the Arduino

7.3.1 Summary of Contributions: RQ2

In both the MoveLab workshop and the Day of the Dead Puppet camp, the tools caused the participants to run into barriers in terms of their engagement and self-efficacy with computing. The impact of these tools matter because if we want to be inclusive and equitable in the types of people supported to participate in computing, our learning experiences need build proficiency and knowledge for learners to continue working with the material outside the learning environment. Within the MoveLab workshop and Day of the Dead Puppet camp we identified some of the obstacles that participants faced (RQ2.1) along with capturing the reflections of some of the participants (RQ2.4). We then expanded upon our investigations in the comparative classroom study of the Breadboard and BitBlox, which provided an in-situ study of the obstacles (RQ2.1) and breakdowns of participants (RQ2.2) and how they collaborated (RQ2.3). The Arduino laboratory study was then conducted to more thoroughly explore the participants obstacles, mistakes, thoughts, and processes (RQ2.1, RQ2.2, RQ2.3) as they worked with the Arduino and different

prototyping tools. Post-study debriefs provided more insights into their reflections on the experience (RQ2.4).

7.3.2 Obstacles, Breakdowns and Bugs of Novices

The MoveLab workshop and Day of the Dead Puppet camp provided us with some rough insights into how the participants were struggling with the Arduino. The breadboard became a point of confusion with almost all participants. The experience stimulated the development of BitBlox to spread out the connections, provide visibility to the connections and offer more flexibility in how the tool was integrated into the users' work processes. The comparative classroom study investigated these tools with novice high school students. In the study, we found that participants had difficulties with the tools, small components, the terminology, and the concept of a node within the circuit. While the comparative classroom study offered insights into a real-world context, it was difficult to get a comprehensive view of the prevalence of errors. The laboratory study gave us the ability to track all the errors the participants made leading to a code book representative of these errors. The errors that were most prevalent have been listed in Table 36.

Table 36 – Summary of Breakdowns from Arduino Laboratory Study

<i>Location</i>	<i>Breakdown</i>
HW	Incorrect Circuit Formation
	Leaves out Component
	LED Backwards
	Open Circuit
	Short Circuit
	Wrong Arduino Pin
SW	Code Blocks Not Connected
	Delay Block Misconceptions and Errors
	Pin Signal (HIGH/LOW)
	PinMode Initialization Misconceptions and Errors
	Sequentiality of Code Execution
	Wrong Arduino Pin Selected
HW+SW	Programming w/o Plugging in the Arduino
	Reprograms Arduino With Same Program
	Wrong Assessment of Bug Location
	Incorrectly Thinks Problem is Solved

The hardware errors were made up of six main categories: using the wrong Arduino pin, creating a short circuit, creating an open circuit, assessing the LED directionality incorrectly, leaving out a component in their circuit, and forming their circuit incorrectly so there were components either in parallel or series when they should not be. The software errors were made up of six main breakdowns: referencing the wrong Arduino pin, setting the pin signal incorrectly (i.e. high/low), failing to connect their code blocks, errors interpreting the sequentiality of code execution, misunderstanding or misusing the delay block, and misunderstanding or misusing the pinMode block for pin initialization. The hardware + software breakdowns provided us with insight of four mistakes participants made in their understanding of the programming and electronics. The errors were: trying to program the Arduino without plugging it in, reprogramming the Arduino with the same code that is already uploaded, incorrectly thinking they had solved a problem when they had not, and incorrectly assessing where a bug was in their setup. Other obstacles participants encountered were: participants confused about the purpose of the prototyping

tools, uncertainty in how to create a common ground, understanding all of the visual information displayed on the tools correctly, and interpreting how to use new code blocks.

7.3.3 Work Processes of Novices

In the MoveLab workshop, the Day of the Dead Puppet camp, and the comparative classroom study we were able to identify work processes surrounding the collaboration between participants and the researchers. In these studies, we identified the difficulties posed by the small size of the breadboard. Participants had trouble interacting and even seeing the breadboard throughout the process of working with another participant. The leaders were noted as having difficulties helping students through issues because the small size made it difficult for them to reference and talk about points in the circuit. However, in the classroom study, we found that the symbols such as the “+” and “-” on the rails of the breadboard provided reference markers that could be used to talk about points in the circuit. In the case of the Breadboard, we found groups taking apart their whole circuit instead of debugging what they had built (one group did this in the BitBlox). We found that the BitBlox spread out the circuit making it easier to collaborate because more than one person could interact and engage with the circuit at once. Participants snapped the BitBlox together in various ways, which, depending on their strategies, led to more or less organized circuits. In both the Breadboard and BitBlox groups, we noted participants color coding their circuits with the wires in order to improve organization after they ran into errors. The organizational affordances were a learned tactic that participants found useful in adapting.

In the laboratory study, we identified participants’ strategies when creating the code and circuits to solve problems as well as their strategies for debugging. When participants

were creating the code and circuits to solve the tasks, they seemed to be applying a bottom up approach where they would implement a *schema* (Rist, 1989, 1991) that they learned in the previous activities and then tinker with it until they got the correct solution. This work aligns with the plan composition work in CS education, which has identified that novices often start with a *schema*, or chunk of code for a specific purpose, in order to begin solving a problem (Rist, 1989, 1991). When it came to solving the problems as they worked towards the correct setup for the tasks, they often were able to get through their errors when their misconceptions visibly inhibited them from solving the problem. Issues that did not have visible feedback or it was difficult to see (ex. brightness of the LEDs) participants had more trouble identifying and fixing their mistakes.

When it came to implementing debugging strategies, we found that participants did not have the ability to systematically work through their issues in an efficient manner. The findings suggest that participants lacked methods to check parts of their setup, lacked knowledge of the types of errors they should check for, and did not realize the importance of thoroughly checking their assumptions. Furthermore, the findings provided insight into the common actions—such as changing their delays, reuploading the same code, and rebuilding their circuits—as indicators that participants were stuck. Indicators can be useful for future systems and IDEs that help novices.

7.3.4 *Novice Student Reflections on Their Experience*

In the participants reflections, they discussed their perspective of the overall experience and how their perspectives of computing shifted. The participants across the studies emphasized the importance of creating something in which they found value. In the

MoveLab workshop and the Day of the Dead Puppet camp, the participants reflected positively on their final themes. The participants in the MoveLab found the experience relevant to their lives, and often referred to the theme of their dance as the best part. The participants in the Day of the Dead project loved their final artifacts because they were able to make it their own and were proud of the puppets that they had created. While two participants in the Day of the Dead were not able to create something that they were thematically interested in, they still remarked on their computational object as their favorite thing. We identified other ways their values manifested in the environment and their pride in what they created. In the Arduino lab experiment, participants also emphasized the need for relevance if they were going to work with the Arduino again. While some participants were able to identify ways that the Arduino would be relevant for them, others did not see any value for the Arduino in their lives.

The participants also reflected on the various prototyping tools that they used. The ones that used the modules felt that they were much easier, finding the BitBlox a difficult and confusing transition. The participants who used the BitBlox and Breadboard found that the coloring within the BitBlox was useful as well as the ability to spread out the circuit. They found the number of pieces to be more of a hinderance than a benefit despite the fact that they could snap together. The participants remarked on the cleaner design of the Breadboard, although they mentioned the connection scheme and size of as possible drawbacks. Participants also identified different use cases that might make sense for the Breadboard or the BitBlox. Although they did not always agree on what these use cases should be, they were able to think of situations in which they might find each tool useful.

The participants reflections also revealed their low self-efficacy with the material. In the MoveLab and the Day of the Dead Puppet camp, the participants often reported on the difficulty they had with the tools in ways that indicated their apprehension of working with the Arduino. Participants in the laboratory study were given tools and materials with a focus on understanding how to improve the participants learning and self-efficacy. We found that while participants noted that the fear and uncertainty they came into the study with improved, they were still apprehensive about their next experience being as beginner friendly. The participants identified the comprehensiveness of the guide and the blocks-based programming as things that made them feel more comfortable, decreasing their apprehension.

7.4 RQ3: Designing Physical Computing Tools for Novices

7.4.1 Summary of Contributions: RQ3

The investigation into the design of physical computing tools was stimulated from some of our initial findings in the MoveLab and Day of the Dead Puppet camp, which guided an explicit investigation into the tools. We outlined the spectrum of modularity to bring attention to the design choices within tools for their ability to support learning (RQ3.2). We examined two tools along this spectrum—the Breadboard and BitBlox—in the comparative classroom study. Our findings highlighted how the tools contributed to the mistakes participants made and their experiences learning with one another (RQ3.1, RQ3.2). The laboratory study then examined four tools—static modules, variable modules, BitBlox, and the Breadboard—in a within and between-subjects study. The study helped to understand the usability errors posed by the different tools (RQ3.1). While the data we

gathered provided qualitative insight on how the tools impacted learning and self-efficacy (RQ3.2) our quantitative data was inconclusive.

7.4.2 Usability of Physical Computing Tools

The findings from the participants usage and success with the different circuit prototyping tools provides an understanding of the benefits and drawbacks within the designs of each of the tools. The four prototyping tools were the static and variable modules, the BitBlox and the Breadboard. The static and variable modules were easy for the participants to figure out how to use, indicated by the participants' ability to get up and running with the static modules more quickly and with fewer errors compared to the other tools. When they transitioned to the variable modules and realized they had to plug in the components, the participants did not have difficulties figuring out where to put the components. The participants using the modules did face usability and faulty hardware issues leading to a number of open circuit bugs.

The BitBlox also had some advantages and disadvantages to its design. The coloring and separation of connections successfully eliminated the majority of slips caused by participants plugging components into the wrong area by accident. While participants reflected on the benefit of the coloring in their post-study debriefs, the data from the study indicated that the color was also sometimes misinterpreted by participants who created shorts in their circuits and those who were looking to match colors between different blocks to create connections. We identified the same interpretive errors in the comparative classroom study with the BitBlox. In the comparative classroom study, we also noted benefits for organization that the modularity of the BitBlox enabled as the participants

created circuits snapping blocks together in different ways. However, in the lab study we found drawbacks to the modularity of the tool. The BitBlox increased the number of components that the participants had to think about and use, and the video data revealed how snapping together the blocks was not a seamless operation as they built their circuit. Because of the usability issues, we found that participants would not snap together the blocks, causing an unorganized circuit that could result in open circuits when the blocks pulled apart.

The MoveLab workshop, Day of the Dead Puppet camp, comparative classroom study, and laboratory study all identified issues with the size of the breadboard causing errors because of slips in using the tool. Furthermore, participants had difficulties initially learning and applying the connection scheme without visible indicators that they could reference. While the participants' errors persisted in the classroom study, the participants in the laboratory study were able to work through their issues in the initial activities and become proficient at using the connection scheme. The college students probably had different proficiencies than the high school students, but this could have also been due to contextual differences of being in a laboratory setting versus a classroom setting.

In the Arduino and electronic components, we saw how the small components contributed to slips plugging in components to the wrong Arduino pin, plugging the LED in backwards, and, in the classroom study, plugging in the button backwards. Further, the small size required more time for participants to decipher information in these components as they tried to avoid introducing bugs. While the more time spent using the tools might have led to more learning gains, it is unclear and a possible direction for future work.

