

Aligning Capabilities of Interactive Educational Tools to Learner Goals

Tom Lauwers

CMU-RI-TR-10-09

*Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy in Robotics*

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

*Submitted in partial fulfillment of
the requirements of the Program for
Interdisciplinary Education Research (PIER)*

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:

Illah Nourbakhsh, Chair, Carnegie Mellon Robotics Institute
Sharon Carver, Carnegie Mellon Psychology Department
John Dolan, Carnegie Mellon Robotics Institute,
Fred Martin, University of Massachusetts Lowell Computer Science

© 2010, Tom Lauwers.

ABSTRACT

This thesis is about a design process for creating *educationally relevant* tools. I submit that the key to creating tools that are educationally relevant is to focus on ensuring a high degree of alignment between the designed tool and the broader educational context into which the tool will be integrated. The thesis presents methods and processes for creating a tool that is both well aligned and relevant.

The design domain of the thesis is described by a set of tools I refer to as “Configurable Embodied Interfaces”. Configurable embodied interfaces have a number of key features, they:

- Can sense their local surroundings through the detection of such environmental and physical parameters as light, sound, imagery, device acceleration, etc.
- Act on their local environment by outputting sound, light, imagery, motion of the device, etc.
- Are configurable in such a way as to link these inputs and outputs in a nearly unlimited number of ways.
- Contain active ways for users to either directly create new programs linking input and output, or to easily re-configure them by running different programs on them.
- Are user focused; they assume that a human being is manipulating them in some way, through affecting input and observing output of the interface.

Spurred by the growth of cheap computation and sensing, a large number of educational programs have been built around use of configurable embodied interfaces in the last three decades. These programs exist in both formal and informal educational settings and are in use from early childhood through adult and community education. Typically, configurable embodied interfaces are used as tools in three major and sometimes overlapping areas: computer Science education, creative and engineering design education, and traditional science and math education.

This work details three examples of collaborations between technologists and educators that led to the creation of educationally successful tools; these three examples share a focus on creating a configurable embodied interface to tackle a specific cognitive and affective set of learning goals, but differ completely in the location of the learning environment, the age and interests of the learners, and the nature of the learning goals. Through the exploration of the methods used, an analysis of the general and context-specific features of the design processes of the three

ABSTRACT

accounts, and a comparison of the process used in this thesis to a conventional engineering design process, this work provides case studies and a set of guidelines that can inform technologists interested in designing educationally relevant embodied interfaces.

ACKNOWLEDGMENTS

I would first and foremost like to thank my thesis advisor, Dr. Illah Nourbakhsh. He has been the ideal advisor for me. Illah is extremely helpful, especially at resolving the inevitable roadblocks that occur during doctoral studies. He encourages his students and staff to maintain a healthy work/life balance, something that as a new father I greatly appreciate. He has been considerate and fully supportive of my personal interests and pursuits (one of which resulted in a chapter in this thesis). On top of all of this he has a moral compass that I try hard to emulate, and a sense of humility that will cause him to reflexively dismiss this entire paragraph.

My thesis committee has contributed significantly and substantially to this thesis. I would like to thank Sharon Carver for exposing me to curricular design and alignment through her class, Educational Goals, Instruction, and Assessment, and for offering to meet with me consistently throughout the thesis process. Fred Martin's detailed comments, both in writing and during extensive phone conversations, have strengthened the concluding analysis significantly. Thanks as well to John Dolan for a detailed read, catching numerous spelling and grammatical errors, as well as for high-level comments on alignment in both the proposal and thesis that have strengthened my treatment of that subject.

I enjoyed the good fortune to work with interdisciplinary teams of engineers, designers, and educators in each of the projects described in the thesis. For the Braille Tutor project, I would like to thank Nidhi Kalra, for initiating the dialogue with teachers at the Mathru school for the blind, thinking of the idea, and, crucially for me, asking me if I would like to help her design the hardware for the project. The project's growth and funding are attributable to TechBridgeWorld, especially Bernardine Dias, whose management of the project and guidance in how to disseminate it have provided me with useful examples to follow, and Sarah Belousov and Ermine Teves, who quickly resolved any day-to-day problems that came up. I'd also like to thank Tom Stepleton and Daniel Dewey, for their work advancing the tutor software, and the many students in TechBridgeWorld internships that have tested the tutor in numerous locales. Lastly and most importantly, I'd like to thank the teachers and students at the Mathru School for the Blind.

For Robot Diaries, I would like to thank Emily Hamner and Debbie Bernstein, without whom Robot Diaries would never have begun or been sustained. We have worked together on the project for four years now, and I am glad to be blessed with such good friends and colleagues. I would like to thank Carl DiSalvo for his work in the early years of Robot Diaries and for being the only one of us capable of employing an effective 'parent' voice during the workshops. Though her participation in the project was brief, Kristen Stubbs made a permanent impact by helping us formulate the learning goals and curricula for the second round of

ACKNOWLEDGMENTS

Robot Diaries. I thank Chris Bartley for his work on the TeRK software framework and for his heroic efforts to keep the software one week ahead of the girls in the fall 2006 pilot. I would like to thank the teachers who have worked with us during the project, especially Dana Wettergreen and Kim Winbush at PALS, Barb Bianco at Falk, Ann Shoplik and Pam Piskurich at C-MITES, and Christine Nguyen at the Sarah Heinz House. Finally, a major thank you to the Heinz Endowments for supporting the program financially.

The CSbots project would not have been possible without the involvement of Don Smith of CCAC, who worked with me to design assignments, and piloted the alpha and Finch robots in his courses for three semesters. Emily Hamner helped with the initial evaluation of CS textbooks and with organizing the high school pilot. It would have been very difficult to organize a two day, thirty teacher workshop without her help. I would like to thank the teachers who participated in CSbots, especially participants of our initial pilot year: Carolyn Stewart, Norm Messa, Dianne Meskauskas, Mari Hobkirk, and David Opfer. The CSbots project was funded through the Heinz Endowments, the Arthur Vining Davis Foundation, and the National Science Foundation's CCLI program.

I enjoyed participating in two vibrant communities during my graduate studies at CMU: PIER and the Robotics Institute. The PIER program influenced me in a number of important ways. The courses and ed-bags exposed me to education research that has shaped many of the ideas in this thesis. The program itself funded half of my graduate studies, and by so doing, allowed me to move the scope of my thesis somewhat outside the bounds of a traditional robotics Ph.D. Finally, I'd like to thank members of the community for interesting and topical conversations; Jamie, Ruth, Matt, David, Jack, and many others.

The Robotics Institute has been my intellectual home since before I entered the doctoral program. I'd like to thank the many who have made this journey the most enjoyable and stimulating time of my life: Brian, Rachel, Martin, Sanjeev, Marek, Jonathan, Pras, Brad, Dean, Anna, Liz, Alex, Matt, Jean, Karen, Suzanne, Ralph, George, and many others.

My parents I thank for providing me with an idyllic childhood and all of the intellectual and moral resources needed to make my own way. I could not have accomplished this without their love and support.

My wife Krissie I thank for her understanding, patience, and support, and especially, once again, her *patience*. No one should have had to listen to so many complaints about writing, and few could have done so and responded in a sympathetic way every time. My daughter Lena I thank for being the perfect antidote to the thesis writing blues.

DEDICATION

For Lena, and all tomorrow's students.

Contents

Contents	11
1 Introduction	17
1.1 Definitions of Educational Technology	17
1.2 Configurable Embodied Interfaces	19
1.3 Thesis Statement and Contributions	20
1.3.1 Statement	21
1.3.2 Contributions	21
1.4 Organization of the Work	21
2 Configurable Embodied Interfaces in Education	23
2.1 Computer Science Education	23
2.1.1 Manual Programming	24
2.1.2 Visual and Textual Programming	26
2.2 Creative and Engineering Design	27
2.2.1 Controllers	27
2.2.2 Programs	29
2.3 Science Education	31
2.4 Summary	33
3 Methods for the Design of Configurable Embodied Interfaces	35
3.1 Alignment in Instructional Design	35
3.2 Extending Alignment	37
3.3 Participatory Design	38
3.4 Design-Based Research	40
3.5 Implications of these Methods for the Design Process	43
3.5.1 A Common Design Process	44
3.5.2 Implementation	45

CONTENTS

4	The Braille Writing Tutor	47
4.1	Writing Braille	47
4.2	Timeline of the Project	49
4.2.1	My Role on the Project	49
4.3	Ideation	50
4.3.1	Other Tutor Systems	51
4.4	Design, Pilot, and Evaluation of the First version of the Braille Tutor	51
4.4.1	Goals, Instruction, and Assessment of a Braille Writing Curriculum	52
4.4.2	Design Constraints	54
4.4.3	E-slate Design	56
4.4.4	The Tutor Software	57
4.4.5	Field Study and Evaluation	59
4.5	Second Design Cycle	65
4.5.1	E-slate Changes	65
4.5.2	Early Tutor Improvements	69
4.6	On-going Software and Curricular Improvements	70
4.6.1	Foreign Language Writing Support	71
4.6.2	Local Accents and Languages Audio Support	71
4.6.3	Motivational Games	71
4.6.4	E-slate hardware modifications	72
4.6.5	Pilots and Evaluation	73
4.7	Summary	74
5	Robot Diaries	75
5.1	Motivation and Program Goal	75
5.2	Timeline of the Project	76
5.2.1	My Role in the Project	77
5.3	Ideation	77
5.3.1	Focus Group	78
5.4	Participatory Design Sequence	81
5.4.1	Curriculum Progression	82
5.4.2	Summer Workshop	83
5.4.3	One-Day Workshops	90
5.4.4	Fall Workshop	91
5.4.5	Validation of the Approach	101
5.5	Designing for Dissemination	101
5.5.1	Learning Goals	102
5.5.2	Curriculum	104
5.5.3	Assessment Approach	112

5.5.4	Tools	113
5.5.5	Evaluation Strategy	115
5.5.6	Fluency Moments - an Analysis Methodology	117
5.5.7	Piloting the Design	118
5.5.8	Evaluation	119
5.6	Next Steps	122
5.6.1	Next Steps: Adapting Robot Diaries to Formal Educational Settings	122
5.6.2	Improving Disseminability	122
5.7	Summary	123
6	CSbots	125
6.1	Motivation and Program Goal	125
6.2	Timeline of the Project	126
6.2.1	My Role in the Project	127
6.3	Approach	127
6.3.1	Initial Evaluation	127
6.3.2	Iterative Design	127
6.3.3	Partnerships	128
6.3.4	Alignment	128
6.4	Ideation	128
6.5	Initial Evaluation	129
6.5.1	Textbook Survey	129
6.5.2	Survey of Educators	132
6.5.3	Partners' Prior Curricula	134
6.6	Initial Design	135
6.6.1	Learning Goals	135
6.6.2	Curriculum	136
6.6.3	Robot Platform	138
6.6.4	Software	139
6.7	Pilots and Evaluation	141
6.7.1	CCAC Summer 2007	142
6.7.2	Ohlone fall 2007	143
6.7.3	CCAC fall 2007	144
6.7.4	High School Pilots	154
6.8	Redesign	158
6.8.1	Design Constraints	159
6.8.2	Robot	159
6.8.3	Software	163
6.8.4	Curriculum	163

CONTENTS

6.9	Finch Pilots and Evaluation	166
6.9.1	CCAC Pilot	166
6.9.2	High School Pilots	178
6.10	Next Steps	179
6.10.1	Charter Schools	180
6.10.2	Additional Language Support	180
6.10.3	Commercialization	181
6.11	Summary	181
7	Summary, Analysis, and Conclusions	183
7.1	Summary of the Design Process	183
7.1.1	Ideation	185
7.1.2	Initial Evaluation	185
7.1.3	Constraint-Finding Process	186
7.1.4	Systems Alignment Cycles	190
7.1.5	Measuring Alignment	193
7.1.6	Dissemination	194
7.2	Similar Design Processes	196
7.2.1	Learner-centered Design	197
7.3	The Domain of Alignment-Centered Design	198
7.4	Alignment-Centered Design and Engineering Design	198
7.4.1	Ideation	200
7.4.2	Initial Evaluation	200
7.4.3	Prototyping	200
7.4.4	Testing	201
7.4.5	Analysis	202
7.4.6	Dissemination	202
7.5	Challenges and Limits of Alignment-Centered Design	202
7.5.1	Evaluation	202
7.5.2	Human Resources	204
7.5.3	Student Interests and Background	205
7.5.4	Constraints	206
7.5.5	Conventional Engineering Design: The CMUCam	206
7.6	Conclusions: A Choice of Process	208
A	Robot Diaries Curriculum	209
A.1	Dispositional Goals	309
B	Assignments for CCAC CIT-111 and CIT-130	311

CONTENTS

C Finch Documentation	341
Bibliography	364

CONTENTS

List of Figures

2.1	A Logo Floor Turtle (reprinted courtesy of Terrapin software)	24
2.2	A Robot made of Roblocks(reprinted courtesy of Modular Robotics)	25
2.3	The Basic STAMP (left) and Handy Board (right) (Handy Board photo courtesy Fred Martin)	28
2.4	A panoramic view of the Pittsburgh 2008 FIRST regional competition	30
2.5	The Vernier Labquest (photo courtesy Vernier LLC)	32
3.1	Aligning goals, instruction, and assessment	36
3.2	Adding a tool the alignment process	38
4.1	A schematic of a Braille cell (left) and the letter ‘t’ (center). The black circles represent embossed dots while the light grey circles represent un-embossed dots. A sample of Braille (right).	48
4.2	A Braille slate and stylus.	48
4.3	Timeline of the Braille Tutor project	49
4.4	The prototype E-slate taken for field testing at the Mathru School for the Blind.	57
4.5	Students at Mathru use the Braille Writing Tutor.	62
4.6	The teachers use the Braille Writing Tutor at the tutor station we set up in the computer lab.	63
4.7	Version 2 of the E-slate	66
5.1	Timeline of the Robot Diaries project	77
5.2	Robots made from craft materials	85
5.3	The Doodlechat interface	92
5.4	The chosen design and girls’ instantiations of the design	93
5.5	The Qwerk microcontroller	94
5.6	The RuR software program	95
5.7	The Express-O-Matic software program	96
5.8	The Roboticon Messenger software program	96