While the small components posed issues, we found that participants were particularly receptive to noticing and identifying visual information that was larger. However, with the number of visual cues, the participants had difficulty sorting through and interpreting the relevant and irrelevant information. The findings suggest that the design of the microcontroller boards could benefit from pairing down the amount and types of information presented, while expanding the design to provide more visible and comprehensive identification of the relevant information. While these slips did happen, we found that participants were pretty astute at identifying everything except their block connections.

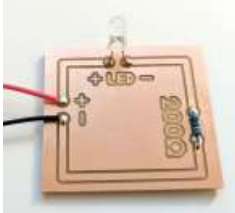

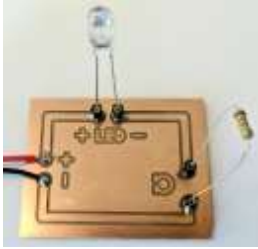


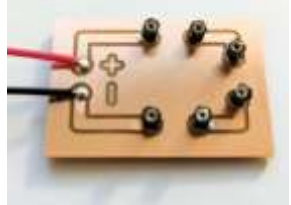
The software was shown to embody many of the benefits of approachability and ease of use found in prior work investigating blocks-based environments with older students (Weintrop & Wilensky, 2015a). The software enabled the participants to easily start creating code for the first time, and participants reflected on the simplicity of the environment and the ease with which they could pick up the coding. However, there were some errors caused by usability issues with the programming environment. The breakdowns related to slips in using the software were: accidentally referencing the wrong Arduino pin in the code blocks, setting the pin signal incorrectly, accidentally setting the pinMode to be an input instead of an output, and not connecting blocks properly. Some of the slips were caused because participants brought out a block and simply forgot to change any of the settings (ex. forgot to set the pin reference). Other times, the bugs were introduced because participants tried to select a different setting but missed the selection on the drop-down menu. Another slip visible in the software was when participants forgot to plug in their Arduino to upload code. When this happened, the compiler identified an

error programming, but the lack of specificity led some participants to misinterpret it. Creating better methods to help novices identify and fix errors in their setup, is an essential area of research for supporting learners to become proficient with physical computing.

7.4.3 *Abstraction Layers Impact on Learning and Self-Efficacy*

The various abstraction layers in the physical computing tools were broken down in the spectrum of modularity using transparency and interactions as a way to conceptualize how the tools could support learning. Building on the ideas from black-box and glass-box scaffolding (Hmelo & Guzdial, 1996, p.), the spectrum explores how we can create glass-boxes to develop proficiencies with different concepts and electronic components. The abstraction layers are characterized by the *transparency*—what is visible or obscured from the learner—and the *affordances for interaction*—how learners manipulate and combine the tools as they build computational objects. These two characteristics are inseparable—the transparency affects what the learner interacts with, and what the learner interacts with in turn affects what is transparent to the learner. The underlying premise is that you can explore the components for what interactions are enabled to understand what concepts students are exposed to. The spectrum breaks down the components into six levels as shown in Table 37.

Table 37 – Spectrum of Modularity

	LEVEL	EXAMPLE
Multiple Component Circuit Boards	Level 1: Plug-and-Play Circuits with Static Electronic Components	
	Level 2: Plug-and-Play Circuits with Static and Variable Electronic Components	
	Level 3: Plug-and-Play Circuits with Variable Electronic Components	
Single Component Circuit Boards	Level 4: Static Single Electronic Component Modules	
	Level 5: Variable Single Electronic Component Modules	
Multiple Component Circuit Board	Level 6: Open Circuit Modules	

On top of the modularity of the peripheral electronics, I explored the design space of the open-ended prototyping tools using the Breadboard as a starting point. The BitBlox were created as a probe to understand how visibility into the connection scheme and flexibility in its design (i.e. snapping BitBlox together) would impact the participants learning experience. While the MoveLab and Day of the Dead Puppet Camp, both only used the breadboard, the comparative classroom study explored the Breadboard and the BitBlox, and the Arduino laboratory explored the variable and static Plug-and-Play modules in conjunction with the Breadboard and BitBlox. Through investigation of these different abstraction layers in the tools we have begun to bring insights into how they impact learning and self-efficacy (some of which was discussed in the sections above).

Participants in the MoveLab workshop and Day of the Dead Puppet camp expressed concern over their abilities to be able to work with the electronics without help of the instructor. Furthermore, while the participants had access to all of the concepts it was unclear what they learned because the burden of debugging and creating working computing artifacts was often pushed onto the leaders (more so in the MoveLab because of the shorter time frame). Participants reported on their low self-efficacy, as they reflected on needing the help of someone who is skilled in electronics in order to be able to work with the electronics. The findings demonstrated that the breadboard and Arduino were inappropriate for those learning contexts. We further explored the tool design in the comparative classroom study and the laboratory study.

In the comparative classroom study, we noted different designs that were useful in helping participants collaborate with one another in ways that facilitated learning. The BitBlox spread out the circuit so that participants could both be engaged in the activity at once and thus consistently participant in the learning activities. The Breadboard limited the ability to have a continuous engagement because of the size. The Breadboard, however, had markings that helped the participants and instructors be able to talk more seamlessly about the circuit. We found this helped as the participants struggled to pick-up all the new terminology.

In the Arduino laboratory study, we intended to get conclusive quantitative data on how the abstraction layers were impacting the participants self-efficacy and learning outcomes from the pre-/post-tests. However, these results were inconclusive, which could have been due to the faulty hardware in the modules and the low number of participants in each group. An improved study design is outlined in section 8.1, which will be conducted in future work. Outside of the pre-/post-tests, the qualitative data from how the participants transitioned between tools provided some insight into what they learned. In presenting the spectrum of modularity, we hypothesized that learning is related to the concepts that are made visible based on the interactions they require from the participants. The difficulties the participants transitioning between the variable modules to the BitBlox provided support for this hypothesis. In the case of the variable modules, the need to translate the picture of the circuit to create connections between the components was offloaded to the module although still physically visible. When they transitioned to the BitBlox, accomplishing this translation was their greatest difficulty, causing them more open circuit, short circuit, LED backwards, and incorrect circuit formation breakdowns than the other groups. Those that

had already used the BitBlox and Breadboard were already forced to make these translations in the use of the first too and therefore, made fewer mistakes.

7.5 Conclusion

RQ1. How does design of physical computing learning environments affect students' ability to integrate their diverse values into their learning experience?

- 1.1 How can we conceptualize the design of the learning environment from the perspective of students' diverse values?
- 1.2 How can we analyze the learning environment from the perspective of students' diverse values?
- 1.3 What characteristics of the learning environment in physical computing activities impact students' ability to integrate their diverse values into their learning experience?

The initial studies of the MoveLab and Day of the Dead Puppet camp provided the beginning investigations into how we can integrate computing as an expressive medium in learning experiences in ways that support students to dynamically integrate their values and culture into the learning environments. We defined the concept of values-based learning in order to provide an explicit focus on how we integrate values into both the design and analysis phase of exploring learning environments. We integrated infrastructure and participatory design methods as a way to design from a values-based learning perspective (RQ1.1). In our analysis, we demonstrated the usefulness of value-domains, the self-concept, and boundary objects in understanding values-based learning environments (RQ1.2). Through this process, we identified characteristics of the infrastructure that impacted and engaged with participants' values throughout the learning experience (RQ1.2).

RQ2. *What are novice students' experiences when working with the Arduino for the first time?*

- 2.1 What are the obstacles, questions and confusions that novice students have when working with the Arduino?
- 2.2 What are the common breakdowns, misconceptions and errors of novice students when working with the Arduino?
- 2.3 What are the work processes of novices working with the Arduino?
- 2.4 What are novice students' reflections on working with the Arduino?

The MoveLab and Day of the Dead Puppet camp provided us with the initial findings on the obstacles that participants faced when working with the Arduino. Analysis of the post-study interviews also captured their reflections of the experience (RQ2.1, RQ2.4). The comparative classroom study of the Breadboard and BitBlox provided us with an in-situ study that more explicitly focused on the tools. We identified the obstacles and breakdowns of the participants, as well as captured the participants collaborative practices (RQ2.1, RQ2.2, RQ2.3). The Arduino laboratory study was then conducted to gather data in a constrained environment focused on the tools to gather comprehensive data on an individual's experience. The findings contributed the identification of breakdowns in the hardware and software leading to a codebook representative of these issues (RQ2.1, RQ2.2). Applying this codebook, provided an understanding of the prevalence of the different types of obstacles, breakdowns, and bugs that participants were faced with (RQ2.1, RQ2.2). Examining how participants approached solving tasks and debugging their setups gave us insight into their work processes (RQ2.3). Post-study debriefs then provided more insights into their reflections on the experience (RQ2.4).

RQ3. *How should we design physical computing tools for novice learners to increase their knowledge and self-efficacy?*

- 3.1 What characteristics of the hardware and software tools impact their usability in individual and group settings?

3.2 How do different abstraction layers in the design of physical computing tools impact students' learning and self-efficacy?

The MoveLab and Day of the Dead Puppet camp brought the usability of the hardware tools to our attention (RQ3.1) as well as the impact the tools can have on users' self-efficacy (RQ3.2). To examine the hardware tools from the perspective of learning, the spectrum of modularity teased apart the various abstraction layers. The analysis provides a way to conceptualize the design for learning opportunities using the transparency and affordances for interaction (RQ3.2). The tools were then further examined in the comparative classroom study, which provided insights into usability of the Breadboard and BitBlox in collaborative work, and in the Arduino laboratory study which provided insights into the usability of the Breadboard, BitBlox and Modules in individual work (RQ3.1). The comparative classroom study identified the ways in which some of the usability issues could exclude participants from the learning experience or create opportunities for engagement while facilitating discussion with educators (RQ3.2). The laboratory study provided another perspective exploring the tool design in a constrained environment. In this investigation participants worked with a combination of tools: static modules, variable modules, BitBlox, and the Breadboard. The study helped to understand the usability errors posed by the different tools (RQ3.1). The study was designed as a within and between-subjects study to examine the tool effect on self-efficacy and learning, but the quantitative data gathered provided inconclusive findings. However, the qualitative data provided other insights on how the tools impacted the overall learning experience (RQ3.2).

CHAPTER 8. FUTURE WORK

8.1 Overview

The future work from this dissertation encompasses two main directions that need to be merged together. The first is a continued exploration of the infrastructure for student authorship of values-based learning experiences, and the second is specifically in relation to the tool design for supporting values-based learning in physical computing education. I will briefly describe these avenues forward in the next few sections.

8.2 Further Explorations of Infrastructure for Values-Based Learning

The work still needs to provide a more cohesive understanding of the benefits and drawbacks of a values-based learning approach to the design and analysis of learning environments. Thus far the construct has been examined in the context of heavily resourced project-based work with computing as an expressive medium. However, we did not answer questions as to whether or when other approaches might be more useful than an approach centered on student values. For example, the approach of culturally relevant curricula might be more appropriate as a way to frame the design of learning environments when it is more critical that teacher constrain the learning activities. It is currently unclear how these methods have a place or a way to integrate into more structured curricula.

The studies conducted within this research demonstrated success in stimulating dialogue and artifacts that were inclusive and representative of participants' values. Furthermore, the participants acknowledged the importance of this characteristic of the learning environment. However, there are many open questions for values-based learning

that need to be explored in future research. One important avenue for exploration, which drives how this construct can be used and applied, is in how we frame the relationship between computing and student values. In this work we limited the scope to how student values manifested in the dialogue and artifacts within the learning environment. However, it neglects to consider how the computing knowledge and experience translates into the participants lives outside the learning environment. Continued investigations into the ways we define and analyze how computing experiences integrate values can lead to a more comprehensive understanding of equity and inclusivity in computing education.

Future research should continue investigations of values-based learning across contexts. The two learning environments presented in this study were heavily resourced, which is a good starting point to see if something can work but limits our understanding of how these experiences would fare in more common learning environments. Research needs to address what characteristics of the infrastructure are important based on the constraints of different learning environments. One aspect of this concerns how educators are supported in integrating the values-based learning into their environments. While the researchers in these studies collaborated with other organizations and educators, they were still heavily involved in designing and running these experiences. We still do not know what types of infrastructure are the most important for supporting educators in designing and carrying out these types of experiences, or what types of pedagogical and content knowledge is most important for them to embark on these types of experiences. Future research exploring these questions with educators who have different experiences and areas of expertise can begin to understand these questions.

Another aspect of the context of these experiences is the question of whether or not informal learning experiences of this nature have a place in formal K-12 education. While we do have data of these types of experiences succeeding in college level courses, we do not know if these experiences could or should be translated into formal K-12 school environments. My inclination is that they should be, but there are many stakeholders in these environments that would determine if, when and how K-12 education would benefit from these types of experiences.

8.3 Further Exploration of Tools and Learning for Values-Based Learning Experiences

8.3.1 Re-Design of Physical Computing Tool Lab Study

While the lab studies were supposed to provide a quantitative analysis, the data mostly served for the opportunities it provided for a deep qualitative analysis on novice experiences with the Arduino tools. Outside of needing more participants to improve the power, the faultiness of the modules inhibited me from answering some of my questions surrounding the usefulness of the modularity of the tools. I hypothesized that the modules would have decreased the amount of time and errors the participants ran into, which if true would make a case for these types of tools. However, I was not able to understand this question. While different tool designs should be explored within the next study, based on the findings from this study, I will go over the improvements to the study design for future explorations.

The lab study was previously created as a within and between-subjects design; however, we did not provide all participants with the same tools, which makes the use of

the MANOVA questionable. I would change this such that we have a between-subjects design in which the participants only use one tool. This will provide an opportunity to use the MANCOVA analysis while cutting down on the time it takes for each participant to go through the study. The independent variable will be the tool the participants are using; the dependent variables will be the post-tests and number of bugs introduced; and the pre-tests will serve as the covariates in the analysis. The power analysis would be completed based on the number of tools cross-analyzed (i.e. levels) in order to determine the number of participants that should be placed in each group. By designing the study in this way, it will help to answer questions concerning the benefits of a more modularized hardware tool while making the data collection more attainable.

8.3.2 *Moving Forward with the Tool Investigations*

The analysis of the tools was the first step in understanding how we might improve the opportunities for participants to learn about the various disciplines that physical computing encapsulates. The findings provide insights into the tool design as well as the work processes of novices that highlight directions for future work. The tool design implicates values of the participants in a few ways, one is in how and what they learn and the implications it has for them to apply the domains in ways they value, and another is in what they build and how it integrates their values. Within the open-ended prototyping tools, I began exploration of how to teach using these tools so that we can eventually understand the implications for how participants are enabled to use the knowledge they gain. The prototyping tools demonstrated clear improvements in how they could be designed. While the BitBlox connection scheme was easier for participants to pick up in the classroom environment, the cognitive load and the obstacles to their work processes provides insights

for improvements. The Breadboard had a cleaner and static design, which helped decrease the amount the participant had to deal with; however, the lack of visibility into the connections contributed to a higher barrier to entry and unintentional slips because of the close connections within the tool. Merging the design choices within these two tools such that there are fewer pieces for the participants to deal with, have a visible connection scheme, and provide a layout that is slightly more spread out than the breadboard could provide improvements over the current two tools.

In terms of the affordances that the circuit modules provide for mitigating certain learning curves, we still do not know how their design impacts learning and the overall experience of students in physical computing learning environments. In one of the previous sections, I outlined another experiment that could be conducted in order to continue to explore these tools in the laboratory environment in order to answer these questions that my research did not get at. On top of continued laboratory experiments, the design affordances of the modules should be explored in real-world settings. The comparative classroom study with the Breadboard and BitBlox demonstrated how the differences in the laboratory and classroom setting led to different insights for the tools and the impact they have on the socio-technical environment.

The studies presented in this work did reveal various ways in which novices approach problems and the difficulties that arise. The application of these findings to the design of tools that can support these processes is important for building proficiency and enabling participants to design in these environments. For example, the task of debugging errors showed to be difficult for the novice participants. Not only did our participants struggle, but in Booth et al.'s experiment they demonstrated that this difficulty spread even

to those who have had multiple years of programming experience and thus were familiar with debugging programs (Booth et al., 2016). Research should contribute to how we might teach debugging strategies and how we might design tools that can better support participants to debug their creations. While there are tools for debugging circuits, such as voltmeters they also pose a usability barrier for novices who might not know the symbols and language used in the tool or how to implement it correctly in the circuit in order to get a reading of current or voltage.

As we explore designs for various situations the tools need to be explored for the affordances for the different computing objects that can be achieved and the implications these objects have on student values. We explored the tools within the domains of dance and interactive computational artifacts. While we did not focus our data collection or analysis on how these tools fit within these domains, it was clear that they could be improved for both. Understanding more concretely how the materiality and modularity impact the types of creations within different domains has implications for the types of values that can emerge in these environments.

The designs of the various physical computing tools should continue to be explored in acknowledgement of goals for learning, which should tie into values-based learning. When analyzed from the perspective of values, certain situations might call for black-boxing aspects of the hardware and software rather than incorporating the learning curve associated with them. Part of what needs to be understood is what types of proficiencies we should be designing for if we are to empower students within the computing domains. We still do not have a comprehensive understanding of what concepts are most important for participants who might use this for their own personal applications, or what concepts

can be ignored in different cases. These questions are important for how the learning environments and knowledge becomes situated in the lives of students. By exploring these avenues of research, we can better understand how to empower students with the power of computing.

APPENDIX A. LABORATORY STUDY MATERIALS

A.1 Pre-Screening Survey

Basic Information

1. Name:
2. Email:
3. Age:
4. Name of College You are Currently Enrolled In:
5. Semesters Completed in College:
6. Degree program you are currently enrolled in:

Mark only one oval.

- Bachelor's degree
- Master's degree
- PhD degree
- Associate's degree
- Other: _____

7. What is your major in your degree program?

Expertise Rating in Computing

8. Rate your expertise in Electronics

Mark only one oval.

Novice 1 2 3 4 5 6 7 8 9 Expert

9. Rate your expertise in Computer Programming

Mark only one oval.

Novice 1 2 3 4 5 6 7 8 9 Expert

10. Rate your expertise in Physical Computing

Mark only one oval.

Novice 1 2 3 4 5 6 7 8 9 Expert

Prior Class Experience

11. Did you take a physics AP exam in high school?

Mark only one oval.

- Yes
- No

12. How many high school classes have you taken that involve circuits?

Mark only one oval.

- 0 Classes
- 1 or More Classes

13. How many college classes have you taken that involve circuits?

Mark only one oval.

- 0 Classes
- 1 or More Classes

14. How many high school classes have you taken that involve computer programming?

Mark only one oval.

- 0 Classes
- 1 or More Classes

15. How many college classes have you taken that involve computer programming?

Mark only one oval.

- 0 Classes
- 1 or More Classes

16. Have you ever used a microcontroller before, such as the Arduino, Teensy, Micro:Bit, Raspberry Pi, etc.?

Mark only one oval.

- Yes
- No
- I can't remember

17. Describe any informal educational experiences (ex. club or a summer camp) that you have had with electronics or computer programming.

A.2 Pre-/Post-Test of Self-Efficacy and Knowledge

Section #1

Rate your confidence in doing the following Arduino physical computing tasks using a scale of 1 (not at all confident) to 7 (absolutely confident). If a specific term or task is totally unfamiliar to you, please mark 1.

1. I could complete a programming task if I had a lot of time to complete the task and I was told what the program had to do.
2. I could complete a programming task if I was told what the program had to do and I could call someone for help if I got stuck
3. I could complete a programming task if someone showed me how to solve the problem first
4. I could find ways of overcoming a problem if I got stuck at a point while working on a programming task
5. I could create a program to blink an LED
6. I could create a program to read or write a signal (ex. buzzer or temperature sensor)
7. I feel comfortable programming on my own
8. I could build a circuit if I was given the components, told what the circuit had to do, and had a lot of time to complete the task.
9. I could build a circuit if I was given the components, told what the circuit had to do, and could call someone for help if I got stuck
10. I could build a circuit if I was given the components and someone showed me how to do it first
11. I could find ways of overcoming a problem if I got stuck at a point while building a circuit
12. I could create the circuit to blink an LED if given the components

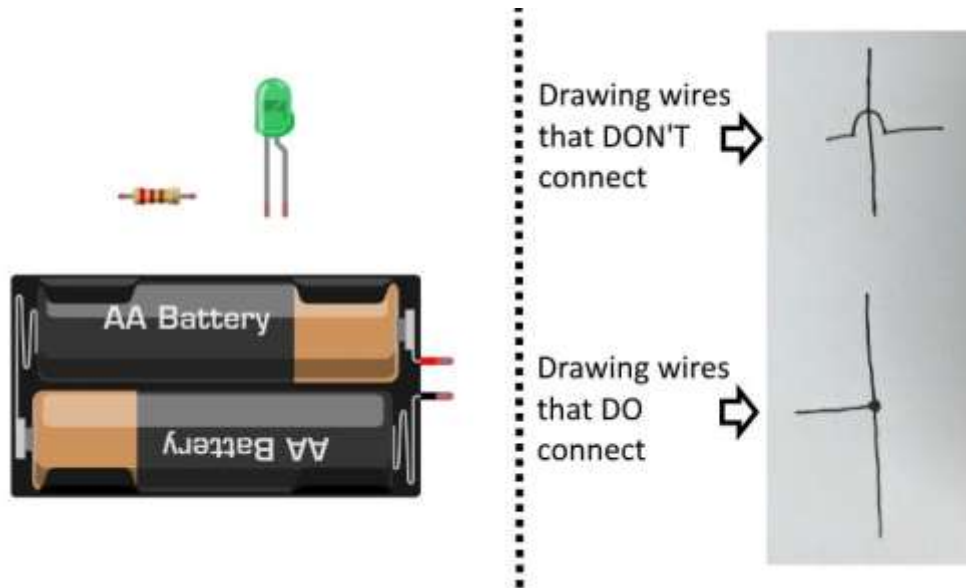
13. I could create a new circuit if I was given the components and a picture of the connections
14. I feel comfortable building circuits on my own
15. I could complete a task using an Arduino and electronic components if I was given the specifications and the components, and I had a lot of time to complete the task
16. I could complete a task using an Arduino and electronic components if I was given the specifications and the components, and I could call someone for help if I got stuck
17. I could complete a task using an Arduino and electronic components if I was given the specifications and the components, and someone showed me how to solve the problem first
18. I could find ways of overcoming a problem if I got stuck at a point while working on a task with the Arduino
19. I could figure out how to use the Arduino to have a button control multiple LEDs if I was given the components and a diagram of the circuit connections
20. I could figure out how to use the Arduino to read a temperature sensor to control a motor if I was given the components and a diagram of the circuit connections
21. I feel comfortable using the Arduino on my own