LIST OF FIGURES

5.9	The Hummingbird microcontroller	114
5.10	The Arts&Bots software environment	115
5.11	One girl’s final robot from the PALS workshop	119
6.1	Timeline of the CSbots project	126
6.2	Distribution of survey respondents	133
6.3	The first year robot platform	138
6.4	Initial skeleton file for software framework	140
6.5	Programs that make the robot say “hello world” (top) and print “Hello World” to screen (bottom)	141
6.6	Students in the CCAC robot lab	144
6.7	Overall retention rates in the fall semester CS1 course at CCAC	149
6.8	Week-by-week retention rates in the fall semester CS1 course at CCAC	149
6.9	Percent of students who listed the current assignment as their fa- vorite to date	150
6.10	Student estimates of their grade on each assignment	151
6.11	Student sources of frustration	152
6.12	Percent of students who linked assignment content to the external world	153
6.13	Mean grades of passing students, 2003-2007	153
6.14	Sketches of Finch shell concepts	160
6.15	Finch shells	161
6.16	The Finch robot’s sensors and actuators	162
6.17	Percent of students rating the current assignment as their favorite	171
6.18	Confidence of students in spring 2009 CIT-111	171
6.19	Confidence of students in fall 2009 CIT-111	172
6.20	Confidence of students in fall 2009 CIT-130	172
6.21	Sources of frustration for students in spring 2009 CIT-111	173
6.22	Sources of frustration for students in fall 2009 CIT-111	174
6.23	Sources of frustration for students in fall 2009 CIT-130	174
6.24	Retention of students in CIT-111 in pilot year compared to four prior years	176
6.25	Retention rates excluding students who dropped before exam 1	177
6.26	Grades of passing students in CIT-111 compared to four prior years	178
7.1	Alignment-centered design process steps	184
7.2	A simplified model of engineering design mapped to alignment- centered design	199

List of Tables

4.1	Summary of E-slate changes between version 1 and 2	67
5.1	Learning goals	105
5.2	Sequence of activities for first curricular module	107
5.3	Sequence of activities for part two	109
5.4	Sequence of activities for part three	111
6.1	Summary of key textbook attributes	130
6.2	Topic orderings. The topics are variables (V), simple I/O (I), flow control (F), arrays (D), exceptions and errors (E), and advanced I/O (A).	131
6.3	Schedule for a CS1 robotics course	137
6.4	Confidence and interest ratings for summer robotics assignments .	143
6.5	Listing of assignments used in CCAC pilot	145
6.6	Percentage of students who reported each category as their best-known programming language	147
6.7	Student reasons for taking course	148
6.8	Student ratings of interest in and confidence with computing . . .	148
6.9	Summary of teacher ratings of workshop	158
6.10	Schedule for CCAC's 2009 CIT-111 class	164
6.11	Schedule for CCAC's 2009 CIT-130 course	165
6.12	Students in the course and responding to pre/post surveys	167
6.13	Average ages of enrolled and passing students	168
6.14	Percent of students in CIT-111 reporting each category as their best-known programming language	168
6.15	Percent of students in CIT-111 reporting a given topic as their favorite subject in school	168
6.16	Percent of students in CIT-130 reporting a given topic as their favorite subject in school	169

LIST OF TABLES

6.17	Percent of students who prefer to program computers or robots . .	170
------	---	-----

Chapter 1

Introduction

By Tom Lauwers

This thesis is about a design process for creating *educationally relevant* tools. I submit that the key to creating tools that are educationally relevant is to focus on ensuring a high degree of alignment between the designed tool and the broader educational context into which the tool will be integrated. The thesis presents methods and processes for creating a tool that is both well aligned and relevant.

We begin by describing some key definitions and underlying assumptions of the thesis, as well as detailing the objective of the work and providing an organizational overview of the thesis. As the thesis is about educational tools and technologies, a good place to start the discussion is the definition of educational technology.

1.1 Definitions of Educational Technology

There are two competing definitions of educational technology in society. One focuses on the tools and media used in education: blackboards, textbooks, documentaries, computer programs, Internet social media, and so on; this definition is commonly used by technologists and media creators. The other, used by historians and social scientists, considers the definition of the word 'technology' more deeply; technology, under this second definition, is not a product, but a kernel of knowledge that allows the knowledge holder to act in a practical and systemic way. Under this definition, educational technology has a long and rich history; the introduction of the first professional teachers in the fifth century BC in Athens, the creation of the first universities and the development of the scholastic method by Pierre Abelard and St. Thomas Aquinas, and the development of the sequential curriculum accompanied by a textbook are all developments of educational tech-

1. Introduction

nology (Saettler, 1990). Educational technology developments occurred in the 20th and 21st centuries as well: The creation of Bloom's taxonomy to organize the cognitive (Bloom, 1956) and affective goals (Krathwohl et al., 1973) of a course of learning. The development of cognitive models to aid the design of math curricula and computer-based tutoring programs (Rittle-Johnson and Koedinger, 2001). All of these examples share a commonality: *None of them required the development of new tools or media.* Considering technology in this way, one can view educational technologies not as mere tools that exist on the periphery of educators' and students' experiences, but as the core systems that educators use to organize the learning of their pupils.

Ironically, technologists, the author included, have an inherent affinity for making the next cool new tool, and thus rarely consider a more expansive view of educational technology. This perspective leads to a lack of understanding that in order for new tools to be adopted, those tools need to be part of a broader instructional and technological context. Despite the overly optimistic views of their creators and supporters, tools are rarely adopted to the extent that seems possible to technologists when a tool is created. In 1913, one of the world's best technologists, Thomas Edison, made the following prediction:

“Books will soon be obsolete in the schools. Scholars will soon be instructed through the eye. It is possible to teach every branch of human knowledge with the motion picture. Our school system will be completely changed in ten years.”

- Thomas Edison, quoted in the New York Dramatic Mirror on July 9, 1913 (Saettler, 1990)

This woefully incorrect prediction illustrates that historically new educational tools have come into use in schools at a much slower rate than technologists expect. There are systemic reasons for this, including the time required to train teachers in the use of the new tool, a conservative attitude to change that is inherent to most large institutions, and in many cases, a lack of funding to support the new tool. However, these reasons typically only apply once the tool has already cleared its first major hurdle: educational relevance. The reason behind Edison's and others' wild optimism is that they conflate educational relevance and educational potential. Educational relevance requires that the tool has become sufficiently refined for educators to use, and crucially, to create educational content with the tool; in essence, it means that the primary advocates of the tool are no longer technologists, but content creators. To go back to the audiovisual movement of the early 20th century exemplified by Edison's quote, movie making may have been technically possible in 1913 by well-trained experts, but it would still be several decades

before non-technologists were making educational documentaries and curriculum designers understood movies well enough to properly align and integrate the new tool with existing curricula (Saettler, 1990).

The process of creating educational relevance can be long and drawn out, but there are methods for shortening it. In emerging technological domains that contain educational potential, partnerships with educators and content creators at the very beginning of the design process may lead to educational relevance more quickly. In the next section I describe one emerging technological domain that is ready to have its educational potential accelerated into educational relevance.

1.2 Configurable Embodied Interfaces

This thesis is concerned with the process for designing educational tools within a technological domain I refer to as “Configurable Embodied Interfaces”. Configurable embodied interfaces have a number of key features, they:

- Can sense their local surroundings through the detection of such environmental and physical parameters as light, sound, imagery, device acceleration, etc.
- Act on their local environment by outputting sound, light, imagery, motion of the device, etc.
- Are configurable in such a way as to link these inputs and outputs in a nearly unlimited number of ways.
- Contain active ways for users to either directly create new programs linking input and output, or to easily re-configure them by running different programs on them.
- Are user-focused; they assume that a human being is manipulating them in some way, through affecting input and observing output of the interface.

Configurable embodied interfaces contain within their definition the classical ‘sense-think-act’ paradigm of robotics, and some robots certainly are embodied interfaces. However, many robots are not user-focused and/or are not designed for users to change their programs; an automated arm at a car factory (neither user focused nor user-programmable), robotic receptionists and museum guides (not user-programmable), and a tele-operated bomb squad robot (not user-programmable) are all examples of robots that do not fall within the embodied interface domain.

1. Introduction

On the other hand, the LEGO Mindstorms robotics construction kit and associated software (Lego, 2009), the discontinued Sony Aibo robot dog, and Roblocks (Schweikardt and Gross, 2008), a kit of intelligent blocks that alter their behavior based on the overall block configuration, are all examples of robots that are also embodied interfaces.

Due to their emphasis on the user, embodied interfaces are also well represented in the consumer electronics sector. Many new smart phones are embodied interfaces: these phones emphasize multimodal sensing and actuation, user configurability through the addition of new 'apps' or programs, and certainly assume a human being is interfacing with them. Though ten years ago few computers could be categorized as embodied interfaces, newer computers, especially the netbook strain that is oriented towards communication, contain webcams, microphones, and even accelerometers. Many consumer electronics remain outside the realm of the embodied interface; mp3 players, ordinary 'dumb' cell phones, and digital cameras are not easily user modifiable.

Though not all embodied interfaces are marketed to educational activity, their user focus, user configurability, and multimodal interactivity ideally suit them to education. Learning occurs through interactions, be it a teacher lecturing to a class, a student reading a textbook, or a user playing an educational computer game. The interactions enabled by embodied interfaces are qualitatively different from those available through other methods; notably, an embodied interface can impart concepts by demonstrating them in the physical world while also interfacing with a computer to perform experimental analysis.

As they become increasingly ubiquitous and powerful, embodied interfaces are growing increasingly relevant in the world of education. While some educators have adopted embodied interfaces for use in their classrooms, the design of embodied interfaces still proceeds largely without the input of educators. This input is crucial in determining all aspects of the interface design: which sensing and actuation modalities are necessary, how the tool should appear, and how it should be programmed or configured by the user. As with other promising technologies for education in the past, it is necessary to involve educators in design before embodied interfaces can make their way into the classroom in a systemic way.

1.3 Thesis Statement and Contributions

The design features of a given embodied interface relate directly to the types of concepts that are appropriately demonstrated with the interface - imagine attempting to teach about acceleration with an interface that is not capable of either moving or registering motion. It is the learning goals and student interests, then, that

should determine the features and interaction modalities of the embodied interface. Unfortunately, technologists may not be wholly aware of these goals when creating a new embodied interface, while those who are aware of these crucial design considerations, namely teachers and students, may not be aware of the potential set of sensors and output mechanisms, and may not get much input until after the interface has been designed and is ready for testing. Embodied interfaces can be powerful educational tools, but as with previous technologies, it is necessary to involve educators and students in the design process before their full potential is realized.

1.3.1 Statement

This thesis is concerned with altering the design process dynamic by answering a question of 'how'; specifically, *how does one design an embodied interface so that it is aligned with the goals of the end users of the tool?* This question leads to a number of others: How do technologists discover the goals of the end users? How do these goals translate into design constraints? What are the methods used in order to achieve alignment? How does one determine when an embodied interface is properly aligned?

1.3.2 Contributions

To answer these questions this work details three worked examples of collaborations between technologists and educators that led to the creation of educationally successful tools. These three examples share a focus on creating an embodied interface to tackle a specific cognitive and affective set of learning goals, but differ completely in the location of the learning environment, the age and interests of the learners, and the nature of the learning goals. Through the exploration of the methods used, an analysis of the general and context-specific features of the design processes of the three accounts, and a comparison of the process used in this thesis to a conventional engineering design process, this work provides case studies and a generalizable design process that can inform technologists interested in designing educationally relevant configurable embodied interfaces.

1.4 Organization of the Work

Chapter Two discusses related work in embodied interfaces, tracing the history of the domain to date and pointing to current commercial and educational successes within the field. Chapter Three details the principal methods used in the main body of the work, specifically, alignment and instructional design and their extension to

1. Introduction

the design processes of the embodied interfaces covered in the work, design-based research, and participatory design. Chapters Four through Six are case studies of the design processes of embodied interfaces used in education. All three of these case studies were interdisciplinary efforts involving teams of researchers; as such, when writing about these efforts I use the first person plural to indicate the collective nature of the efforts. In all three cases the projects are on-going and so it is important to detail the period of time of the analysis resulting in this thesis: the design processes that I describe occurred in a period from 2006 to 2009. Each chapter starts with a brief timeline of events for the project, and each chapter has a section discussing the current direction of the project. Chapter Four covers the development of the Adaptive Braille Writing Tutor, a project to aid in the teaching of Braille writing in developing countries. Chapter Five discusses Robot Diaries, a program to create an embodied interface and associated curriculum that motivate middle school girls to pursue further study in Science, Technology, Engineering, and Mathematics (STEM) fields. Chapter Six presents CSbots, a program to strengthen retention and achievement among students in introductory computer science education through the integration of a custom-developed embodied interface. Chapter Seven presents alignment-centered design, a model of the design processes used in chapters Four to Six, abstractly describes the steps in the model and ends by discussing similar design processes and how the process relates to a model of engineering design. A summary reading of this thesis can be achieved by reading this chapter, chapter Three, and chapter Seven.

Chapter 2

Configurable Embodied Interfaces in Education

Configurable Embodied Interfaces have a number of characteristics that make them well suited to education, especially in Science, Technology, Engineering, and Mathematics (STEM) fields. These characteristics include configurability or programmability, the ability to sense, characterize, display, and act on local objects and ambient environmental cues, and the assumption that a human being is manipulating and using the interface in some way. Spurred by the growth of cheap computation and sensing, a large number of educational programs and devices have been created in the last three decades. Programs using these tools exist in both formal and informal educational settings and are in use from early childhood through adult and community education. Typically, configurable embodied interfaces are used as tools in three major and sometimes overlapping areas: computer Science education, creative and engineering design education, and traditional science and math education. A history of these tools and their uses is presented below, divided for clarity into these three areas, though in many cases the tools are used to support learning goals that encompass and occasionally transcend STEM education.

2.1 Computer Science Education

With an emphasis on configurability and programmability, Configurable Embodied Interfaces can be very useful tools in the sphere of computer science education. In fact, what may be the first Configurable Embodied Interface, the system comprised of the Logo Turtle and associated programming environment (Papert, 1980), was targeted precisely at helping children learn to program. Using Logo, children could program a small floor robot, the 'turtle' (Figure 2.1), to move a specified distance,

2. Configurable Embodied Interfaces in Education

turn, beep, and draw shapes on paper using a small pen placed under the turtle's body. Following in these footsteps, configurable embodied interfaces have been created to support computer science education across all age ranges. I detail some of these efforts, starting with manual programming wherein manipulations of a physical object generate programs, and continuing through programs written using visual/iconic and textual languages.



Figure 2.1: A Logo Floor Turtle (reprinted courtesy of Terrapin software)

2.1.1 Manual Programming

Curlybot (Frei et al., 2000) is an educational tool to allow young children to develop notions of motion, computation, and geometry. The tool is composed of a simple two-wheeled robot which moves on a flat surface; the robot can be manually moved by a child and then plays back those motions. As curlybot mirrors not just position but also acceleration and velocity, the tool allows children to explore in a wide variety of ways with a very simple tool. Children as young as four can create geometric patterns, can develop stories where the curlybot appears to respond at the right moment, and can create routines that use motion to express intent and emotion in these stories. Topobo (Raffle et al., 2004) follows the thread of manual programming and lifts it off the two-dimensional surface using an architecture of custom-designed snap-together parts that allow children to construct an unlimited variety of shapes. With Topobo, once a shape has been snapped together children can manipulate the shape and then press a button to have the shape play back those manipulations; if, for example, the child created a four-legged creature and then manipulated the legs to make it walk, Topobo would remember those manipula-

tions and walk on its own during playback.



Figure 2.2: A Robot made of Robloks(reprinted courtesy of Modular Robotics)

Robloks (Schweikardt and Gross, 2008) allows children as young as six to construct robots using a kit of programmable 40mm cubes. The cubes consist of three main varieties: Sensor cubes, capable of sensing light, sound, or distance to an object, actuator cubes, capable of rotating attached blocks, translating over a surface with tank treads, making noise or glowing, and finally blocks that perform mathematical and logical operations on values coming in from other blocks. When manually programming with this kit, the very construction of the robot affects the way in which it operates - snap together a light sensing block with a tank tread block, and the resulting robot will speed up when the light gets brighter. Snap a light sensor to a logical compare block and a tank tread, and now the robot will turn on full speed if the light sensor value reaches a threshold but will stop completely if the light reading goes below the threshold. In effect, using this programming methodology, the robot's topology and morphology become the program. In addition to a manual programming interface, Robloks also enables users to program their creations using visual and textual interfaces (Schweikardt and Gross, 2006). These additional levels of interface require the use of a computer and a special block, called the 'Comm' block that interfaces between the robot and the computer. The comm block transmits data to the computer regarding the topology of the robot and which blocks are used in the robot. Using the visual programming interfaces, students can modify the behavior of some of the blocks (causing, for ex-

2. Configurable Embodied Interfaces in Education

ample, a tank tread block to reverse direction) and drag a visual representation of the robot to actuate the blocks. At the deeper textual level, students can change the 'firmware' or basic software running on each individual block, changing the very nature of the blocks (for example, an operator block, instead of being a simple multiply, could become a complex equation). With three levels of programming interface, Roblocks has the ability to interest students from first grade through college.

2.1.2 Visual and Textual Programming

Manual programming works well to introduce young children to programming and computer science concepts, but as students become capable of abstract thought, other ways of programming become more interesting and useful. LEGO Mindstorms (Lego, 2009) is likely the most popular kit of parts to construct a configurable embodied interface. The kit consists of a large number of regular LEGO pieces, as well as customized motor and sensor blocks. At its core sits a programmable brick with ports where sensors and motors can be attached. The kit ships with a visual programming language that allows programming through a drag and drop interface. The LEGO Mindstorms language is based on the programmable bricks project (Resnick et al., 1996; Martin, 1988) at MIT's media lab, which was itself a refinement of the LEGO/LOGO system (Resnick and Ocko, 1991). In addition to the LEGO-provided language, which is appropriate for 6-12th graders, an extensive community of hobbyists and educators have created support to program LEGO Mindstorms with a number of textual languages including Java, C/C++, .NET, Python, and Processing. With added language support, the LEGO kit becomes useful in introductory (Fagin and Merkle, 2003) and intermediate (McNally, 2006) college courses as well.

Though the LEGO kit is heavily used, a number of other configurable embodied interfaces exist that use textual programming. At Carnegie Mellon, a student-taught course (Shamlan et al., 2006) in which programming robot behaviors and creating robotic art are the main activities attracts nearly half its students from the humanities, business, and art schools. In this course, robots are programmed in a subset of the C programming language, and assignments including light tracking, robot dances, and solving mazes.

A high school summer course (Nourbakhsh et al., 2003) in which students built a robot from a kit and then learned to program it to exhibit progressively more complex behaviors found increased student motivation to pursue further study in science and technology; girls in the program did especially well, catching up to boys on measures like confidence in science and technology.

A major study of the use of pre-constructed LEGO mindstorms robots as ed-

educational tools in introductory computer science education at the US Air Force Academy (Fagin and Merkle, 2003) found that students using the robots did not show improved learning or retention in computer science. The authors suggested two reasons for this failure:

- LEGO Mindstorms robots are typically too expensive for student ownership, and so students must work on robot programming assignments in labs with limited hours.
- Debugging is complicated by the fact that it takes several minutes to test a program as one needs to observe the behavior of the robot and ensure it matches the desired behavior.

Taking the lessons of this study to heart, two recent initiatives are aiming to develop a robot specifically tooled for computer science education. The Institute for Personal Robots in Education has developed a Python-programmable robot that employs a wireless bluetooth tether to allow programs to run primarily on the computer (Blank et al., 2007). The CSbots project, detailed in Chapter Six, has developed a USB-tethered robot that is programmed in Java with programs that run on the computer. In both initiatives, students borrow or purchase the robot for a semester and so are capable of working on their assignments at any time.

2.2 Creative and Engineering Design

With a focus on sensing and acting upon the local environment, configurable embodied interfaces lend themselves easily to design projects. Although there is some overlap with configurable embodied interfaces to support computer science instruction, interfaces that support learning design tend to have an important additional characteristic: the choice of sensors and actuators that are part of the configurable embodied interface is open-ended. Generally the types of configurable embodied interfaces that work well in the design world have at their core a controller capable of interfacing with a number of different sensors and actuators; the controller provides inputs and outputs, but a large part of the design exercise is designing and constructing the embodied interface, both physically and programmatically. I present a number of these controllers, as well as programs that have been developed using these controllers.

2.2.1 Controllers

The BASIC Stamp (Parallax, 2009) and the Handy Board (Martin, 2000) are two of the earliest controllers to have been released in this space, in 1992 and 1993

2. Configurable Embodied Interfaces in Education

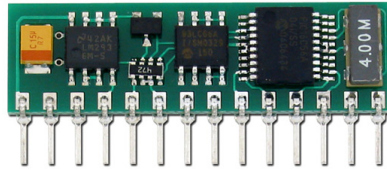


Figure 2.3: The Basic STAMP (left) and Handy Board (right) (Handy Board photo courtesy Fred Martin)

respectively. These two controllers are archetypes of all subsequent controller designs, representing two extremes along an axis of modularity. The BASIC Stamp was designed to be extremely small and portable - its form factor was designed so that it could be plugged into a breadboard or circuit board. This pluggability was a strength for those capable of designing additional support electronics around the Stamp, but a weakness for those who wanted to begin immediately to create and program new devices. The Handy Board integrated everything necessary to serve as a core controller; it even included a rechargeable battery. A fair amount of care was taken to ensure that the connectors to sensors, motors, and servos were well defined and logically placed such that anyone could create a device without creating support circuitry. Despite major differences, both of these controllers included a custom programming interface designed to be easy to learn and use by students.

Though a number of controllers have entered the space since the early 90s, they all fit on an axis between extreme modularity (as in the BASIC Stamp) or extreme integration (as in the Handy Board); and all the successful ones are programmable using easy to learn languages. Modern examples of controllers include the Brainstem (Acroname, 2009), which fits neatly between the Stamp and Handy Board on the modularity-integration axis, the Qwerk (Nourbakhsh et al., 2007), which, with on-board Wireless, USB host capability, and the ability to drive four motors, 8 digital I/O, and 16 servos, seems descended completely from the Handy Board, and the Arduino (Banzi, 2008), the hardware for which is open source, thus allowing anyone to create custom circuit boards including an Arduino unit and representing

the modern-day extreme in modularity.

Modern-day controllers go beyond those specifically designed for design and engineering. Palm devices, Game Boys, and iPhones have all been used as central controllers for configurable embodied interfaces. The Palm pilot robot kit (Reshko et al., 2002) is an omnidirectional robot that was controlled using a Palm Pilot. The XBC (LeGrand et al., 2005) is a module that plugs into the Game Boy Advance, providing the Game Boy with a number of ports to read sensors and control motors. Finally, hobbyists have used iPhones to control humanoid and wheeled robots, and it would not be surprising to see a commercial offering towards this end in the near future.

2.2.2 Programs

Programs that use configurable embodied interfaces as a launching pad for teaching engineering and creative design are numerous and consist of a wide variety of approaches and target audiences; these can roughly be divided into competition- and arts-based programs.

Competitions

The best-known application of embodied interfaces to support design skills is the robotics competition. These competitions have grown explosively at middle and high schools over the last decade, with current participation in the largest programs reaching 75,000 students. Although competitions vary in a number of ways, most require students to use a standard controller and a kit of parts, so as to place teams on the same footing regardless of financial strength. Although it is beyond the scope of this review to describe every robot-based tournament, it is worthwhile to consider the two largest such competitions: FIRST (FIRST, 2009a; Yim et al., 2000; Melchior et al., 2005) and Botball (Botball, 2009; Miller and Stein, 2000). The FIRST organization hosts three competitions, a high school league in which students receive a kit of parts and controller developed by Innovation First (FIRST, 2009b), the FIRST LEGO League in which elementary and middle school students build robots using the LEGO Mindstorms kit, and the FIRST Vex Challenge which uses a robotic assembly kit developed by Innovation First. Botball uses an advanced controller and LEGOs for construction material. In all cases, the competition rules are revised annually, and students are given no more than two months from the announcement to design, build, and test their robots. Tournaments occur on a local and national scale, and tend to mimic sporting events complete with mascots, cheering fans, and intense competition between teams.

2. Configurable Embodied Interfaces in Education



Figure 2.4: A panoramic view of the Pittsburgh 2008 FIRST regional competition

Art and Storytelling

Recently, recognition that robot competitions do not appeal equally to all students has led to the development of a second approach to attracting students to STEM through configurable embodied interfaces. Rusk et al. (2008) list four key characteristics of this approach:

1. Focus on themes, not challenges. Themes should be broad enough to allow students to connect their personal interests to the design, while narrow enough to foster the feeling of a shared experience.
2. Combine art and engineering. Most students have had experience building for art and are familiar with craft materials, by incorporating these the technology becomes an addition to a process with which the students are comfortable.
3. Encourage storytelling. Children's play modes can be divided into two types: patterners and dramatists. Patterners are already attracted to robot competitions; so to capture a new audience it is important to appeal to dramatists.
4. Organize exhibitions. The public display of end-products is an appealing feature of competitions, and should be retained by organizing public exhibitions.

Several recent programs show how these characteristics can be implemented. In Artbotics (Kim et al., 2007) high school students work together with undergraduates and are tasked with creating interactive art exhibits; the exhibits are publicly displayed at a local museum. Another such program, Robot 250 (Fitzpatrick,

2008), was developed for Pittsburgh's 250th anniversary celebration; community workshops were held at a number of museums in the city, and in these workshops adults and children created robotic and kinetic sculptures from crafts materials that responded to light, temperature, sound, or air pollution.

Marrying electronics and textiles, the Lilypad Arduino (Buechley and Eisenberg, 2007) allows teenagers to create programmable, wearable electronics using conductive thread and flexible electronics.

Focusing on elementary school age children, PETS (personal electronic teller of stories) (Druin, 1999) supports creativity and design skills through a combination of storytelling and construction. Using PETS, children snap together a robot and then have the robot act out stories written by the child using the My PETS software package. The PETS robot will emotionally express through physical motion whenever it reaches an emotional keyword (happy, sad, angry, etc.) in a story.

Engineering and Design Education

Configurable embodied interfaces are routinely used in formal engineering education at the university level. The FIRST competition's roots trace back to two courses at MIT; the first of these, 2.70 has been hosted in the mechanical engineering department since 1970. In 2.70, students spent the entire semester developing mechanical contraptions to compete against each other in an end-of-year challenge. The second of these, 6.270 (Martin, 1994), was developed partially to provide an analogous course to electrical engineering and computer science, and featured similar challenges appropriate to the backgrounds of those students.

2.3 Science Education

Though most configurable embodied interfaces are used in engineering and computer science education, a few are used primarily as tools in science and math instruction. Naturally, there is significant overlap between these skills - a good engineering design program will include necessary concepts of physics like torque and circuit fundamentals, but these concepts are learned incidentally. This section details a few configurable embodied interfaces where the primary purpose is learning science concepts.

Supporting scientific discovery and deduction skills, the SENSE project (Tallyn et al., 2004) focuses on introducing middle school-age students to environment sensing applications. Students use pollution sensors connected to mobile computers to capture the data, and then create computer visualizations of the data.

Neighborhood Networks (DiSalvo, 2007) is a program open to both adults and

2. Configurable Embodied Interfaces in Education

children that encourages scientific exploration of environmental characteristics like temperature, humidity, and air pollution. The program uses a device, the Canary, that contains several built-in sensors as well as the capability to actuate servos. In addition to scientific exploration, neighborhood networks includes many of the arts and creative design activities previously mentioned; participants are encouraged to use the Canary to both discover where environmental characteristics like sound and air pollution are poor in their neighborhood, and to use the Canary's ability to actuate servos to create kinetic sculptures that represent these intangibles and advocate for their reduction.