Section #2

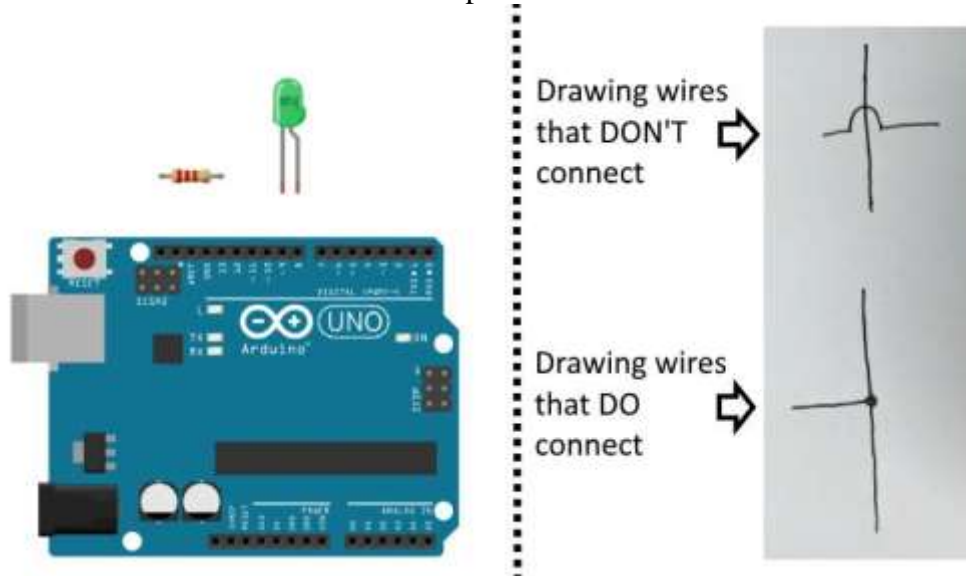
Fill out each question to the best of your ability. If the question involves a circuit and code pay attention to both the circuit and the code. Some of the questions have small text on the pictures, which we could not seem to enlarge efficiently. If you are having difficulty reading anything, ask the researcher for help deciphering the tiny text. If you do not know the answer to any of the questions mark "I don't know" as your answer rather than taking a blind guess.

For the purposes of this experiment, we will not allow you to come back to questions or skip between the questions on the test. Please **DO NOT** hit the back button to go look at other questions after you have answered them. Doing so would invalidate your answers as we need to keep this consistent for all participants in the study.

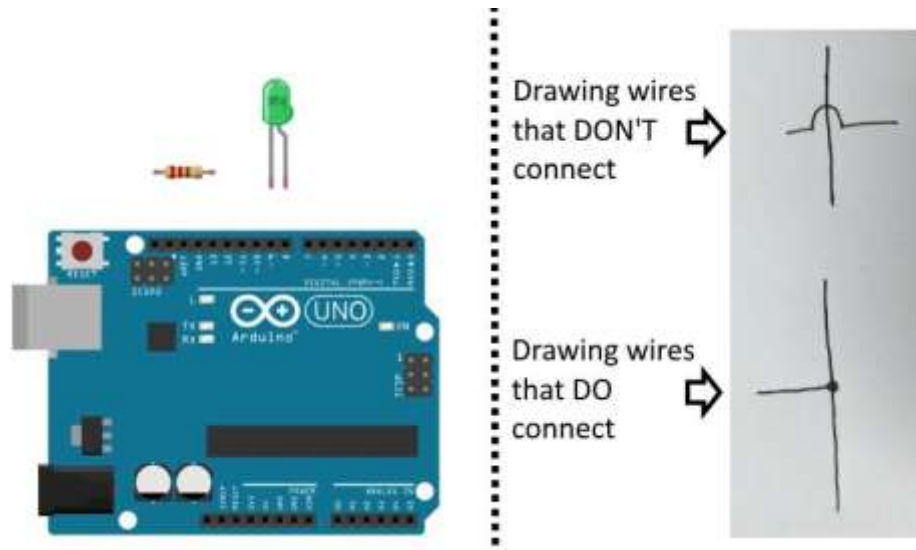
1. Ask the researcher for the materials to complete this question. You will be answering on a sheet of paper rather than on the computer. Using the pictures of electronic components (battery, resistor and LED), build a working (closed) electrical circuit by taping down the pieces and using your pencil to draw any of the connections for a working circuit. If you have to draw two wires that cross paths, follow the directions on how to draw wire connections in the picture below.



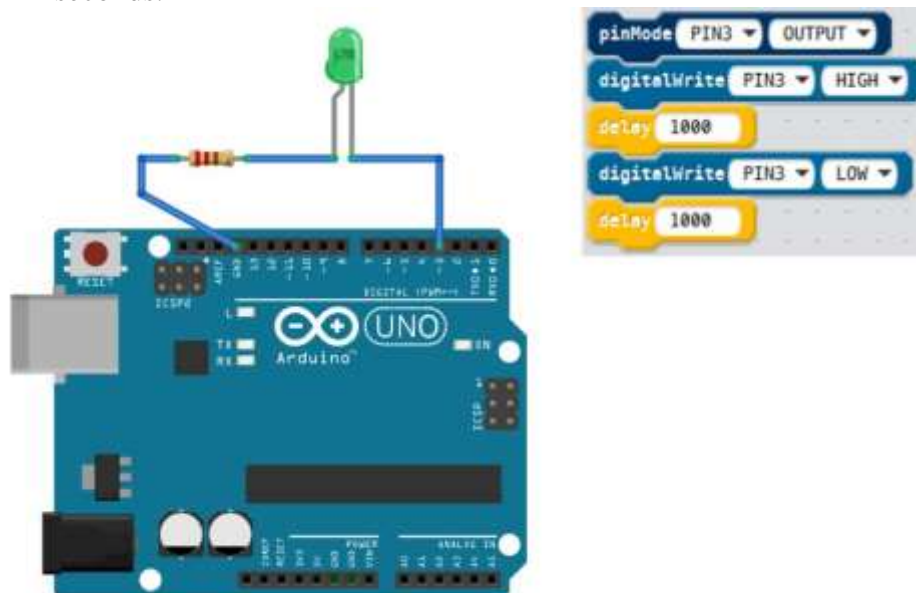
2. You will be answering on a sheet of paper rather than on the computer. Using the pictures of electronic components (Arduino, resistor and LED), build a working (closed) electrical circuit by taping down the pieces and using your pencil to draw any of the connections for a working circuit. Use the 5v pin on the Arduino for power. If you have to draw two wires that cross paths, follow the directions on how to draw wire connections in the picture below.



3. You will be answering on a sheet of paper rather than on the computer. Using the pictures of electronic components (Arduino, resistor and LED), build a working (closed) electrical circuit by taping down the pieces and using your pencil to draw any of the connections for a working circuit. The microcontroller on the Arduino is programmed to blink an LED on digital pin 10. If you have to draw two wires that cross paths, follow the directions on how to draw wire connections in the picture below.



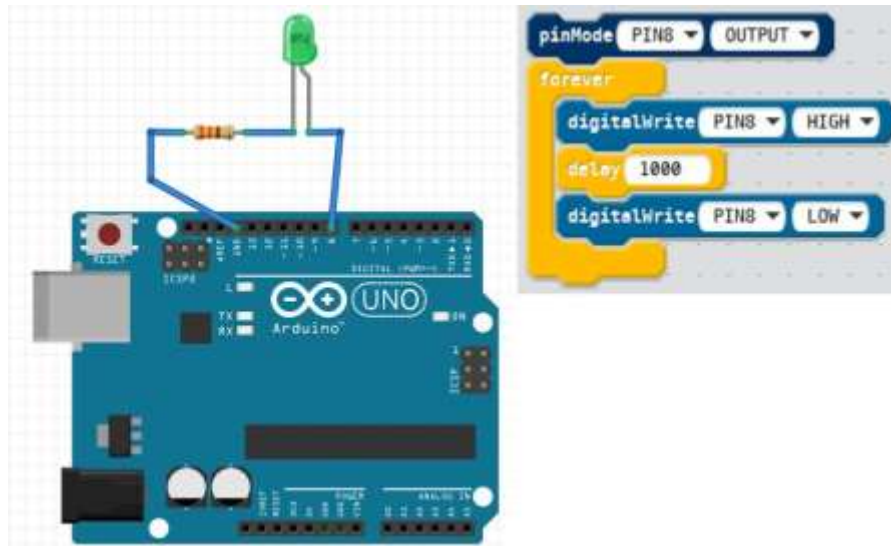
4. If the following code was uploaded to the following circuit, what would you see happen? (analyze both the circuit and the code for correctness). The delay is in milliseconds.



Mark only one oval.

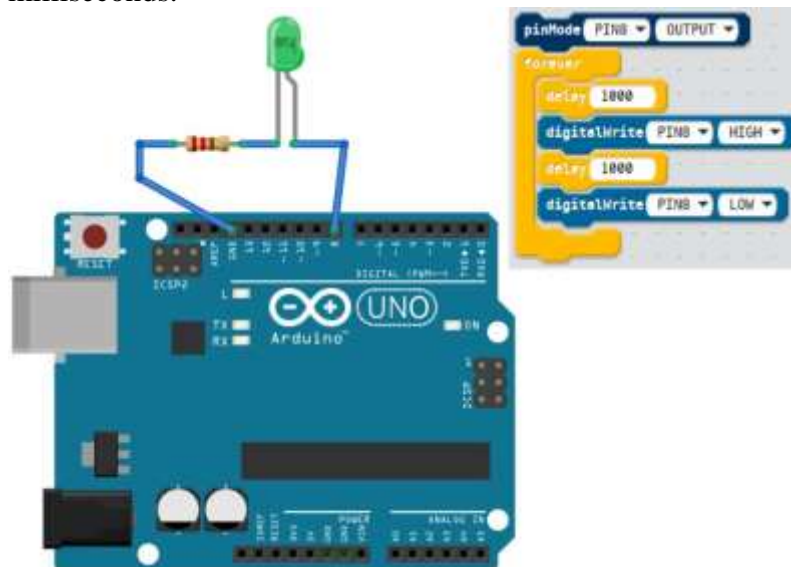
- The LED would look like it would blink every second and repeat forever
 - The LED would look like it would turn on for 1 second and turn off forever
 - The LED would look like it is always on
 - The LED would look like it is always off
 - I don't know
5. If the following code was uploaded to the following circuit, what would you see happen? (analyze both the circuit and the code for correctness). The delay is in

milliseconds.



Mark only one oval.