Figure 2.5: The Vernier Labquest (photo courtesy Vernier LLC)

Commercially, the Vernier Labquest (Figure 2.5) (Vernier, 2009) is a configurable embodied interface that records, graphs, and analyzes data from up to six sensor inputs. Accompanying the Labquest tool, Vernier has created upwards of 100 sensors that are capable of monitoring everything from temperature and light to dissolved oxygen, EKGs, and radiation. The company has produced curricula aligned to state standards associated with sensor packages for all major science fields.

2.4 Summary

Configurable embodied interfaces have shown themselves to be of use in a number of educational contexts to date. It is my opinion that as the underlying hardware technologies (sensing, actuation, and processing) continue to improve, we will see increasing use of these tools, both towards applications like science and engineering education where they have been used traditionally, and in new application areas like math or the social sciences.

2. Configurable Embodied Interfaces in Education

Chapter 3

Methods for the Design of Configurable Embodied Interfaces

As identified in chapter one, a principal reason for the failure of many educational tools is due to designers not accounting for the larger educational context. This failure boils down to a single issue: a mis-alignment between the capabilities and created content of the tool and the needs and goals of educators. The central goal of the design process of educational tools, then, should be the ever closer alignment of the capabilities of the tool to the learning goals, instructional methods, and assessment techniques of the curriculum for which the tool is designed. This chapter presents methods used in the works presented in chapters 4, 5, and 6 to create well-aligned educational tools. I discuss the origins of the concept of alignment in instructional design and the constraints alignment imposes on the design of configurable embodied interfaces for education, as well as participatory design and design-based research, two methods for discovering these constraints and creating well-aligned designs.

3.1 Alignment in Instructional Design

Alignment is a well-known curriculum design principle; essentially it advises the course designer to align the learning goals, instruction, and assessment, so that each supports the other (Figure 3.1). An example of a poorly aligned class is one in which a teacher has given an exam that assesses materials or concepts that were not covered by prior lectures or exercises. Although in some cases classes may be intuitively aligned by their designers, aligning a course is a process that can

3. Methods for the Design of Configurable Embodied Interfaces

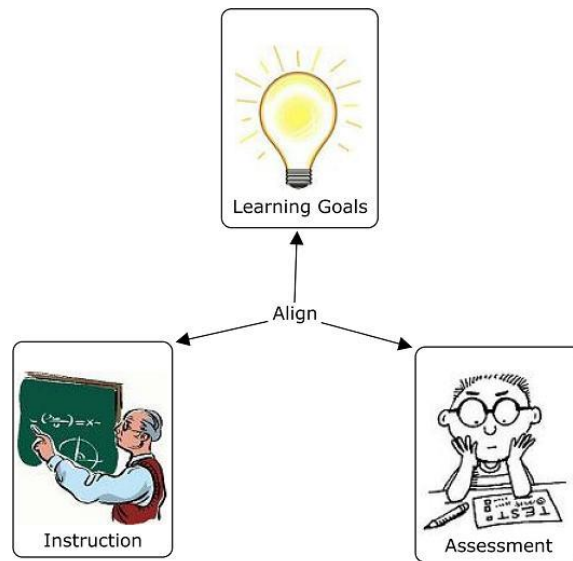


Figure 3.1: Aligning goals, instruction, and assessment

be performed in a premeditated, explicit way (Wiggins and McTighe, 2005). This alignment is often done through a process known as *backwards design*, which begins by detailing the learning goals for the students, as these goals will drive the required types of assessment and instruction. Once learning goals are determined, assessments are devised that can measure those outcomes. Finally, instructional tactics, exercises, and activities are devised to allow students to meet the desired outcomes. Although Figure 3.1 doesn't explicitly represent it, student interests and background are an important part of this model; they are considered when formulating appropriate learning goals and especially when creating instructional activities that will be compelling. At each step, it is necessary to reconsider the entirety of the design - for example, when the instruction is designed the assessment and goals are reconsidered to ensure that there will be no misalignment between the three major categories.

The notion of aligning the objectives of a learning activity with the assessment of that learning and the instructional techniques has likely been around as a common-sense principle for as long as formal education has existed. Modern-day alignment in instructional design was popularized in part by Alan Cohen, who called it a 'magic bullet' (Cohen, 1987). He presented four studies (Koczor, 1984; Tallarico, 1984; Fahey, 1986; Elia, 1986) in which instruction and assessment were deeply aligned, and noted that in all of these studies educationally significant effect sizes of one to three standard deviations were found. Each study also tested degree

of alignment by variously aligning or misaligning assessment and instruction, and found that alignment benefited low-achieving students by a greater amount than high-achievers, especially on more difficult portions of the assessment.

While the studies illustrated by Cohen had few subjects and involved instructional periods of less than two hours, recent studies of alignment have indicated similar effects over much larger groups and longer time spans. A study of instructional alignment to the IOWA basic skills test in mathematics involving over 4000 third grade students found that one year after the curriculum was aligned to the test there were statistically significant improvements in scores across all income, gender, and ethnic lines (Mitchell, 1999).

A criticism of alignment, especially when applied to improving standardized test scores, is that it simply proves that teaching to the test improves test scores. These criticisms are generally misplaced; alignment of instruction to tests which do not support stated learning goals would result in an overall misaligned design. If standardized tests are the appropriate measure of certain learning goals, then it follows that teaching to those tests will cause the largest number of students to attain the learning goals specified. A caveat does exist if teachers know precisely which items are on the test, and have their students learn those items by rote. In such a case, students may pass an assessment having learned none of the underlying learning goals.

3.2 Extending Alignment

Alignment through backwards design is a method that can be extended to the design of educational technology. The features of an interface provide a fourth design node, feeding back on and aligning with the three traditional nodes (Figure 3.2). Though it is tempting to classify new educational tools as part of instruction, an embodied interface can enable new assessments, and even new learning goals; similarly, learning goals, instructional methods, and assessments all affect the interface's interactivity and design. In Chapter Four, I discuss the Braille Tutor, which modified both the instruction and assessment portions of the curriculum into which it was introduced. The Robot Diaries program, introduced in Chapter Five, centered an entire new activity around a new tool - in some ways, the learning goals themselves were defined by the capabilities of the tool. In Chapter Six, I discuss CSbots, an example in which the curriculum was modified as little as possible and the tool was designed in a highly constrained way to ensure that goals, assessment, and some instruction could remain the same.

Considering the entire system leads to a design process that is qualitatively different from the engineering design process aimed at creating new tools. All design

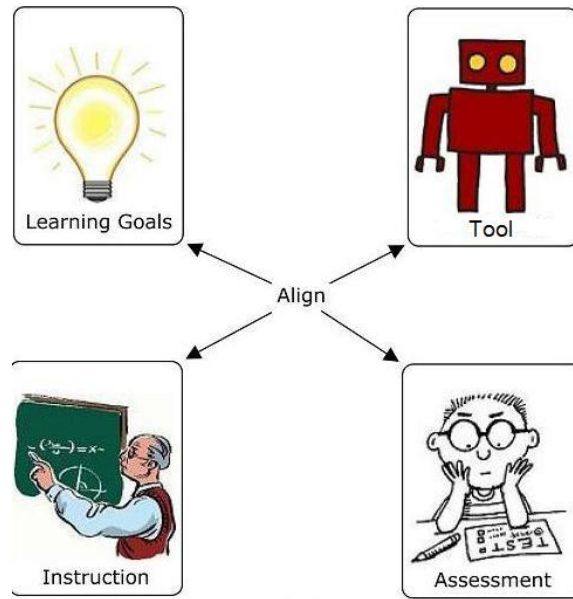


Figure 3.2: Adding a tool the alignment process

is characterized by constraints and trade-offs, and the difference between a traditional engineering design process and one that is guided by alignment is a difference of constraints. For example, there are many ways in which one might design a configurable embodied interface with the goal in mind of using it in computer science education; however, if the designer is unaware of the constraints imposed by the goals, assessments, and instruction used in the course, he might make crucial mistakes. The question for the designer, then, is how to discover those constraints. The processes described in Chapters Four to Six include two major methods for constraint discovery - participatory design and design-based research.

3.3 Participatory Design

Participatory Design (Schuler and Namioka, 1993; Cornwall and Jewkes, 1995) is a method to replace the mutual incomprehension between designers and end-users with mutual knowledge. The notion of involving end-users in product design came first from Scandinavian studies of democratizing workplace products (Bjerknes et al., 1987), but it has grown vastly in the past two decades and is now commonly used in numerous different contexts. The key theme of participatory design research is involvement and collaboration with end-users from the beginning of a

design process; in a sense, the research team needs to recruit these collaborators into a design co-op, in which the trajectory of the design is controlled by the entire group, not just by the designers. It is important in participatory design to maintain a democratic atmosphere, which can occasionally lead to design decisions made by the group with which the designer does not fully agree.

The benefits of using participatory design methods to design configurable embodied interfaces in educational settings are many. Configurable embodied interfaces are a relatively new class of tools whose capabilities and feature space are mostly unexplored and not well known to educators. At the same time, the complexity of educational settings and educational technology imposes constraints on any educationally relevant design that are unknown to the designer. By forming participatory design co-ops, the designer receives immediate and constant feedback from a sample of the very same people who will eventually be using the interface and gains a greater understanding of their goals, while the involved end-users feel a sense of authorship over a design that directly applies to them. Using these methods, it may be possible to speed the process of turning an idea with educational potential into an educationally relevant tool, as design constraints that may have gone unnoticed by the designer are incorporated earlier. It is also possible that adoption of the tool will be accelerated, as educators who collaborated on the design can advocate for it using the language of their field.

It is important to note that while participatory design methods always involve researchers collaborating with end-users, the depth of this relationship varies significantly and thus the definition of participatory design in the research literature is not stable. Biggs (1989), working on a synthesis of farmers' participation in research in nine national programs, declares an observed continuum of participation:

- **Contractual.** People are contracted into the projects of researchers to take part in their inquiries or experiments.
- **Consultative.** People are asked for their opinions and consulted by researchers before interventions are made.
- **Collaborative.** Researchers and local people work together on projects designed, initiated and managed by researchers.
- **Collegiate.** Researchers and local people work together as colleagues with different skills to offer, in a process of mutual learning where local people have control over the process.

The projects described in Chapters Four to Six all fall roughly in the collaborative category of this continuum; each was initiated and managed by a research

3. Methods for the Design of Configurable Embodied Interfaces

group, but end users were involved from the beginning of the design process and guided the design trajectory in significant ways.

In one case particularly appropriate to the research documented in Chapter Five, (Druin, 1999) extends participatory design principles to enable children to become effective designers and researchers in the process. She calls this extension “cooperative inquiry”, and notes three major characteristics of the framework:

1. It is a multidisciplinary partnership with children.
2. It involves field research that emphasizes understanding context, activities, and artifacts.
3. It involves iterative low-tech and high-tech prototyping.

The third characteristic appears to be very important for involving children. By prototyping with low-tech materials that require no prior experience to use (e.g., clay, paper, craft materials), children and adults are placed on the same experiential footing. Druin also mentions three major techniques that comprise cooperative inquiry. During contextual inquiry both adults and children observe, take notes, and interact with child users. This stage is mostly an observational and data-gathering stage. In the participatory design stage, children and adults mock up and prototype with low-tech materials. Finally, in the technology immersion phase, children are immersed in high technology to provide them with as much access to technology as they desire; this stage aims to discover children’s patterns of use with the technology, which can inform the high-technology designs of the low-technology prototypes developed during participatory design. These techniques tend to flow in chronological order, but Druin emphasizes that the nature of the process is such that phases of work are not necessarily distinct and that techniques can be interspersed with one another

In another study appropriate to Chapter Four, Brand and Schwittay (2006) argue for the use of participatory design techniques in the development of programs for the developing world. They analyze the results of the LINCOS project, a project to provide rural villages with Internet-connected kiosks, and argue that the reason for the project’s eventual failure was due to an inattention to the needs and concerns of the local people for whom the technology was ostensibly created.

3.4 Design-Based Research

A common critique of traditional laboratory research in the learning sciences and psychology is that the results are not usable in educational practice; due to the huge number of uncontrollable variables, a laboratory study of learning does not

easily translate to a classroom intervention that uses the study results to improve learning. By tightly coupling research to educational context, Design-Based Research aims to respond to this critique by providing researchers with the flexibility to create contextually appropriate educational designs. The seminal papers in the field (Brown, 1992; Collins, 1992) make the case for moving research into education from the laboratory to the classroom, and introduce the concept of the design experiment, an experiment in which the engineering of an innovative educational environment is conducted contemporaneously with studies of those innovations. Hoadley (2004) excellently contrasts design-based research and more traditional forms of experimentation, pointing to four major differences between the two:

1. In design-based research, there is a strong blurring between the researcher and the participant, as the researchers participate deeply in the experiment, and the participants influence the experiment and possibly even the research questions.
2. Results are shared with the community *without* the expectation that the research is universally applicable.
3. Planned comparisons occur, but researchers follow revelations where ever they may lead, tweaking intervention and measurement as necessary.
4. Enacted interventions are the outcome of the research, and documentation of the design, rationale for it, and how understanding changed over time are vital academic contributions to the educational literature.

Despite characteristics that violate tenets of traditional research design, Hoadley argues that in some ways the results produced by design-based research are more rigorous; specifically, they are more focused on context and better at connecting interventions to outcomes through mechanisms. Hoadley also addresses the idea of alignment between theory, treatment, and measurement (somewhat analogous to learning goals, instruction, and assessment, respectively), claiming that the iterative nature of design-based research leads to increasingly better alignment.

A potential threat to validity and rigor is the impact of the surrounding context versus that of the educational innovation. Tabak (2004) explicitly defines these as the endogenous and exogenous elements of the study. The endogenous consists of all aspects of the activity that were not added or modified by the innovation, while the exogenous consists of all that is specified by the design. There is a bias among researchers to overly credit the exogenous aspects of the design. As innovations in design-based research are typically underspecified, the details are filled in with endogenous aspects, and so should be studied and documented as robustly as the exogenous aspects.

3. Methods for the Design of Configurable Embodied Interfaces

Although design-based research involves the study of educational innovations, it does not encompass studies of the effectiveness of educational programs that were developed without theoretical backing. Sandoval (2004) lays out a path to strengthen the theory-based aspect of design-based research by introducing the notion of the embodied conjecture. An embodied conjecture is:

- derived from knowledge of learning in particular domains.
- specific enough to be rejected or refined empirically, unlike 'design principles'.
- such that the refinement of the conjecture can improve not just a learning environment, but cause refinements in learning theory itself.
- multiply embodied in the aspects of the design; it's in tools, materials, assignments, etc. in ways that embody its hypothesized role in supporting learning. This implies that it is not possible to imply causality for one part of the design.

It is the last of these characteristics that is difficult to do in practice. The conjecture is "embodied" because it is built into every aspect of the design experiment with the end goal being that success or failure of the experiment allows for the researcher to reject or accept the conjecture. Since the conjecture is a theoretically-based hypothesis, evidence to support or reject the conjecture provides for a possibility to contribute to broader research literature. Class (2009), chapter 8 provides an excellent example of how embodied conjectures were used in the construction of a learning experiment.

Bielaczyc (2006) specifically addresses the design of curricula involving technological tools. Her Socially Interactive Framework seeks to identify the critical variables of classroom social structure, which she lists as:

- Cultural beliefs such as how learning and knowledge are conceptualized, how student and teacher social identity is understood, and how the purpose of the tool is viewed.
- Practices such as the planned learning activities, associated participant structures of students and teachers, and the coordination of activities using and not using the tool.
- Social-techno-spatial relations, such as the configuration of the space between students, teachers, the Internet, and the tool.
- Interactions with the outside world.

3.5. Implications of these Methods for the Design Process

She emphasizes that these variables have the ability to increase or decrease the effect of the tool if not explicitly considered as part of the design. This proposal is not dissimilar from the notion of alignment, especially aligning with learning characteristics and interests.

As a relatively young field, design-based research's very boundaries are still in question. Bell's analysis of the field found no fewer than four foci of research; developmental psychology, cognitive science, cultural psychology, and cognitive anthropology variants of design-based research (Bell, 2004). Sandoval and Bell (2004) ask three still unanswered questions regarding design-based research:

- What exactly counts as design-based research?
- What kinds of knowledge can design-based research produce?
- What standards do, or should, exist to judge the quality of design-based research?

3.5 Implications of these Methods for the Design Process

Participatory design, design-based research, and alignment are used in the following three chapters to enable the designer to discover design constraints and goals stemming from the educational context into which the design will be placed. Each of these methods brings something different to this discovery process:

Participatory Design. Participatory design is used in the following chapters primarily as a method for familiarizing the designer with the educational setting, and for setting the initial design trajectory. It works very well to generate specific ideas (contributed by either the designer or the participants), and to rapidly prototype design ideas. Once a design becomes more fully developed, and thus more difficult to alter quickly, the participatory design method becomes less powerful. In some cases, starting the design process with participatory design can also lead to the creation of an enduring partnership between designer and participant(s) such that even when participatory design is no longer a useful method, the original participants still play an advisory role in the project.

Design-Based Research. design-based research (DBR) is concerned with narrowing the design space of an intervention or tool through multiple iterations of a cycle of design, pilot, and evaluation of interventions to achieve increasingly better outcomes and alignment. It is this characteristic of the method that we rely on most in the following three accounts. However, there is a significant difference between design-based research and the type of work described in the next three chapters - simply put, we had less control over our interventions than typical DBR

3. Methods for the Design of Configurable Embodied Interfaces

projects. In DBR, researchers create and often administer the entire intervention, and attempt to account for all exogenous and endogenous variables. This approach is not the case for some of the work I describe, especially the work in Chapter Six in which the research team has no control over the goals or assessment methods of the intervention.

The reason that DBR projects are usually more tightly controlled than the case studies stems from somewhat differing goals between DBR and our work. Our goal is the creation of educationally relevant tools that can be *broadly disseminated* at the completion of the design process. There are two outcomes to DBR, the enacted intervention, which *does not need to be broadly disseminable*, and an analysis of the intervention and underlying embodied conjecture that ideally leads to broader insights into learning and contributes to the learning sciences - information that is broadly disseminated through research journals. Embodied conjectures in design-based research, as defined above, are theoretical hypotheses around which the entire intervention, including the goals, instruction, and assessment, are designed - and while the intervention and the conjecture are refined through multiple cycles of design, pilot, and evaluation, the idea is that the intervention eventually serves to prove or disprove the conjecture. Fortunately, despite the fact that we were not always able to define all aspects of the intervention, our use of backwards design to align the goals, instruction, assessment, and features of the tool in the intervention allows us to use the notion of embodied conjectures to make broader contributions to the learning sciences.

Alignment and Backwards Design. The backwards design process guided the design portion of the Design-Pilot-Evaluate cycle; we began each new design phase by listing the learning goals and deriving from them assessments, instruction, and tool features. This fairly hierarchical structure became more useful during the later DPE cycles; in the first cycle, the wide-open design space and use of participatory design meant that our design ideas were changing too rapidly for backwards design to be effective.

3.5.1 A Common Design Process

Taken together, these three methods are used in a common design process that applies to each of the three projects presented in the next chapters. Each project begins with partnerships between designers and educators; in one case these partnerships are formed prior to the formation of idea for a tool(as in Chapter Four), and other times the research team has a vague notion of a potential tool and engages in participatory design to focus the idea (as in Chapters Five and Six). The first of the cycles of design-pilot-evaluate (DPE) begin, though in this initial cycle, design and pilot tend to be merged into one participatory design experiment. Eval-

uation of this pilot points to areas in which the design can be improved and refined, kicking off the next design cycle. The first cycle generally leads to the discovery of a number of design constraints specific to the educational setting. These constraints allow us to start subsequent DPE cycles using the notion of alignment and employing backwards design. Each project then iterates through one or more DPE cycles, with the aim to achieve increasingly better alignment and learning gains.

3.5.2 Implementation

In the abstract, the methods and design process offer a tidy approach to solving the problem of creating an educationally relevant tool. In the real world, many questions about implementation remain, and it is for the purpose of providing answers to these questions that the case studies of chapters 4-6 and the in-depth summary and analysis of chapter 7 exist. To open these chapters of real-world experience, the author suggests some questions to consider:

- How different are the first and subsequent design cycles in each project?
- How does one define well-aligned?
- How does one know when to stop iterating through our design-pilot-evaluate cycles and attempt to disseminate?
- What are the differences between this design process and a standard engineering design process?

3. Methods for the Design of Configurable Embodied Interfaces

Chapter 4

The Braille Writing Tutor

The Braille Writing Tutor¹ is a configurable embodied interface developed to allow visually impaired children in developing countries to learn to write Braille using a slate and stylus with the benefits of a computerized tutor assessing and instructing them (Kalra et al., 2007a,b). This chapter describes the design trajectory of the Braille Tutor project, starting with ideation and proceeding to the current system. Specifically, we discuss Braille and how it is written, the learning goals, instruction, and assessment into which the tutor was introduced, how the idea originated, and the design-pilot-evaluate cycles that have been completed.

4.1 Writing Braille

Braille, the primary method of reading and writing for the blind, is a tactile system in which embossed dots representing letters, symbols, and numbers can be read with the fingers. A Braille letter is formed by embossing some subset of six dots arranged in a 3 x 2 cell. Figure 4.1 shows schematics of a Braille cell and a photograph of a page of Braille. For the blind, literacy in Braille is often the key to independence at home and work (Schroeder, 1989). Despite the advantages that Braille literacy imparts, there are a number of barriers to learning Braille in developing countries. According to the Mathru Educational Trust for the Blind in Bangalore, the main barrier in India's case is limited opportunities for education because parents and families of blind children often do not realize the possibility or value of educating their child. Even when the desire to educate is present, children may not receive sufficient guidance at home or in traditional schools because very

¹Much of the research reported in this chapter was originally published in a paper (Kalra et al., 2007b) that was co-written with Nidhi Kalra, Tom Stepleton, Daniel Dewey, and M. Bernardine Dias. As some sections are excerpted from the paper, their permission was received before reprinting here.

4. The Braille Writing Tutor

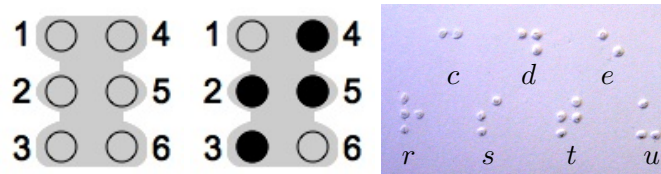


Figure 4.1: A schematic of a Braille cell (left) and the letter ‘t’ (center). The black circles represent embossed dots while the light grey circles represent un-embossed dots. A sample of Braille (right).

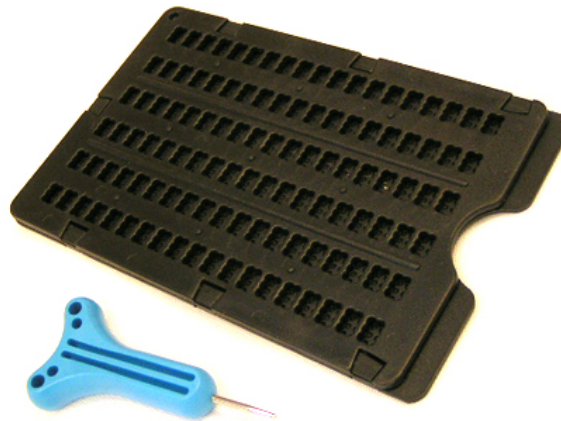


Figure 4.2: A Braille slate and stylus.

few people are trained to teach Braille. Unfortunately, poorer areas tend to have both a disproportionately high number of blind people (World Health Organization, 2004) and fewer resources for educating them.

Furthermore, the traditional method of writing Braille itself creates formidable challenges to literacy. In developed countries, Braille is usually embossed with a six-key typewriter known as a Brailler; these devices are fast and easy to use but also cost over US\$600 each (Perkins, 2006). In developing countries, such devices are prohibitively expensive and Braille is almost always written with a slate and stylus as shown in Figure 4.2². Using these tools, Braille is written from right to left so that the page can be read from left to right when it is removed from the slate and turned over.

²Reprinted with permission from LightHouse for the Blind and Visually Impaired.

4.3 Ideation

The idea of the Braille Writing Tutor was conceived through extensive discussions with teachers from the Mathru Education Trust for the Blind in Bangalore, India, our first partner in a collaborative design process. The Mathru School is a residential and educational facility for approximately 50 children enrolled in grades one through eight. In addition to providing the standard curriculum for the state of Karnataka, Mathru teaches daily living skills such as mobility and food preparation, offers vocational training such as computer classes, provides medical care, and encourages talent, personality development, and self-confidence. Six of the eight teachers at Mathru are themselves blind or visually impaired. Additionally, most Mathru students come from the very poorest of Karnatakas villages where they may previously have had no access to running water or electricity, much less computers and electronics.

The concept of the Braille Tutor originated from an in-depth, consultative dialogue with Mathru. We approached Mathru knowing that blindness can create extreme life challenges for those in developing countries, but without a clear picture of the specifics. The teachers at Mathru were intimately familiar with those challenges but were unaware of how technology could help. Therefore, we began by asking Mathru for a laundry list of all the difficulties their students faced, from education to personal care to food to transportation. Their list included being unable to determine whether water was clean or dirty, but, to our surprise, pointedly did not include things like having difficulty playing team games such as cricket. The list revealed that reading and writing were problematic for young children and to investigate further we requested photos and videos of their students writing Braille at different levels. From these videos we identified the writing difficulties mentioned earlier in this chapter (many of which were never specifically articulated by teachers or students) and realized that technology could play a role in mitigating these challenges. We developed the Braille Tutor concept through dialogue with researchers at Carnegie Mellon University, with blind adults in the Pittsburgh area, and through continued discussions with Mathru. The participation of the teachers in the ideation process was especially important, as clear needs and goals were not specifically articulated by the end-users. In the case of the tutor, the Mathru school did not have much exposure to technology and its potential uses and we (the designers) did not have much exposure to the needs of the visually impaired in developing regions. Thus, this iterative and conversational process helped us to explore design concepts as we clarified our concept of the school's needs, and iterate on these ideas with the teachers in the school until we came to a common understanding of what was possible and what was useful.

4.3.1 Other Tutor Systems

Before beginning work on the tutor, we collected information on other tutoring systems. Intelligent tutors exist for a range of subjects and skills including math (Koedinger et al., 1997), English reading (Mostow et al., 1994), speaking (Bunnell et al., 2000), and computer programming (Weber and Brusilovsky, 2001). Encouragingly, many of these tutors have achieved success in the classroom. Nevertheless, they have limited impact on our goals for a number of reasons. Firstly, of course, they are not tailored to writing. Secondly, they require sight and use written instruction extensively in the tutoring process; in contrast, a tutoring system for the blind usually depends heavily on audio feedback. Thirdly, this limitation means that most existing automated tutoring systems for the blind are fairly simple (e.g. the Talking Braille Tutor® teaches only individual symbols (Peaco, 2004) and cannot teach complex skills such as writing using a slate and stylus. A notable exception is the Speech Assisted Learning (SAL) device that tutors reading and math using a stand-alone refreshable Braille display (Chamot, 2006). Fourthly, like SAL (which costs US \$4600) and the Talking Braille Tutor (which costs US \$300), most assistive technology is prohibitively expensive because the number of blind people who could potentially access it is very small. An exception is the low-cost Sparsha system (Lahiri et al., 2005), which is a software package specifically designed for the blind in India, supporting English and a dozen native Indian languages. With Sparsha, a blind user can type on either a normal keyboard or a novel input device that uses the six-key modality of writing Braille, and translate the Braille to text in any of the supported languages. As such, the Sparsha system complements the Braille Tutor nicely, with the former improving the experience of writing Braille on a computer and the latter teaching manual Braille writing skills. We believe that the Project LISTEN English reading tutor (Mostow et al., 1994), which listens to children read aloud and provides audio feedback, has the most relevance to our work as it uses an alternate medium of interaction (spoken words) and teaches a basic literacy skill (reading). Nevertheless, the need remains for a writing tutor specifically tailored to meet the needs of the blind in developing countries.