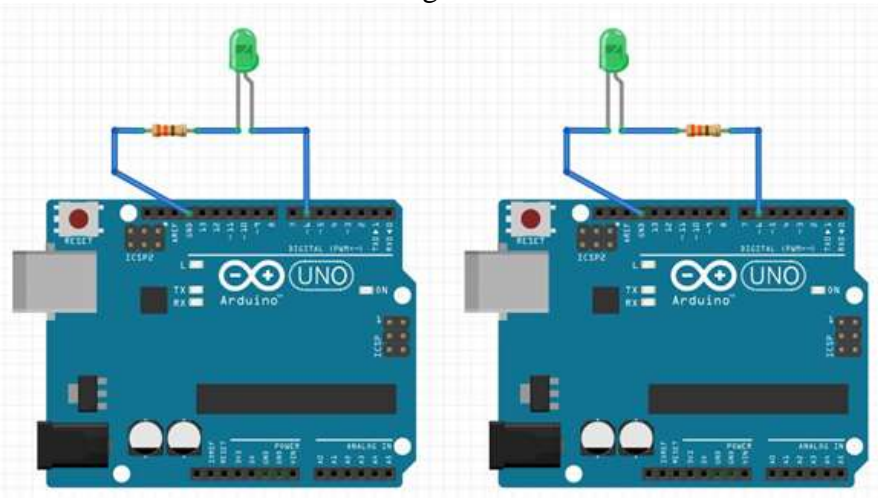
- The LED would look like it would blink every second and repeat forever
 - The LED would look like it would turn on for 1 second and turn off forever
 - The LED would look like it is always on
 - The LED would look like it is always off
 - I don't know
6. If the following code was uploaded to the following circuit, what would you see happen? (analyze both the circuit and the code for correctness). The delay is in milliseconds.



Mark only one oval.

- The LED would look like it would blink every second and repeat forever
- The LED would look like it would turn on for 1 second and turn off forever
- The LED would look like it is always on
- The LED would look like it is always off
- I don't know

7. If code to blink the LED was uploaded to both circuits with an output at pin 6, which circuit's LED would be brighter?



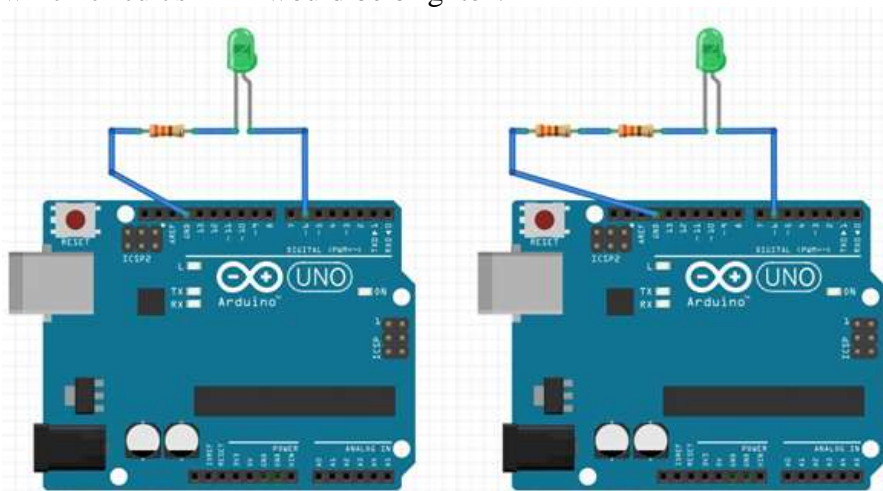
Circuit A

Circuit B

Mark only one oval.

- LED in Circuit A
 LED in Circuit B
 LED in Circuit A = LED in Circuit B
 I don't know

8. If code to blink the LED was uploaded to both circuits with an output at pin 6, which circuit's LED would be brighter?



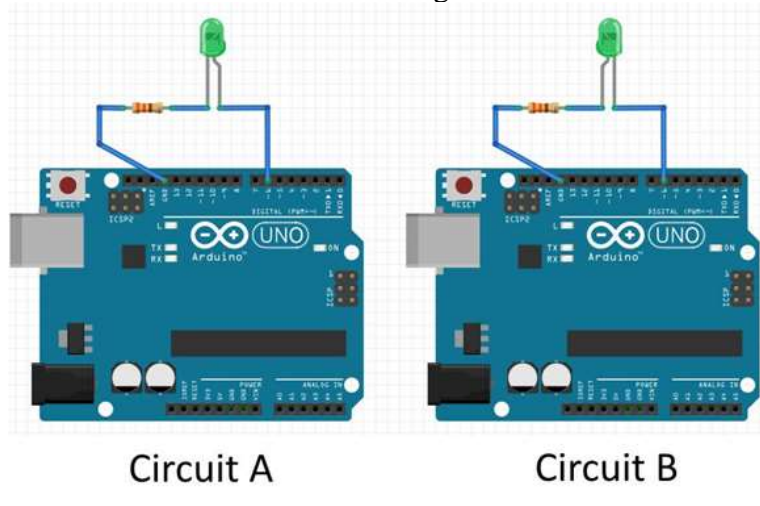
Circuit A

Circuit B

Mark only one oval.

- LED in Circuit A
 LED in Circuit B
 LED in Circuit A = LED in Circuit B
 I don't know

9. If code to blink the LED was uploaded to both circuits with an output at pin 6, which circuit's LED would be brighter?



Mark only one oval.

- LED in Circuit A
- LED in Circuit B
- LED in Circuit A = LED in Circuit B
- I don't know

10. If the following code was uploaded to a correctly built circuit with output pin 10. What would happen to the LED? The delay is in milliseconds.



Mark only one oval.

- The LED would look like it would blink every second and repeat forever
- The LED would look like it would turn on for 1 second and turn off forever
- The LED would look like it is always on
- The LED would look like it is always off
- I don't know

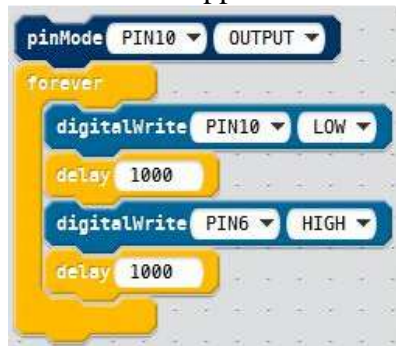
11. If the following code was uploaded to a correctly built circuit with output pin 10. What would happen to the LED? The delay is in milliseconds.



Mark only one oval.

- The LED would look like it would blink every second and repeat forever
- The LED would look like it would turn on for 1 second and turn off forever
- The LED would look like it is always on
- The LED would look like it is always off
- I don't know

12. If the following code was uploaded to a correctly built circuit with output pin 10. What would happen to the LED? The delay is in milliseconds.



Mark only one oval.

- The LED would look like it would blink every second and repeat forever
- The LED would look like it would turn on for 1 second and turn off forever
- The LED would look like it is always on
- The LED would look like it is always off
- I don't know

A.3 Demographics Survey

1. Name:
2. Age:
3. Gender:

Mark only one oval.

- Male
- Female
- Other: _____

4. Race/Ethnicity:

Check all that apply..

- White/Caucasian
- Black/African American
- Hispanic, Latino(a), or Spanish Origin
- American Indian or Alaska Native
- Asian
- Other: _____

5. Name of the college you are enrolled in:
6. Semesters Completed in College:

REFERENCES

- 16x2 LCD Keypad Shield for Arduino Version B. (2017). Retrieved January 10, 2017, from <http://store.linksprite.com/linksprite-16x2-lcd-keypad-shield-for-arduino-version-b/>
- Ahmadzadeh, M., Elliman, D., & Higgins, C. (2005). An analysis of patterns of debugging among novice computer science students. *ACM SIGCSE Bulletin*, 37(3), 84. <https://doi.org/10.1145/1151954.1067472>
- Akkerman, S. F., & Bakker, A. (2011). Boundary Crossing and Boundary Objects. *Review of Educational Research*, 81(2), 132–169. <https://doi.org/10.3102/0034654311404435>
- Alexander, P. A. (2004). A Model of Domain Learning: Reinterpreting Expertise as a Multidimensional, Multistage Process. In *Motivation, emotion, and cognition: Integrative perspectives on intellectual functioning and development* (pp. 273–298).
- Au, K. H. (1980). Participation structures in a reading lesson with Hawaiian children: Analysis of a culturally appropriate instructional event. *Anthropology & Education Quarterly*, 11(2), 91–115.
- Au, K. H., & Kawakami, A. J. (1994). Cultural congruence in instruction. In *Teaching diverse populations: Formulating a knowledge base* (Vol. 24).
- Azevedo, F. S. (2013). The Tailored Practice of Hobbies and Its Implication for the Design of Interest-Driven Learning Environments. *Journal of the Learning Sciences*, 22(3), 462–510. <https://doi.org/10.1080/10508406.2012.730082>
- Bandura, A. (1994). *Self-efficacy*. Wiley Online Library. Retrieved from <http://onlinelibrary.wiley.com/doi/10.1002/9780470479216.corpsy0836/full>
- Bandura, A. (2006). Guide for constructing self-efficacy scales. *Self-Efficacy Beliefs of Adolescents*, 5(307–337). Retrieved from <http://books.google.com/books?hl=en&lr=&id=Cj97DKKRE7AC&oi=fnd&pg=PA307&dq=%22ambiguity+about+exactly+what+is+being+measured+or+the+level+of%22+%22object+of%22+%22courses+of+action,+setting+proximal+goals+to+guide+one%E2%80%99s+efforts,%22+%22so+that+skills+in+dissimilar+domains+are+developed%22+&ots=cG3WrVqGt8&sig=Rl9MrHeH4qGx8zETu7Ay7dnetcU>
- Bandura, A., Barbaranelli, C., Caprara, G. V., & Pastorelli, C. (2001). Self-Efficacy Beliefs as Shapers of Children's Aspirations and Career Trajectories. *Child Development*, 72(1), 187–206.