4.4 Design, Pilot, and Evaluation of the First version of the Braille Tutor

The Braille Tutor as originally conceived consists of both a hardware and software design component - the hardware would function as an input device for a computer to detect Braille writing, while the software would run on that computer and provide feedback to the student. The hardware design portion began in January 2006,

4. The Braille Writing Tutor

while much of the software was intentionally written during the first pilot of the Braille Tutor at Mathru in summer 2006. During the hardware design phase, we were in frequent contact with Mathru and we were also able to test some of our early designs with members of Pittsburgh's blind community. During this period, we developed an understanding of the design constraints on the hardware device, dubbed the 'E-slate', imposed by both the learning goals and curriculum used at Mathru, and those imposed by the educational setting. We begin by describing what we learned of the learning goals, instruction, and assessment at Mathru.

4.4.1 Goals, Instruction, and Assessment of a Braille Writing Curriculum

Our early partnership with Mathru provided us with details of the goals, instruction, and assessment of a Braille writing curriculum. Though some of the details of the instruction and assessment may be specific to our initial partner school, we believe that the overarching learning goals, challenges, and assessment methods are fairly standard across schools for the blind in developing countries. Subsequent experience with two additional schools, one in Zambia and another in Qatar, has reinforced this assumption.

Goals

The ultimate learning goal of any Braille writing curriculum is for the student to master writing Braille. This is typically accomplished over several years, and consists of several chronologically ordered subgoals:

- Students must understand the concept of the six dot cell, and how specific letters are encoded using patterns of dots. This skill is crucial for both reading and writing.
- Students must then learn that writing involves embossing these dots, and that once one letter has been embossed, it is important to move to the next cell to write the next letter.
- Once students can switch cells, they must then learn to switch from the end of one line to the beginning of the next.
- When students have mastered the alphabet, they can move on to Grade 2 Braille, which includes punctuation, patterns for numbers, and shorthand abbreviations (for example, in English there is a single six dot pattern for the word 'and').

4.4. Design, Pilot, and Evaluation of the First version of the Braille Tutor

Throughout this process students are continuously pushed to write faster, so that students can write fast enough to take notes in other classes. Students who have mastered Braille writing with slate and stylus can write at the same speed as sighted people can write with pen and paper.

Assessment

Teachers use both formative and summative methods for assessing students in these classes. As most of the teachers themselves are blind, formative assessments involve teachers placing their hands on a student's hands while the student is writing. They will then attempt to correct the student if they don't feel her moving from one cell to another or switching lines properly. For summative assessments, teachers rely on frequent quizzing - for example, one popular assessment is to ask students to write out the alphabet; students are given one minute and must write as many letters as they can, starting with A. The resultant paper is graded based on how many letters were successfully completed. As students improve, assessment tasks focus on sentence composition and punctuation; eventually entire essays are written using the slate and stylus.

Instruction

Instruction proceeds in small classes of five or six students, with a teacher lecturing briefly and then suggesting practice problems. Classes are divided roughly by age and ability level. While students practice, the teacher will walk around the room and give individual attention to students, providing formative assessment as previously described. At Mathru, students in grades two to six who have not yet mastered Braille are given one hour of instruction per day.

Challenges to Instruction

As we observed videos of the students being taught Braille writing and through conversations with teachers, we identified three major challenges to learning to write Braille using slate and stylus:

- Since the embossing method of writing Braille creates raised dots on the opposite side of the paper, Braille is read from left to right but must be written from right to left. Therefore, children must learn mirror images of all letters, doubling the alphabet and creating a disparity between the written and read forms of each letter.

4. The Braille Writing Tutor

- Critical feedback is delayed until the paper is removed and then flipped over and read. For young children, this delay can make Braille conceptually challenging since the act of writing has no discernible, immediate effect. It also takes longer for both the student and the teacher to identify and correct mistakes and this slows learning.
- The paper used for embossing is expensive and in short supply.

These challenges sit at the intersection between instruction and assessment, and stem from the specific dynamics of writing with a slate and stylus, and formed the basis of some of the design constraints and goals.

4.4.2 Design Constraints

There were two sets of constraints and goals that defined our first hardware design process; one relating to the educational value of the tutor, the other relating to the project goals of creating a piece of hardware that was inexpensive and robust enough to function in the educational setting.

Educational Constraints

The educational constraints were primarily garnered from our understanding of the learning goals of the curriculum at Mathru, and the challenges to learning that we identified.

- *Transferable Learning.* Students learning writing with the tutor must be able to transfer this learning to a regular slate and stylus. Therefore the tutor experience must be made as similar to using the slate and stylus as possible.
- *Multiple Cells and Rows.* Students need to learn to move from cell to cell and row to row, therefore the new input device should have multiple cells and rows.
- *Immediate Feedback.* The stylus position should be reported to the computer at all times to enable immediate feedback.
- *Mirror Mode.* The tutor should have a mode where students write in the same direction that they read.
- *Easily Understood.* The tutor's speech module must be understandable given the age and background of the learner. Depending on the circumstances, it may use local languages, local dialects, and age-appropriate voices.

4.4. Design, Pilot, and Evaluation of the First version of the Braille Tutor

It is important to note here that these constraints were not all well-defined at the start of the design process - instead, they represent the constraints as we understood them at the point of the last hardware iteration of the first version of the E-slate, just prior to the pilot of the tutor at Mathru. For example, we were not aware of the importance of having multiple rows on the input device until a communication with Mathru three months into the design process, and this constraint caused a revision of the E-slate.

Robustness and Cost Constraints

In addition to constraints relating to educational goals, the setting of the project indicated a number of additional constraints.

- *Low-Cost* Unlike other tutors, ours must be affordable to members of the base of the economic pyramid who live on less than US\$2 a day. We hope to make it affordable to every village or rural school even if it cannot be affordable to individuals. Our target price was US\$20 per unit for systems requiring an external computer. This price would be achievable if large quantities (> 1000) of tutors were purchased.
- *Low-Power* In developing countries, electricity may be unreliable, in limited supply, or simply unavailable. The tutor must maximize the resources available, be robust to unreliable power, and be able to be powered by alternative sources. Our target power consumption is 300mW, or enough to operate for about 50 hours on 4 AA batteries.
- *Robust* The tutor's hardware components must be rugged enough to be extensively used and abused by students for a long time.
- *Easily Operated* The tutor must be easily and independently operated by a blind person. This means that both the hardware and the software must be accessible to someone with little or no computer literacy or experience with electronics. It must also provide guidance that can be utilized without the presence of a teacher.
- *Locally Maintainable* The tutor must be designed with easily available components so that if any of the electronic components fail, repairs can be made on-site or nearby. This means using commonly available materials and manufacturing techniques.

These constraints describe the guiding forces behind our design process, but much of the detail is omitted - for this, we detail the resulting E-slate design in the next section.

4.4.3 E-slate Design

We designed the first version of the E-slate over a six-month period at Carnegie Mellon University; the design and assembly of a field-testable prototype constituted a V-Unit, an independent study course offered through the TechBridgeWorld initiative (Dias, 2010). During this period, the E-slate circuit was refined and re-designed four times; after each iteration we gathered feedback from local engineering and human-computer interaction communities as well as from Mathru and made improvements.

The most challenging aspect of the design of the E-slate was to match the user experience as closely as possible to that of a regular slate and stylus while still meeting our low-cost and low-power constraints. As shown in Figure 4.4, the input area of the E-slate consists of two rows of 16 Braille cells each and is integrated directly into the circuit board to maximize robustness and minimize cost. A cutout from a normal plastic slate is placed over top of the two Braille rows to give students the exact same feel as when writing on a standard slate. We used an extremely low-cost and low-power microcontroller, the Atmel ATMEGA88 (Atmel, 2006) in conjunction with a custom resistor network decoding circuit to handle the sensing of stylus location in the input area. If the stylus is in contact with any dot in the input area, the Atmega88 senses which dot in which cell the stylus is contacting and transmits the information to a computer over the serial port. The stylus is a standard Braille stylus modified to connect it to the slate via a wire soldered to its metal tip.

A small speaker and four buttons provide a basic interaction modality between the student and the E-slate, even when it is not connected to a computer. The speaker emits a tone whenever the stylus makes contact with a dot in a Braille cell, with each dot being mapped to a different musical note. The four buttons activate and deactivate several features. Button 1 toggles the heartbeat LED, which was used by us as a visual indicator for debugging early versions of the E-slate's software. Button 2 mutes the speaker so that more advanced students can use the E-slate without tonal feedback. Button 3 reverses the direction of the text, allowing students to choose between writing right to left (as is typical when writing with a standard slate and stylus) or left to right (which is the direction Braille is read in)². Button 4 was included to allow an unprogrammed hardware input to the tutor software - it simply sends an acknowledgment to the computer when the button is pressed. The stylus connection port also contains two additional inputs for buttons located on the stylus. One button is placed on the stylus for students to indicate the completion of a character or word.

²During our field study, we found that writing from left to right did not support transfer of writing skills to a regular slate and stylus and so disabled this feature.

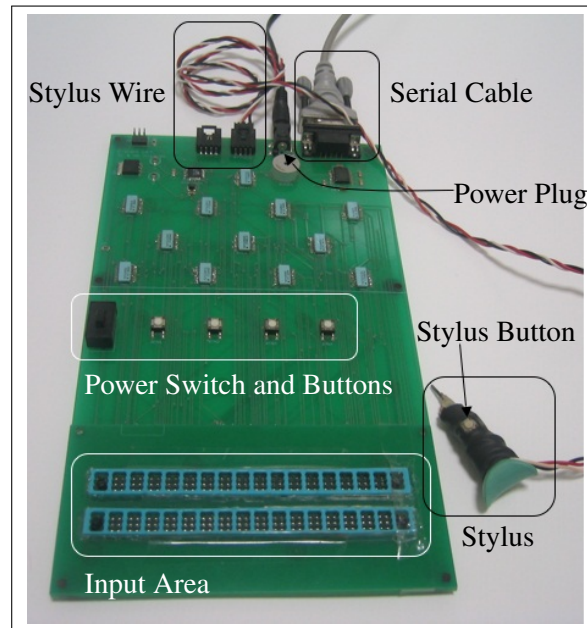


Figure 4.4: The prototype E-slate taken for field testing at the Mathru School for the Blind.

We initially considered several off-grid methods of powering the E-slate: solar cells, batteries powered by a solar charger, regular disposable batteries, and a hand crank to charge batteries. Unfortunately, except for disposable batteries which have a low but recurring cost, all of these options were more expensive than the E-slate itself. We decided that since the current iteration of the E-slate requires a computer powered by the electrical grid to be useful, expensive off-grid solutions should be explored at a later time. The E-slate is powered by an inexpensive AC/DC wall adapter which has an input range of 100 to 240 VAC at 50/60 Hz, and outputs up to 300 mA at 6 VDC. This input range is globally compatible with all electrical grid standards, and so the only adaptation that must be made to use it in different countries is to purchase an appropriate plug adapter.

4.4.4 The Tutor Software

The E-slate hardware and a basic software environment to allow recognition of the stylus position by a host computer was completed in time for the summer field study. This architecture was sufficient to allow rapid development of the host tutoring software during the first two weeks of the six-week field study; during this

4. The Braille Writing Tutor

period, the lead researcher worked in close coordination with the teachers at the Mathru School for the Blind. The software was tailored to the needs of the students throughout the course of the field study. Together, we outlined three main stages of skill acquisition for the Braille student. The first step is to understand the concept of Braille and to emboss the six dots in a cell. The second step is to learn the unique combinations of dots that make up each letter and write the alphabet. The third step is to put letters together into words, put words together into sentences, and learn math symbols and punctuation. We created three different tutoring programs with emphasis on each of these skills and with capabilities to transition to the next skill. These programs roughly mapped to the second, third, and fourth standards (equivalent to US grade levels).

Although Braille forms exist for many languages including the students' primary languages of Kannada and Tamil, they are taught Braille in English first because it is the standard approach and relatively simple (many Indic languages have more than 64 (2^6) characters and a single character may require more than one cell). Therefore the software tutor was limited to English Braille.

The software tutor receives input from the E-slate regarding the learner's actions on the slate, where an "action" is either a contact between the stylus and the slate or a press of one of the five buttons. The tutor interprets these actions using a state machine and provides feedback tailored to the skill being learned.

Second Standard Braille Writing Tutor.

The second standard tutoring software meets the needs of the beginner student learning the concept of the six-dot cell. Whenever the stylus is touched to the slate, the tutor speaks the position of the dot that was touched. This helps the student understand the cause and effect relationship between embossing on a slate and creating letters. It also teaches the spatial relationships between the different dots. The second standard tutor smoothly transitions to teaching and reinforcing the alphabet once the six-dots concept is learned: when the student presses the button on the stylus, the tutor will speak the letter written on the current cell. For simplicity, none of the other buttons on the E-slate have any effect in this tutor. For the second standard tutor, we used a Mathru teacher's digitized voice for the dot and letter feedback. Firstly, there is a finite number of letters and positions so digitizing the feedback was feasible. Secondly and more importantly, we found that younger children using the second standard tutor may not be familiar with foreign accents and would feel more comfortable and learn faster if they heard their own teacher's voice.

Third Standard Braille Writing Tutor.

The third standard tutor partially meets the scaffolding needs of advanced students. The third standard tutor retains the position and letter feedback for reinforcement and spelling practice. It encourages speed by not requiring a student to press the stylus button to register a letter; instead, a letter is registered whenever the student transitions between cells. It additionally provides word feedback by keeping a character history and uses text-to-speech (TTS) synthesis to speak the last word written when a student “double-clicks” the button on the stylus. For the TTS engine we use Cepstral’s ®female American-English Callie³ voice which Cepstral donated to the project. To augment the word feedback, we provide functionality both to erase and then correct previously-written letters and to spell the letters in the last word. This allows students to work on spelling as well as writing.

Fourth Standard Braille Writing Tutor.

The fourth standard tutor provides the remaining scaffolding for the advanced student. Here, we remove the position feedback as the learner is presumably familiar with the positions and finds such feedback cumbersome. This tutor additionally provides feedback on the last sentence written using the same text-to-speech engine. It also recognizes math symbols and punctuation which may require multiple cells per symbol, which have two symbols mapped to the same letter, or both. The tutor uses a decision tree to determine which symbol is intended.

4.4.5 Field Study and Evaluation

While at the Mathru school, we surveyed and observed students and teachers with regards to learning with, acceptance of, and usability of the tutor. Our target group was students in grades two and three as they had begun to learn Braille but had not yet mastered it, and so we would expect them to receive the greatest benefit from the Braille Tutor. This group consisted of six students in each grade for a total of twelve students. Ordinarily they had Braille class in four one-hour periods each week; for our study, they used the Braille Writing Tutor for forty minutes of the one hour and used a regular slate and stylus during the remaining twenty minutes. Although we focused our study on these twelve students, the tutors were constantly operational and we allowed any interested student or teacher to use them. As our goals for this pilot were to evaluate the feasibility and features of the Braille Tutor and the number of students involved was not statistically powerful, we forwent the opportunity to have a control group or crossover study in favor of allowing

³<http://www.cepstral.com>

4. The Braille Writing Tutor

students unlimited and unstructured access to the tutor to measure their interest and responsiveness.

Learning Gains

We measured all twelve target students' proficiency in Braille once at the beginning of our one-month study and once at the end and evaluated their improvement by assessing the skills learned; students were tested on a regular slate and stylus to ensure that student learning with the tutor transferred to regular slate and stylus despite the experiential differences between the two systems. Although it is difficult to attribute improvements solely to the tutor, we can determine its impact somewhat by understanding students' prior abilities. We tested how many cells the students could fill in with all six dots embossed (which we call the "six-dots test") and how many letters they could write (which we call the "alphabet test") in a fixed period of time; these are standard assessments the Mathru teachers use at this grade level to measure Braille writing proficiency. We also evaluated the number of mistakes made during the test; a mistake was defined as erroneously omitting or adding a dot or failing to leave a space between letters.

We can categorize the students into three groups based on a qualitative analysis of their pre- and post-trial test results. Four of the twelve students (referred to as Group A) demonstrated complete understanding of the Braille concept and could write the alphabet quickly and with few mistakes *before* we began the study. The tutor mainly provided advanced practice for these students. The second group (Group B) consisted of five students who lacked proficiency before the study but attained demonstrable proficiency by the end. The third group consisted of three students (Group C) who did not understand the concept of Braille and showed a significant lack of proficiency *both* before and after the study. We are interested in these last two groups to understand how the tutor may have helped those in Group B and why it did not help those in Group C.

Two of the five students in Group B (the group showing improvement) understood the concept of Braille before the study began but made frequent mistakes. At the end, they wrote significantly faster and made far fewer mistakes. Specifically, one student's abilities jumped from writing seven letters with four mistakes to writing thirteen letters perfectly, and the other student's abilities jumped from writing 23 letters with eight mistakes to writing 26 letters with one mistake. For these students, we suspect that the improvement was probably just the natural result of practice. Although the Braille Writing Tutor may have sped up their learning in comparison to using a regular slate because it increased their interest in writing, we cannot confidently make this claim without a control group.

The remaining three students in Group B made significant *conceptual* advances:

4.4. Design, Pilot, and Evaluation of the First version of the Braille Tutor

in the first proficiency test they showed a distinct lack of understanding of Braille and were unable to emboss all six dots in a single cell. By the end of the study, two of the three students each wrote five letters with no mistakes and the other student completed the six-dots exercise in three cells. From our discussions with their teachers, we believe this is probably a direct result of getting immediate feedback from the tutor, as these students had not demonstrated any understanding of Braille in the several months of instruction prior to the study. The case of one particular student highlights one way in which the tutor's instant feedback can produce a necessary conceptual advance. This student's writing usually consisted of a single cell with all the dots embossed, regardless of the assignment. It appeared that he had no conceptual understanding of Braille though he had been in Braille class for a few years. To the teachers' delight, this student began writing Braille as soon as he was asked to use the tutor. Apparently this student had always been writing every letter in the same cell, thus creating the completely embossed single cell. This was not evident to the student's teacher because the teacher herself was blind and could only feel the results of the writing on the paper; moreover, the student was unable to communicate well and explain what he was writing. Because the second standard Braille Writing Tutor interprets letters after the student presses the button on the stylus, it did not matter that they were all in the same cell. Additionally, the teachers were able to hear the result of this student's writing immediately and soon realized that there was a gap in the student's understanding. In this way, the tutor acted as an assessment tool: it highlighted the student's unique difficulties and was able to provide insight to the teachers.

Although it is not clear why Group C did not benefit from the tutor (or from a month of Braille practice overall), our discussions with the teachers lead us to believe that they may not yet be developmentally ready to learn Braille. That is, we believe that members of Group C may have multiple learning disabilities or may need to develop basic social and personal skills.

Tutor Acceptance

The teachers' response to the tutor was very positive: they believe the tutor can help students overcome difficulties in writing Braille. Moreover, the teachers were eager to continue having the tutor in the classroom and have continued using versions of the Braille Tutor to the present day. Their involvement in improving the software also highlighted their belief in the value of the tutor and they are further interested in bringing the tutor to other schools for the blind.

We evaluated the target students' attitude towards the tutor through an interview at the end of our six-week study. We asked them to agree or disagree with several statements:

4. The Braille Writing Tutor



Figure 4.5: Students at Mathru use the Braille Writing Tutor.

1. I find the tutor useful.
2. I dislike using the tutor.
3. I prefer writing on a regular slate to writing on the E-slate.
4. I want to continue using the Braille Writing Tutor.
5. I think the tutor will help me in learning to write Braille.

To reduce the likelihood that the students would answer favorably simply out of respect or an eagerness to please, we explained to them that honesty was very important and had the teachers administer the survey when we were away from the school. Nearly all of the students (10/12) found the tutor useful and believed it would help them in writing Braille; the remaining two students were in the second grade and had significant difficulty with the overall concept of Braille. However, all students strongly disagreed with the statement “I dislike using the tutor” and emphatically agreed with the statement “I want to continue using the tutor.” Finally, nearly all students (10/12) preferred writing on the E-slate to the regular slate, primarily because they enjoyed the interaction and the voice; the ones that did not were the youngest students who had difficulty finding the button on the stylus. While this indicates a positive response to the tutor, there is concern that



Figure 4.6: The teachers use the Braille Writing Tutor at the tutor station we set up in the computer lab.

students will adopt the E-slate completely because it is easier to use and more interesting. We believe that this can be avoided in the school setting by having frequent exercises on a regular slate and stylus.

We evaluated the students' enthusiasm for writing Braille by observing which students wrote Braille outside of the Braille class period and the frequency of this use. We found that almost every student in the target group used the tutor outside of class at least once a week – only three of the students used it only during class. Interest varied from student to student: a few used the tutor on an almost daily basis while others used it once or twice a week. We also frequently found older students not in the study group and even teachers using the tutor in their spare time simply out of interest (see Figure 4.6). Though highly proficient in Braille, they enjoyed hearing the tutor speak their words and thoughts. However, students' interest in writing Braille extended only to using the tutor; it was rare to find students using the slate and stylus outside the classroom both before the introduction of the tutor and during the field trial.

Usability Observations

In addition to determining the learning gains and interest in the tutor, we observed how easy or difficult it was to set up and use by both teachers and students, asking

4. The Braille Writing Tutor

ourselves the following questions:

- *Hardware assembly.* What are the minimum abilities required to assemble the hardware? How long does it take to learn how to assemble it? How long does it take to assemble?
- *Software installation.* What are the minimum abilities required to install and then start the software? How long does software installation take?
- *Tutor use.* How long does it take to learn how to start the tutor? What are the minimum abilities to start the tutor? How much and what type of instruction is required to learn how to use all the functionalities of the tutor?

We evaluated these features by observing both the teachers' and the students' use of the tutor. Hardware assembly involves plugging one end of the power adapter into a wall socket and inserting the other end into a socket on the E-slate, connecting the serial cable between the E-slate and the PC, and connecting the stylus to the E-slate. We found that the blind teachers and students could only connect the power supply; connecting the serial cable and the stylus was impossible for them because the connectors are small and keyed. We trained a low-vision teacher with prior computer experience to successfully connect the serial cable and power in a single 30-minute training session but the stylus was simply too small to manipulate. Ultimately, this teacher was able to connect the power supply in under a minute but took 2-3 minutes to distinguish between the male and female ends of the serial cable, find the serial port on the back of the computer, and connect them. These difficulties highlighted that we needed to completely redesign the hardware connection method to make assembly possible for our target group.

Software installation was also challenging as it involved a number of complex steps that required familiarity with advanced features of the Windows operating system. In the end, we installed all the software ourselves and successfully trained sighted teachers with significant prior computer experience. This highlighted that software installation also needed to be redesigned significantly to install all components in a single step.

Nevertheless, once the components were installed and connected, the students could find the switch and turn on the E-slate after only five minutes of training. We noticed two issues with students using the tutor: the smaller children had difficulty determining when the switch was set to the on position and when it was in the off position, and both beginner and advanced students were initially afraid of receiving an electric shock from the electrical wire leading from the stylus to the slate. The older students (4th standard and higher) who had prior experience using a PC were

able to start the Braille Writing Tutor software on their own with just minutes of instruction; younger children needed a teacher to do it for them.

At the end of the field study we also conducted a survey of the target group regarding usability. With the help of a teacher, we asked the children to agree or disagree with the following statements.

1. I found the Braille Writing Tutor difficult to use.
2. It took a long time to learn how to use the tutor.
3. I cannot understand the voice used by the tutor.
4. I find it difficult to distinguish between the buttons on the E-slate.

Ten of the twelve students stated that it was easy to learn how to use the tutor and all the students could understand the voice used by the tutor even though the text-to-speech synthesizer used an adult female American voice. We suspect this may be because the students are accustomed to the JAWS screen reading software⁴ which uses a male voice with an American English accent. Finally, roughly half of the students (five of twelve) found it difficult to distinguish between the buttons on the E-slate. We believe that labeling the buttons and giving them unique shapes will alleviate this problem.

4.5 Second Design Cycle

Our experiences at Mathru confirmed the feasibility of the Braille Tutor; students were able to use and learn from the tutor, and the tutors could be independently maintained by the school. Simultaneously, the study highlighted shortcomings of the current system and initiated a new design cycle, leading to refinements of both the E-slate and software tutor. The redesign of the E-slate was completed in the spring of 2007, and the modifications to the software tutor began in spring 2007 and are ongoing and driven by new field studies.

4.5.1 E-slate Changes

Several major changes were made to the E-slate based on realizations stemming from our experiences with the 2006 field study. For the most part, these experiences did not add new design constraints and goals to our lists, but led us to a better understanding of ways to meet those goals. We detail the changes made and the underlying reasons for those changes in the next few pages. Table 4.1 summarizes these changes and Figure 4.7 pictures the second version E-slate.

⁴http://www.freedomscientific.com/fs_products/software_jaws.asp

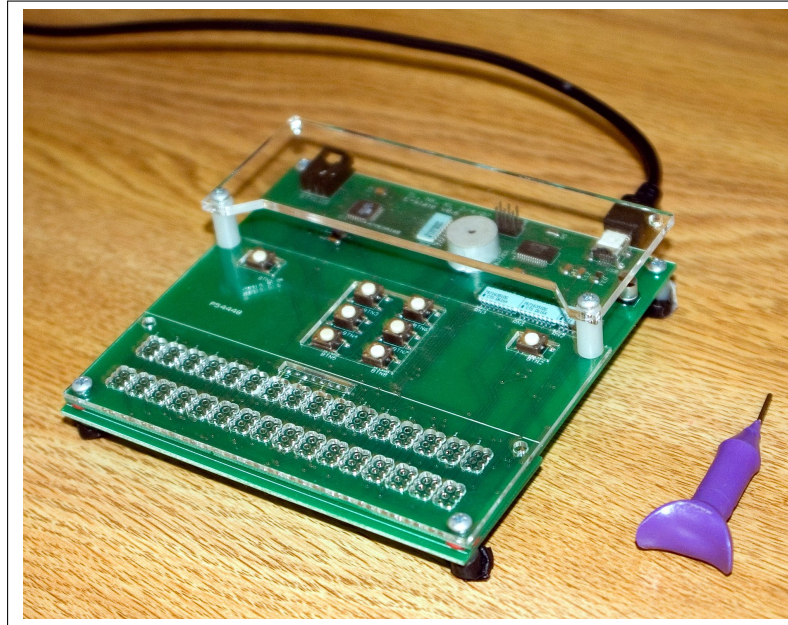


Figure 4.7: Version 2 of the E-slate

Transitioning from Serial to USB

The first E-slate communicated with the computer through a serial port, and had some standalone capability; without connecting to a computer, students could write on the slate and it would beep different notes depending on which Braille dot was sensed. After our experience at Mathru, we realized that a limited standalone mode was unnecessary; students always used a computer with the slate, and computers were generally more available than we had expected. Realizing that the E-slate did not need to be independent from the computer, we changed the communication method from serial to USB. This fairly simple change had a number of important effects:

- USB allows connected devices to draw power from the computer, eliminating the need for a separate power supply and consequently reducing the cost of the system.
- The slate no longer needed a power switch - once connected to the computer, it was powered. This made the E-slate easier to use, especially for younger children who had trouble finding the switch.
- The USB connector is friendlier to plug in than a serial cable, and a separate

Feature	Version 1	Version 2
Computer Communication	Serial	USB
Button Configuration	Four equally spaced buttons and 1 stylus button	6 buttons in a Braille cell arrangement and a button on both the left and right side of the slate
Stylus wired	Yes	No
Power Source	Grid Power	Computer through USB
Braille cell configuration	32 cells in two rows	32 cells in two rows
Dust cover	No	Yes
Standalone capability	Rudimentary	None

Table 4.1: Summary of E-slate changes between version 1 and 2

power cable no longer existed - therefore, the E-slate was easier for teachers to install independently.