- Bardi, A., & Schwartz, S. H. (2003). Values and behavior: Strength and structure of relations. *Personality and Social Psychology Bulletin*, 29(10), 1207–1220.
- Barkhuus, L. (2005). Picking Pockets on the Lawn: the development of tactics and strategies in a mobile game. *UbiComp 2005: Ubiquitous Computing*, 358–374.
- Barron, B.J.S., Schwartz, D. L., Vye, N. J., Moore, A., Petrosino, A., Zech, L., & Bransford, J. D. (1998). Doing with understanding: Lessons from research on problem- and project-based learning. *Journal of the Learning Sciences*, 7(3), 271–311.
- Barron, Brigid J.S., Schwartz, D. L., Vye, N. J., Moore, A., Petrosino, A., Zech, L., & Bransford, J. D. (1998). Doing With Understanding: Lessons From Research on Problem- and Project-Based Learning. *Journal of the Learning Sciences*, 7(3–4), 271–311. <https://doi.org/10.1080/10508406.1998.9672056>
- Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: blocks and beyond. *Communications of the ACM*, 60(6), 72–80. <https://doi.org/10.1145/3015455>
- Bdeir, A. (2009). Electronics as material: littleBits. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (pp. 397–400). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1517743>
- Beheshti, E., Aljuhani, A., & Horn, M. S. (2014). Electrons to light bulbs: Understanding electricity with a multi-level simulation environment. In *Frontiers in Education Conference (FIE), 2014 IEEE* (pp. 1–8). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7044047
- Betz, N. E. (1994). Self-concept theory in career development and counseling. *The Career Development Quarterly*, 43(1), 32–42.
- Bijker, W. E. (1997). *Of bicycles, bakelites, and bulbs: toward a theory of sociotechnical change*. MIT Press.
- Blikstein, P. (2008). Travels in Troy with Freire. *Social Justice Education for Teachers: Paulo Freire and the Possible Dream*, 205–244.
- Blikstein, P. (2013a). Digital fabrication and ‘making’ in education: The democratization of invention. *FabLabs: Of Machines, Makers and Inventors*, 1–21.
- Blikstein, P. (2013b). Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 173–182). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=2485786>

- Blikstein, P. (2015). Computationally Enhanced Toolkits for Children: Historical Review and a Framework for Future Design. *Foundations and Trends® in Human-Computer Interaction*, 9(1), 1–68. <https://doi.org/10.1561/11000000057>
- Blikstein, P., & Sipitakiat, A. (2011). QWERTY and the art of designing microcontrollers for children. In *Proceedings of the 10th International Conference on Interaction Design and Children* (pp. 234–237). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1999070>
- Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J., Guzdial, M., & Palincsar, A. (1991). Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning. *Educational Psychologist*, 26(3&4), 369–398.
- Bonsignore, E., Ahn, J., Clegg, T., Guha, M. L., Yip, J. C., Druin, A., & Hourcade, J. P. (2013). Embedding participatory design into designs for learning: An untapped interdisciplinary resource. In *Proc. CSCL* (Vol. 13). Retrieved from http://www.academia.edu/download/31596228/PDsPotential_CSCL2013_CameraReadyFINAL.pdf
- Booth, T., & Stumpf, S. (2013). End-user experiences of visual and textual programming environments for Arduino. *International Symposium on End User Development*.
- Booth, T., Stumpf, S., Bird, J., & Jones, S. (2016). Crossed Wires: Investigating the Problems of End-User Developers in a Physical Computing Task (pp. 3485–3497). ACM Press. <https://doi.org/10.1145/2858036.2858533>
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*. Retrieved from <http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf>
- Buechley, L., & Eisenberg, M. (2008). The LilyPad Arduino: Toward wearable engineering for everyone. *Pervasive Computing, IEEE*, 7(2), 12–15.
- Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008). The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 423–432). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1357123>
- Butts, W. (1985). Children's understanding of electric current in three countries. *Research in Science Education*, 15(1), 127–130.
- Byrne, B. M., & Shavelson, R. J. (1987). Adolescent Self-Concept: Testing the Assumption of Equivalent Structure Across Gender. *American Educational Research Journal*, 24(3), 365–385. <https://doi.org/10.3102/00028312024003365>

- Byrne, Barbara M, & Shavelson, R. J. (n.d.). On the Structure of Adolescent Self-Concept, 8.
- Camarata, K., Gross, M., & Do, E. Y.-L. (2003). A physical computing studio: Exploring computational artifacts and environments. *International Journal of Architectural Computing, 1*(2), 169–190.
- Cancian, F. M. (1975). *What are norms?: a study of beliefs and action in a Maya community*. Cambridge University Press.
- Chailangka, M., Sipitakiat, A., & Blikstein, P. (2017). Designing a Physical Computing Toolkit to Utilize Miniature Computers: A Case Study of Selective Exposure (pp. 659–665). ACM Press. <https://doi.org/10.1145/3078072.3084339>
- Chalmers, M. (2003). Seamful design: showing the seams in wearable computing (Vol. 2003, pp. 11–16). IEE. <https://doi.org/10.1049/ic:20030140>
- Chan, J., Pondicherry, T., & Blikstein, P. (2013). LightUp: an augmented, learning platform for electronics. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 491–494). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=2485812>
- Chi, M. T. H. (2005). Commonsense Conceptions of Emergent Processes: Why Some Misconceptions Are Robust. *Journal of the Learning Sciences, 14*(2), 161–199. https://doi.org/10.1207/s15327809jls1402_1
- Chu, S. L., Saenz, M., & Quek, F. (2016). Connectors in Maker Kits: Investigating Children’s Motor Abilities in Making (pp. 452–462). ACM Press. <https://doi.org/10.1145/2930674.2930714>
- Cohen, R. (1983). Potential difference and current in simple electric circuits: A study of students’ concepts. *American Journal of Physics, 51*(5), 407. <https://doi.org/10.1119/1.13226>
- Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser, 18*, 32–42.
- Csikszentmihalyi, M., & Csikszentmihalyi, I. S. (Eds.). (1992). *Optimal experience: Psychological studies of flow in consciousness*. Cambridge University Press.
- De Vries, H., Elliott, M. N., Kanouse, D. E., & Teleki, S. S. (2008). Using Pooled Kappa to Summarize Interrater Agreement across Many Items. *Field Methods, 20*(3), 272–282. <https://doi.org/10.1177/1525822X08317166>
- Deitrick, E., Shapiro, R. B., Ahrens, M. P., Fiebrink, R., Lehrman, P. D., & Farooq, S. (2015a). Using Distributed Cognition Theory to Analyze Collaborative Computer

- Science Learning (pp. 51–60). ACM Press.
<https://doi.org/10.1145/2787622.2787715>
- Deitrick, E., Shapiro, R. B., Ahrens, M. P., Fiebrink, R., Lehrman, P. D., & Farooq, S. (2015b). Using Distributed Cognition Theory to Analyze Collaborative Computer Science Learning (pp. 51–60). ACM Press.
<https://doi.org/10.1145/2787622.2787715>
- DesPortes, K., Anupam, A., Pathak, N., & DiSalvo, B. (2016). BitBlox: A Redesign of the Breadboard. *Proceedings of the The 15th International Conference on Interaction Design and Children*, 255–261.
- DesPortes, K., & DiSalvo, B. (2017). Where are the Glass-Boxes?: Examining the Spectrum of Modularity in Physical Computing Hardware Tools (pp. 292–297). ACM Press. <https://doi.org/10.1145/3078072.3079733>
- DesPortes, K., Pathak, N., & Anupam, A. (2016). Circuit Diagrams Vs. Physical Circuits: The Effect of Representational Forms During Assessment. *Frontiers in Education*.
- DesPortes, K., Spells, M., & DiSalvo, B. (2016a). Interdisciplinary Computing and the Emergence of Boundary Objects: A Case-Study of Dance and Technology. *International Society of the Learning Sciences*.
- DesPortes, K., Spells, M., & DiSalvo, B. (2016b). The MoveLab: Developing Congruence Between Students’ Self-Concepts and Computing (pp. 267–272). ACM Press. <https://doi.org/10.1145/2839509.2844586>
- DesPortes, K., Spells, M., & DiSalvo, B. (2016c). The MoveLab: Developing Congruence Between Students’ Self-Concepts and Computing (pp. 267–272). ACM Press. <https://doi.org/10.1145/2839509.2844586>
- Dewey, J. (1938). *Experience and education*. Kappa Delta Pi.
- DiMaggio, P. (1997). Culture and Cognition. *Annual Review of Sociology*, 23(1), 263–287.
- DiSalvo, B., & DiSalvo, C. (2014). Designing for Democracy in Education: Participatory Design and the Learning Sciences. Presented at the 11th International Conference of the Learning Sciences.
- DiSalvo, Betsy, & DesPortes, K. (2017). Participatory Design for Value-Driven Learning. In *Participatory Design for Learning: Perspectives from Practice and Research*.
- DiSalvo, Betsy, Guzdial, M., Bruckman, A., & McKlin, T. (2014). Saving Face While Geeking Out: Video Game Testing as a Justification for Learning Computer Science. *Journal of the Learning Sciences*, 23(3).
- DiSalvo, Betsy, Yip, J., Bonsignore, E., & DiSalvo, C. (Eds.). (2017). *Participatory Design for Learning: Perspectives from Practice and Research*. Taylor & Francis.

- Dougherty, D. (2013). The Maker Mindset. In M. Honey & D. E. Kanter (Eds.), *Design, Make, Play: Growing the Next Generation of STEM Innovators* (p. 7).
- Draw Circuits Kit. (2018). Retrieved from <https://www.vat19.com/item/draw-circuits-kit>
- Dunne, A., & Raby, F. (2013). *Speculative everything: design, fiction, and social dreaming*. MIT Press.
- Eccles, J. S. (1992). The Development of Achievement Task Values: A Theoretical Analysis. In *Developmental Review* (Vol. 12, pp. 1–46).
- Eccles, J. S., & Wigfield, A. (2002). Motivational beliefs, values, and goals. *Annual Review of Psychology*, 53(1), 109–132.
- Edelson, D. C., & Joseph, D. M. (2004a). The interest-driven learning design framework: motivating learning through usefulness (pp. 166–173). Presented at the Proceedings of the 6th international conference on Learning sciences, International Society of the Learning Sciences.
- Edelson, D. C., & Joseph, D. M. (2004b). The interest-driven learning design framework: motivating learning through usefulness (pp. 166–173). Presented at the Proceedings of the 6th international conference on Learning sciences, International Society of the Learning Sciences.
- Edwards, S. H. (2004). Using software testing to move students from trial-and-error to reflection-in-action. *ACM SIGCSE Bulletin*, 36(1), 26. <https://doi.org/10.1145/1028174.971312>
- Eglash, R., Gilbert, J. E., Taylor, V., & Geier, S. R. (2013). Culturally Responsive Computing in Urban, After-School Contexts Two Approaches. *Urban Education*, 48(5), 629–656.
- Ehn, P. (2008). Participation in design things (Vol. 8). Presented at the Proceedings of the Tenth Anniversary Conference on Participatory Design, ACM.
- Ehn, Pelle. (2008). Participation in Design Things. Proceedings of the Tenth Anniversary Conference on Participatory Design.
- Eisenberg, M., Eisenberg, A., Gross, M., Kaowthumrong, K., Lee, N., & Lovett, W. (2002). Computationally-enhanced construction kits for children: Prototype and principles. *Proceedings of the Fifth International Conference of the Learning Sciences*, 23–26.
- Engelhardt, P. V., & Beichner, R. J. (2004). Students' understanding of direct current resistive electrical circuits. *American Journal of Physics*, 72(1), 98. <https://doi.org/10.1119/1.1614813>

- Ericsson, A., Krampe, R. T., & Tesch-Römer, C. (1993). The Role of Deliberate Practice in the Acquisition of Expert Performance. *Psychological Review*, 100(3), 363–406.
- Ericsson, A., & Simon, H. A. (1980). Verbal Reports as Data + Ericsson.pdf. *Psychological Review*, 87(3), 215.
- Evans, J. (2003). In two minds: dual-process accounts of reasoning. *Trends in Cognitive Sciences*, 7(10), 454–459.
- Fischer, G. (2013). Meta-Design: Empowering All Stakeholder as Co-Designers. *Handbook of Design in Educational Technology*, 135.
- Franklin, D., Conrad, P., Aldana, G., & Hough, S. (2011). Animal tlatoque: attracting middle school students to computing through culturally-relevant themes. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 453–458). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1953295>
- Freeman, J., Magerko, B., McKlin, T., Reilly, M., Permar, J., Summers, C., & Fruchter, E. (2014). Engaging underrepresented groups in high school introductory computing through computational remixing with EarSketch (pp. 85–90). ACM Press. <https://doi.org/10.1145/2538862.2538906>
- Gay, G. (2010). *Culturally Responsive Teaching: Theory, Research, and Practice*. Teachers College Press.
- Geertz, C. (1973). *The Interpretation of Cultures*. New York, NY: Basic Books, Inc.
- Gibbins, K., & Walker, I. (1993). Multiple interpretations of the Rokeach Value Survey. *The Journal of Social Psychology*, 133(6), 797–805.
- Goffman, E. (1956). *The presentation of self in everyday life*. New York, NY: Doubleday.
- Goldman, R., Eguchi, A., & Sklar, E. (2004). Using educational robotics to engage inner-city students with technology. In *Proceedings of the 6th international conference on Learning sciences* (pp. 214–221). International Society of the Learning Sciences. Retrieved from <http://dl.acm.org/citation.cfm?id=1149151>
- Graeber, D. (2001). *Toward an anthropological theory of value: The false coin of our own dreams*. New York: Palgrave.
- Granott, N. (1993). *Microdevelopment of co-construction of knowledge during problem solving: puzzled minds, weird creatures, and wuggles*. Massachusetts Institute of Technology. Retrieved from <http://dspace.mit.edu/handle/1721.1/29069>
- Grotzer, T. A., & Sudbury, M. (2000). Moving beyond underlying linear causal models of electrical circuits. In *annual meeting of the National Association of Research in Science Teaching, New Orleans, LA*. Citeseer. Retrieved from

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.977&rep=rep1&type=pdf>

- Hartmann, B., MacDougall, D., Brandt, J., & Klemmer, S. R. (n.d.). What Would Other Programmers Do? Suggesting Solutions to Error Messages, 10.
- Hatch, M. (2014). *The maker movement manifesto*. McGraw-Hill Education.
- Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher*, 30(3), 141–158.
- Hidi, S. (2000). An Interest Researcher's Perspective: The Effects of Extrinsic and Intrinsic Factors on Motivation. In *Extrinsic and Intrinsic Factors on Motivation*.
- Hidi, S., & Renninger, K. A. (2006). The Four-Phase Model of Interest Development. *Educational Psychologist*, 41(2), 111–127. https://doi.org/10.1207/s15326985ep4102_4
- High Sensitivity Sound Detector. (2017). Retrieved from <http://www.electroma.ma/capteurs/79-detecteur-de-son-haute-sensibilite-picarduinoavr.html>
- Hmelo, C. E., & Guzdial, M. (1996). Of black and glass boxes: Scaffolding for doing and learning. In *Proceedings of the 1996 international conference on Learning sciences* (pp. 128–134). International Society of the Learning Sciences. Retrieved from <http://dl.acm.org/citation.cfm?id=1161153>
- Hultén, M. (2013). Boundary objects and curriculum change: the case of integrated versus subject-based teaching. *Journal of Curriculum Studies*, 45(6), 790–813. <https://doi.org/10.1080/00220272.2013.812245>
- Jaakkola, T., & Nurmi, S. (2004). Academic impact of learning objects: The case of electric circuits. Retrieved from <http://www.leeds.ac.uk/educol/documents/00003702.htm>
- James DiSalvo, B., Yardi, S., Guzdial, M., McKlin, T., Meadows, C., Perry, K., & Bruckman, A. (2011). African American men constructing computing identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2967–2970). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1979381>
- Johnson, S., & Thomas, A. P. (2010). Squishy circuits: a tangible medium for electronics education. *Human Factors in Computing Systems*, 4099–4104.
- Jung, M. F., Martelaro, N., Hoster, H., & Nass, C. (2014). Participatory materials: having a reflective conversation with an artifact in the making (pp. 25–34). ACM Press. <https://doi.org/10.1145/2598510.2598591>
- Kaczmarczyk, L. C., Petrick, E. R., East, J. P., & Herman, G. L. (2010). Identifying student misconceptions of programming. In *Proceedings of the 41st ACM technical*

- symposium on Computer science education* (pp. 107–111). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1734299>
- Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., & Lui, D. (2014). A Crafts-Oriented Approach to Computing in High School: Introducing Computational Concepts, Practices, and Perspectives with Electronic Textiles. *ACM Transactions on Computing Education*, *14*(1), 1–20. <https://doi.org/10.1145/2576874>
- Kahneman, D., & Tversky, A. (1982). On the study of statistical intuitions. *Cognition*, *11*(2), 123–141.
- Kalish, J. (2011, December 10). Libraries Make Room For High-Tech “Hackerspaces.” Retrieved November 11, 2015, from <http://www.npr.org/2011/12/10/143401182/libraries-make-room-for-high-tech-hackerspaces>
- Ko, A. J., Myers, B. A., & Aung, H. H. (2004). Six learning barriers in end-user programming systems. *Visual Languages and Human Centric Computing, 2004 IEEE Symposium On*, 199–206.
- Küçüközer, H., & Kocakulah, S. (2007). Secondary school students’ misconceptions about simple electric circuits. *Journal of Turkish Science Education*, *4*(1), 101.
- Kwak, S. S., Kang, H., Lee, H., & Wu, C. (2016). PaperBot: The intelligent paper toy. *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 551.
- Ladson-Billings, G. (1995). Toward a Theory of Culturally Relevant Pedagogy. *American Educational Research Journal*, *32*(3), 465–491. <https://doi.org/10.3102/00028312032003465>
- Ladson-Billings, Gloria. (1990). Culturally relevant teaching: Effective instruction for black students. *The College Board Review*, *155*, 20–25.
- Ladson-Billings, Gloria. (1995a). But that’s just good teaching! The case for culturally relevant pedagogy. *Theory into Practice*, *34*(3), 159–165.
- Ladson-Billings, Gloria. (1995b). Toward a theory of culturally relevant pedagogy. *American Educational Research Journal*, *32*(3), 465–491.
- Ladson-Billings, Gloria. (2008). "Yes, but how do we do it?": Practicing culturally relevant pedagogy. In *City kids, city schools: More reports from the front row* (pp. 162–177).
- Lamont, M., Schmalzbaure, J., & Waller, M. (1996). Cultural and moral boundaries in the United States: Structural position, geographic location, and lifestyle explanations. *Poetics*, *24*(1), 31–56.

- Latulipe, C., & Huskey, S. (2008). Dance. Draw: exquisite interaction. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 2* (pp. 47–51). British Computer Society. Retrieved from <http://dl.acm.org/citation.cfm?id=1531839>
- Laws, P. W. (1991). Calculus-based physics without lectures. *Physics Today*, 44(12), 24–31.
- Lee, K. C. (1991). The problem of appropriateness of the Rokeach Value Survey in Korea. *International Journal of Psychology*, 26(3), 299–310.
- Lee, M. J., & Ko, A. J. (n.d.). Personifying Programming Tool Feedback Improves Novice Programmers' Learning, 8.
- Leigh Star, S. (2010). This is Not a Boundary Object: Reflections on the Origin of a Concept. *Science, Technology & Human Values*, 35(5), 601–617. <https://doi.org/10.1177/0162243910377624>
- LilyPad Light Sensor. (2017). Retrieved from <https://solarbotics.com/product/50486/>
- Lincoln, Y., & Guba, E. (1985). *Naturalistic Inquiry*. Beverly Hills, CA: Sage Publications, Inc.
- Lizardo, O., & Strand, M. (2010). Skills, toolkits, contexts and institutions: Clarifying the relationship between different approaches to cognition in cultural sociology. *Poetics*, 38(2), 205–228.
- Maher, M. L., & Gomez de Silva Garza, A. (1997). Case-based reasoning in design. *IEEE Expert: Intelligent Systems and Their Applications*, 12(2), 34–41.
- Malone, T. W. (1981). Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 5(4), 333–369.
- Margolis, J. (2008). *Stuck in the shallow end: Education, race, and computing*. Cambridge, MA: The MIT Press.
- Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: MIT Press.
- Margolis, Jane. (2008). *Stuck in the shallow end: Education, race, and computing*. MIT Press.
- Martin, C. D. (2004). Draw a computer scientist. In *ACM SIGCSE Bulletin* (Vol. 36, pp. 11–12). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1041628>
- Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). Habits of programming in scratch (p. 168). ACM Press. <https://doi.org/10.1145/1999747.1999796>

- Mellis, D. A., Jacoby, S., Buechley, L., Perner-Wilson, H., & Qi, J. (2013). Microcontrollers as material: crafting circuits with paper, conductive ink, electronic components, and an untoolkit. *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*, 83–90.
- Mohomed, I., & Dutta, P. (2015). THE Age of DIY and Dawn of the Maker Movement. *GetMobile: Mobile Computing and Communications*, 18(4), 41–43.
- Montemayor, R., & Eisen, M. (1977). The development of self-conceptions from childhood to adolescence. *Developmental Psychology*, 13(4), 314.
- Murphy, L., Lewandowski, G., & McCauley, R. (n.d.). Debugging: The Good, the Bad, and the Quirky – a Qualitative Analysis of Novices’ Strategies, 5.
- Noble, S. U. (2018). *Algorithms of Oppression: How search engines reinforce racism*. NYU Press.
- NSF. (2017). *Women, Minorities, and Persons with Disabilities in Science and Engineering*. Arlington, VA: National Science Foundation, National Center for Science and Engineering Statistics (NCSES). Retrieved from <https://www.nsf.gov/statistics/2017/nsf17310/>
- Osborne, R. (1983). Towards modifying children’s ideas about electric current. *Research in Science & Technological Education*, 1(1), 73–82.
- Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.
- Papert, Seymour. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc. Retrieved from <http://dl.acm.org/citation.cfm?id=1095592>
- Papert, Seymour, & Harel, I. (1991). Situating Constructionism. *Constructionism*.
- Pea, R. D. (1984). Language-Independent Conceptual bugs in novice programming. *Technical Report No. 31*.
- Peppler, K., & Glosson, D. (2013). Stitching Circuits: Learning About Circuitry Through E-textile Materials. *Journal of Science Education and Technology*, 22(5), 751–763. <https://doi.org/10.1007/s10956-012-9428-2>
- Perlman, R. (1974). "TORTIS: Toddler’s Own Recursive Turgle Interpreter System.
- Polman, J. L., & Hope, J. M. G. (2014). Science news stories as boundary objects affecting engagement with science: SCIENCE NEWS STORIES. *Journal of Research in Science Teaching*, 51(3), 315–341. <https://doi.org/10.1002/tea.21144>
- Puntambekar, S., & Kolodner, J. L. (2004). Distributed Scaffolding: Helping Students Learn in a ‘Learning by Design’ Environment. *Sadhana*, 6269. Retrieved from

https://www.researchgate.net/profile/Sadhana_Puntambekar/publication/239063299_Distributed_Scaffolding_Helping_Students_Learn_in_a_'Learning_by_Design'_Environment/links/542d3a330cf29bbc126d2221.pdf

- Putnam, R., & others. (1984). A Summary of Misconceptions of High School BASIC Programmers. Technology Panel Study of Stanford and the Schools. Occasional Report# 010. Retrieved from <http://eric.ed.gov/?id=ED258556>
- Qi, J., Huang, A. “bunnie,” & Paradiso, J. (2015). Crafting technology with circuit stickers (pp. 438–441). ACM Press. <https://doi.org/10.1145/2771839.2771873>
- Raffle, H. S., Parkes, A. J., & Ishii, H. (2004). Topobo: A Constructive Assembly System with Kinetic Memory. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.
- Rahimi, E., van den Berg, J., & Veen, W. (2015). Facilitating student-driven constructing of learning environments using Web 2.0 personal learning environments. *Computers & Education*, 81, 235–246. <https://doi.org/10.1016/j.compedu.2014.10.012>
- Renninger, A., Hidi, S., & Krapp, A. (2014). *The role of interest in learning and development*. Psychology Press.
- Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., ... Silverman, B. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60–67.
- Resnick, Mitchel, & Ocko, S. (1990). LEGO/logo--learning through and about design. Epistemology and Learning Group, MIT Media Laboratory.
- Rick, J. (2012). Proportion: a tablet app for collaborative learning (p. 316). ACM Press. <https://doi.org/10.1145/2307096.2307155>
- Rist, R. S. (1989). Schema Creation in Programming. *Cognitive Science*, 13(3), 389–414. https://doi.org/10.1207/s15516709cog1303_3
- Rist, R. S. (1991). Knowledge Creation and Retrieval in Program Design: A Comparison of Novice and intermediate Student Programmers. *Human-Computer Interaction*, 6(1), 1–46. https://doi.org/10.1207/s15327051hci0601_1
- Rogoff, B. (1994). Developing understanding of the idea of communities of learners. In *Mind, Culture, and Activity* (pp. 209–229).
- Rogoff, Barbara. (2001). *Learning Together: Children and Adults in a School Community*. USA: Oxford University Press.
- Rokeach, M. (1979). *Understanding Human Values*. Simon and Schuster.

- Rosner, D. K., & Ryokai, K. (2008). Spyn: augmenting knitting to support storytelling and reflection. In *Proceedings of the 10th international conference on Ubiquitous computing* (pp. 340–349). ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=1409682>
- Sadler, J., Shluzas, L., & Blikstein, P. (2017). Building blocks in creative computing: modularity increases the probability of prototyping novel ideas. *International Journal of Design Creativity and Innovation*, 5(3–4), 168–184. <https://doi.org/10.1080/21650349.2015.1136796>
- Sadler, J., Shluzas, L., Blikstein, P., & Katila, R. (2016). Building Blocks of the Maker Movement: Modularity Enhances Creative Confidence During Prototyping. In H. Plattner, C. Meinel, & L. Leifer (Eds.), *Design Thinking Research* (pp. 141–154). Cham: Springer International Publishing. Retrieved from http://link.springer.com/10.1007/978-3-319-19641-1_10
- Saldaña, J. (2012). *The coding manual for qualitative researchers*.
- Saussure, F. de. (1966). *Course in General Linguistics*. New York: McGraw Hill.
- Schmidtbauer, M., Johnson, S., Jalkio, J., & Thomas, A. P. (2012). Squishy Circuits as a Tangible Interface. *Human Factors in Computing Systems*, 2111–2116.
- Schön, D. A. (1987). *Educating the reflective practitioner: Toward a new design for teaching and learning in the professions*. Jossey-Bass.
- Schulz, S. (2016). Improving Scientific Inquiry through Physical Computing (pp. 289–290). ACM Press. <https://doi.org/10.1145/2960310.2960352>
- Schulz, S., & Pinkwart, N. (2016). Towards Supporting Scientific Inquiry in Computer Science Education (pp. 45–53). ACM Press. <https://doi.org/10.1145/2978249.2978255>
- Schwartz, S. (1999). A theory of cultural values and some implications for work. *Values and Work: A Special Issue of the Journal Applied Psychology*, 48(1), 23–47.
- Schwartz, S. H. (1992). Universals In The Content and Structure of Values: Theoretical Advances and Empirical Tests in 20 Countries. *Advances in Experimental Social Psychology*, 25, 1–65.
- Schwartz, S. H., & Bilsky, W. (1987). Toward A Universal Psychological Structure of Human Values. *Journal of Personality and Social Psychology*, 53(5), 550–562.
- Scott, K. A., Clark, K., Hayes, E., Mruzek, C., & Sheridan, K. (2010). Culturally Relevant Computing Programs: Two examples to Inform Teacher Professional Development. *Society for Information Technology & Teacher Education International Conference*, (1), 1269–1277.

- Scott, K. A., & Hood, D. W. (2009). CompuGirls: Designing a Culturally Relevant Technology Program. *Educational Technology*, 49(6), 34–39.
- Searle, K. A., Fields, D. A., Lui, D. A., & Kafai, Y. B. (2014). Diversifying high school students' views about computing with electronic textiles (pp. 75–82). ACM Press. <https://doi.org/10.1145/2632320.2632352>
- Searle, K. A., & Kafai, Y. B. (2015). Culturally Responsive Making with American Indian Girls: Bridging the Identity Gap in Crafting and Computing with Electronic Textiles (pp. 9–16). ACM Press. <https://doi.org/10.1145/2807565.2807707>
- Searle, K., & Kafai, Y. B. (2015). Boys' Needlework: Understanding Gendered and Indigenous Perspectives on Computing and Crafting with Electronic Textiles. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, 31–39.
- Shaffer, D. W., & Resnick, M. (1999). "Thick" authenticity: New media and authentic learning. *Journal of Interactive Learning Research*, 10(2), 195–215.
- Shashaani, L. (1993). Gender-Based Differences in Attitudes Toward Computers. *Computers Education*, 20(2), 169–181.
- Shavelson, R. J., Hubner, J. J., & Stanton, G. C. (1976). Self-Concept: Validation of Construct Interpretations. *Review of Educational Research*, 46(3), 407. <https://doi.org/10.2307/1170010>
- Sheridan, K. M., Halverson, E. R., Litts, B., Brahms, L., Jacobs-Priebe, L., & Owens, T. Learning in the Making: A Comparative Case Study of Three Makerspaces. *Harvard Educational Review*.
- Simonsen, J., & Robertson, T. (2012). *Routledge handbook of participatory design*. Routledge.
- Sipitakiat, A., Blikstein, P., & Cavallo, D. P. (2004). GoGo board: augmenting programmable bricks for economically challenged audiences. In *Proceedings of the 6th international conference on Learning sciences* (pp. 481–488). International Society of the Learning Sciences. Retrieved from <http://dl.acm.org/citation.cfm?id=1149185>
- Sleeman, D., & others. (1984). Pascal and High-School Students: A Study of Misconceptions. Technology Panel Study of Stanford and the Schools. Occasional Report# 009. Retrieved from <http://eric.ed.gov/?id=ED258552>
- Snap Circuits. (2017). Retrieved January 26, 2017, from <http://www.snapcircuits.net/>
- Sorva, J. (2012). *Visual program simulation in introductory programming education*. Espoo: Aalto Univ. School of Science.

- Star, S. L., & Griesemer, J. R. (1989a). Institutional ecology, translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology. *Social Studies of Science*, (19.3), 387–420.
- Star, S. L., & Griesemer, J. R. (1989b). Institutional ecology, translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(3), 387–420.
- Stefik, A., & Gellenbeck, E. (2011). Empirical studies on programming language stimuli. *Software Quality Journal*, 19(1), 65–99. <https://doi.org/10.1007/s11219-010-9106-7>
- Stefik, A., & Siebert, S. (2013). An Empirical Investigation into Programming Language Syntax. *ACM Transactions on Computing Education*, 13(4), 1–40. <https://doi.org/10.1145/2534973>
- Steinberg, L., & Morris, A. S. (2001). Adolescent Development. *Journal of Cognitive Education and Psychology*, 2(1), 55–87.
- Swann, W. B. (1983). Self-Verification: Bringing Social Reality into Harmony with the Self. In *Sociological Perspectives on The Self* (pp. 33–66).
- Swidler, A. (1986). Culture in action: Symbols and strategies. *American Sociological Review*, 51(2), 273–286.
- Swidler, Ann. (1986). Culture in Action: Symbols and Strategies. *American Sociological Review*, 51(2), 273. <https://doi.org/10.2307/2095521>
- Swidler, Ann. (2000). Cultural power and social movements. In *Culture and Politics* (pp. 269–283). Palgrave Macmillan.
- Swidler, Ann. (2013). *Talk of Love: How Culture Matters*. University of Chicago Press.
- Tanenbaum, J., Tanenbaum, K., & Antle, A. (2010). The Reading Glove: designing interactions for object-based tangible storytelling (pp. 1–9). ACM Press. <https://doi.org/10.1145/1785455.1785474>
- Thames & Kosmos. (2018). Electricity Master Lab: The Physics of Electricity. Retrieved from <http://www.thamesandkosmos.com/index.php/product/category/science-kits/electricity-master-lab>
- Vaisey, S. (2008). Socrates, Skinner, and Aristotle: Three ways of thinking about culture in action. *Sociological Forum*, 23(3), 603–613.
- Vaisey, S. (2009). Motivation and justification: A dual-process model of culture in action. *American Journal of Sociology*, 114(6), 1675–1715.

- van den Haak, M., De Jong, M., & Jan Schellens, P. (2003). Retrospective vs. concurrent think-aloud protocols: Testing the usability of an online library catalogue. *Behaviour & Information Technology*, 22(5), 339–351. <https://doi.org/10.1080/0044929031000>
- van Gog, T., Paas, F., van Merriënboer, J. J. G., & Witte, P. (2005). Uncovering the Problem-Solving Process: Cued Retrospective Reporting Versus Concurrent and Retrospective Reporting. *Journal of Experimental Psychology: Applied*, 11(4), 237–244. <https://doi.org/10.1037/1076-898X.11.4.237>
- Velleman. (2018). Electronic Lab Kit 130 in 1. Retrieved from <https://www.velleman.eu/products/view/?id=341607>
- Weintrop, D., & Wilensky, U. (2015a). To block or not to block, that is the question: students' perceptions of blocks-based programming (pp. 199–208). ACM Press. <https://doi.org/10.1145/2771839.2771860>
- Weintrop, D., & Wilensky, U. (2015b). Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs (pp. 101–110). ACM Press. <https://doi.org/10.1145/2787622.2787721>
- Weintrop, D., & Wilensky, U. (2015c). Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs (pp. 101–110). ACM Press. <https://doi.org/10.1145/2787622.2787721>
- Wenger, E. (1998). *Communities of Practice: Learning, Meaning, and Identity*. New York: Cambridge University Press.
- Westervelt, E. (2016, July 21). 3 Challenges As Hands-On, DIY Culture Moves Into Schools. Retrieved December 29, 2016, from <http://www.npr.org/sections/ed/2016/07/21/486046973/three-challenges-as-more-makers-move-into-schools>
- Whitehouse, H. (2004). *Modes of religiosity: A cognitive theory of religious transmission*. Walnut Creek, CA: Altamira Press.
- William, J. (1899). *Talks to Teachers on Psychology and to Students on Some of Life's Ideals* (Vol. 12). Harvard University Press.
- Winner, L. (1986). Do Artifacts Have Politics? In *In The whale and the reactor: a search for limits in an age of high technology* (pp. 19–39). Chicago: University of Chicago Press.
- Winner, Langdon. (1993). Upon opening the black box and finding it empty: Social constructivism and the philosophy of technology. *Science, Technology, & Human Values*, 18(3), 362–378.
- (2015, March 12). Retrieved from <http://www.bbc.com/news/technology-31834927>