Eliminating the Stylus Wire

The first version of the E-slate used a stylus with a wire soldered to the tip and connected to the E-slate through a keyed connector. This was necessary to allow the E-slate to register the stylus position. The wired stylus also featured a button on the stylus that was used by a student using the second level tutor to indicate that a letter was complete. We were unaware that students at Mathru have been taught to be very wary of electrical wires (for good reason!), and so the wired stylus actually presented a major hurdle to student adoption of the tutor; it was only through extensive conversation with the on-site researcher that students became willing to use the stylus. Based on this experience we modified our sensing circuit for the second E-slate to allow the use of a standard, unmodified stylus. Instead of wiring a stylus, the input area of the E-slate actually has two circuit boards that are electrically connected but separated by a small air gap. A metal-tipped stylus pushes through holes in the first circuit board and contacts the second board, creating a sense-able connection between the two. As with USB this simple change had a number of important effects:

- Students were no longer initially afraid of using the E-slate.

4. The Braille Writing Tutor

- Removing the wire brought the experience of using the E-slate even closer to using a traditional slate and stylus.
- The wired stylus was a custom-made, difficult to replace part - by allowing the use of a standard stylus we eliminated the possibility that a stylus being lost or broken would stop students from using the tutor.

Changing the button pattern

The version one E-slate had four buttons spaced evenly in a single line slightly above the input area on the E-slate, as well as a button the Stylus. Students had difficulty distinguishing these buttons from one another, and a few of the buttons affected hardware changes to the E-slate that were written into the firmware of the E-slate (as opposed to affecting the host software). This was done in case the E-slate was being used without a computer, but in the field study these mode changes in the operation of the E-slate were mostly annoying and distracting to the students. Version two of the E-slate used the button pattern in a more educational way: Two buttons were placed on either side of the slate to use as the 'enter' key, replacing the button that was previously placed on the stylus. Six buttons were placed in a Braille-mimicking 2-by-3 pattern in the middle of the slate, just above the cell input area. By placing the buttons in this configuration, we were able to turn the difficulty that students had in identifying identically spaced and shaped buttons into an educational exercise. These buttons can be used by very young students who do not yet have the fine motor skills necessary to write Braille to practice the alphabet patterns. Although these six buttons have no additional current uses, placing the buttons in a familiar grid enables older students to more readily resolve individual buttons should they be given functionality by future versions of the tutor.

Adding a Dust Cover

The first E-slate used a cutout from a regular slate and stylus glued to the input area for use as a raised guide to allow students to feel where the stylus was. The second version replaced this with two custom laser-cut pieces of plastic, to cover the circuit board and prevent students from coming into direct contact with it. This was done to make the E-slate easier to manufacture, and to make it more robust. Although we did not observe this, it is possible for students touching the circuit board of the E-slate to cause static discharges that render the circuit board inoperative.

4.5.2 Early Tutor Improvements

The tutor software was improved during two distinct phases of the design cycle. After the completion of the second E-slate, the software was updated to reflect the hardware changes and to improve the software user experience. The tutor software was also enhanced and improved during a number of pilots in different parts of the world to maintain relevance to local educational settings.

Our early tutor improvements focused on three main areas, described in detail in the following pages:

- Individualizing the tutor by having it adapt to student ability level.
- Developing a new installation program to address difficulties teachers had setting up the software.
- Developing a software library to handle low-level communication between the E-slate and computer. The development of this library was crucial to enabling the customization of the software tutor to different educational settings around the world.

Individualization

Chief among the educational difficulties faced by students in developing communities is the scarcity of Braille training and Braille teachers. Version two of the Braille Tutor sought to further ameliorate this problem by minimizing the amount of teacher attention required for learning. The software introduced a series of exercises forming a basic Braille writing curriculum; as before, the Tutor provided feedback to the student by narrating the student's actions. In addition, the tutor uses an English-language 'Teacher voice'; this voice is distinct from the voice used to narrate the student's actions. The Teacher voice introduced exercises and gave feedback on how the student performed in each exercise. It also took over narration of miscellaneous events such as Tutor mode changes and software errors. By introducing the Teacher voice, we allowed interaction between the student and the Tutor to approximate interactions between the student and a personal teacher.

Students could begin practicing by selecting from six broad ability levels chosen at the start of each session: learning dots, practicing dots, learning letters, practicing letters, learning words, practicing words; these are analogous to the three grade levels used in version one of the tutor. The Tutor gathered information as the student attempts exercises in order to assign the most useful exercises to the student. For example, while the student is learning letters, each letter is monitored as an individual skill. If the student answered exercises involving the letter 'a' incorrectly, the Tutor would assign more exercises to practice 'a'. If the

4. The Braille Writing Tutor

estimated knowledge of the letter in question drops below a certain threshold, the Teacher voice reminded the student of how to write that letter by speaking a sequence of dots. Within each level, students were provided with exercises tailored to their unique needs.

Fast Installation

As help from sighted individuals was required for teachers to set up the Braille Tutor software, we developed a Braille Tutor installer program. The program provides audio feedback to the user at every decision point, with directions on which keyboard key to press to continue installation or cancel. The use of a mouse is not required. The new program cut the installation time to under two minutes and launched the tutor at the end of the installation.

Extensible Software Design

The second version of the Braille Tutor software suite featured a modular structure designed with future applications and development in mind. At its heart is a flexible software library that handles all low-level communication with the Tutor hardware and provides a variety of convenient interface facilities for the application programmer. The library encodes the user's interactions with the Tutor as a series of events of varying semantic complexity, ranging from the immediate insertion or removal of a stylus to the creation of an entire Braille character in one of the cells. Applications may poll a queue of selected events or be notified of them asynchronously through a callback interface.

The interface library tracks the state of the Braille Tutor hardware with an internal state machine model. By creating new state machine descriptions, the library can be adapted to future versions of the Braille Tutor hardware in a straightforward way. To the extent that hardware versions present compatible feature sets, the Braille Tutor library aims to provide a consistent software interface among them all, allowing maximum compatibility for applications.

4.6 On-going Software and Curricular Improvements

Although no further hardware revisions were made, the creation of an extensible software interface has allowed for the recent development of a number of innovative software extensions⁵: these extensions were made in a similar way as the development of the first tutor, namely on-location with partnering groups. More generally,

⁵The author did not contribute to the work presented in this section. It is reported here as part of the case study to show the recent evolution of the program.

this on-going work highlights an important characteristic of configurable embodied interfaces: They are inherently flexible in terms of software and curriculum, so long as the difficult-to-change hardware features are well aligned to the needs of students and teachers. This characteristic has important implications for relevance and disseminability, which we will discuss in depth in Chapter Seven.

In 2008 and 2009, the second version of the Braille Writing Tutor was piloted in four locations - the Mathru school, the Al-Noor institute in Qatar, the Sefula School for the visually impaired in Mongu, Zambia, and the Uhuru Mchanganyiko Primary School in Dar es Salaam, Tanzania. These pilots (Abimbola et al., 2009; Dias et al., 2009) validated some of the decisions made in creating both the second version of the E-slate and in upgrading the tutor software. Among the improvements stemming directly from these pilots were:

4.6.1 Foreign Language Writing Support

A number of foreign languages were added to the Braille Writing Tutor to broaden the potential reach of the tutor. Support for French, Chinese, Arabic, and Swahili Braille were added. Adding new languages that have fewer than 64 characters (the number of characters supportable by a single six-dot Braille cell) was made very easy thanks to the extensible software design.

4.6.2 Local Accents and Languages Audio Support

The audio feedback of the tutor was modified to support local English accents and local languages by asking local speakers to pronounce the letters, numbers, and simple instructional phrases (for example, ‘well done’, ‘incorrect’) used for providing feedback to the student. This is very important for two reasons: The default American English accent is difficult to understand for young students, and when the tutor sounds like a local speaker, it is no longer an alien object to the students. The modular way in which the Braille Tutor software was written made this process very easy - at one school it took as little as thirty minutes, and consisted of simply creating new audio files for the letters and numbers and copying these into a directory.

4.6.3 Motivational Games

A major problem identified in the Uhuru school was that once students mastered Braille, their writing tended to regress because they stopped receiving instruction in Braille. This is very problematic, as students are expected to take notes in Braille

4. The Braille Writing Tutor

in other classes, and if they make lots of transcription errors their notes become unintelligible. To motivate these students to continue practicing, several educational games were created:

- *The Animal Game.* In the Animal game, students hear an audio file of an animal making a noise and are asked to write that animal's name.
- *Hangman.* In Hangman, the computer comes up with a word and tells the students how many letters are in the word. Students must then write letters of the alphabet that they think the word may contain. Students continue guessing letters until they have either found all the letters in the word, or have made ten incorrect guesses.
- *Music Maker.* The Music Maker game is designed to interest students in using the stylus and finding dots through music. In Music Maker mode each row of dots on the E-slate corresponds to different musical notes and each column of dots represents a single time step or 'beat'. By inserting the stylus into a dot on the E-slate, the user can toggle on and off the musical note corresponding to the row of the dot being played at the beat corresponding to the column of the selected dot. The user can also use buttons on the E-slate to control the tempo and to toggle all notes to the 'off' position.
- *Domino Game.* After observing the popularity of dominoes among blind students in Qatar and realizing the similarity in the six-dot structure of a domino block to that of a Braille cell, a two-player competitive game was devised around the notion of the dot patterns of letters and their similarities to Dominoes.

4.6.4 E-slate hardware modifications

Two modifications were made to the E-slate during the pilots at two different sites. In Qatar, teachers suggested that the slate's buttons were too hard and that repeatedly pressing these buttons quickly became painful. Researchers solved this problem by finding a readily available padded sticker that fit over each individual button very well, thus providing a padded surface for the buttons.

At Mathru, teachers suggested that it would be good to create a 'hard-copy' of the stylus' impressions on the E-slate. This was accomplished by placing a hinged embossing surface beneath the hole pattern area on the E-slate, and placing Braille paper between this surface and the circuit board. This way, when students pushed their stylus into the slate area, they could push through the circuit board and onto the paper, creating a paper record of their impressions.

4.6.5 Pilots and Evaluation

The four pilots were conducted over the summers of 2008 and 2009 through the TechBridgeWorld program at Carnegie Mellon. In each cases, researchers (often undergraduate students) visited the pilot locations and engaged in participatory design to ensure that the Braille Writing Tutor would be locally useful. This participatory design process led to the design and testing of the improvements listed above. In addition to the on-site design of new capabilities for the Braille Writing Tutor, these pilots led to several suggestions for further improvements to the tutor and allowed us to evaluate the effectiveness of the current tutor. These evaluations were not carried out in a unified way, and are still mostly based on observations by the researchers and teachers, and surveys and assessments of small groups of students.

Evaluation of Current Tutor

In terms of observations from teachers and researchers at all the sites, there was common agreement that the enlarged Braille cell button pattern was very useful because it allowed students who did not have the requisite motor skills to use slate and stylus to learn the concept of the six-dot pattern. A second common comment was that the feel of pushing a stylus into the slate portion of the E-slate was significantly different than the feel of pushing a stylus into a regular slate, possibly inhibiting transferability to regular slate and stylus.

A study was conducted at the Mathru school to ascertain the effectiveness of the Braille Tutor in helping younger students to grasp the basic concept of the 6-dot Braille cell. An experimental group of 9 children from grades 1 and 2 used the Braille Writing Tutor for two hours three times per week for five weeks. A control group with 9 children of the same age and beginning ability was similarly given two hours of instruction three times per week, but were not exposed to the Braille Writing Tutor. Each student was administered a pre-test to measure their knowledge of Braille. After the completion of 5 weeks of field testing with the Braille Tutor all of the students were administered a post-test that tested the same skills at the same difficulty as the pre-test. Each of the 9 students in the test group showed noticeable improvement and understood the 6-dot concept. In the control group, only 4 students showed noticeable improvement. The Braille Writing Tutor also aided the teachers in identifying conceptual problems students were having with the six-dot concept. For example, one student was unable to identify all six dots consistently. After the student used the Braille Writing Tutor, teachers discovered that the problem was that the student didn't know how to find dot 1, and so would start at the first dot he found and then try to make the pattern from there.

Suggestions for Future Improvement

Several suggestions were made over the course of the studies for ways to improve the next version of the E-slate (the tutor software was modified on the fly and so any suggestions made were incorporated by the end of a field trial).

Teachers expressed interest in expanding the button cell pattern to allow for two or more cells of buttons, so that students could begin learning about switching cells and about longer characters (some characters require multiple cells to express, like punctuation marks and arithmetic operators).

Teachers also suggested modifying the slate area to make it more reliable and to make it feel more like a regular slate and stylus. At the moment, the stylus needs to be inserted completely vertically into the E-slate in order to register a contact, and as styluses don't have a standard diameter, some styluses are much easier to use with the E-slate than others. While these seem like minor details, this essentially means that students write in subtly different ways on the E-slate than with slate and stylus, thus transfer from one to the other is not seamless.

4.7 Summary

The Braille Tutor is an idea that was formed by bringing together two sets of deep content knowledge inhabiting in the minds of people living half a world apart. The Braille Tutor case study exemplifies best how collaboration can occur between teachers and designers even when those people are physically distant, while highlighting the importance of occasional immersion by designers in the educational context. It is my hope that the project will continue expanding in the way it has: with designers travelling to sites to make modifications to the tutor that reflect the interests and educational needs of students at those sites.

Chapter 5

Robot Diaries

Creative technology experiences currently available to middle school students (e.g., Botball, LEGO Mindstorms, and computer gaming) are designed primarily around individuals, both boys and girls, who already have a strong interest in technology and competition. In order to broaden the range of available technology experiences and specifically increase girls' engagement in the technology community, we created the Robot Diaries program¹ (Nourbakhsh et al., 2007; Hamner et al., 2008a,b). Robot Diaries aims to provide a compelling technology experience that is designed from the ground up to engage diverse students in creating an embodied robotic artifact that is responsive to an individual's interests, emotions and activities. Of the three programs detailed in this thesis, Robot Diaries is unique in that the design space was completely open - we decided on the goals for the program, derived the learning goals from this overarching goal, and designed the hardware, software, curriculum, and training materials. This chapter begins by motivating the program and stating the program goal, and then moves on to the origin of the idea, how the idea was refined through a series of participatory design sessions, and how those sessions led to a concurrent design of all of the elements of Robot Diaries.

5.1 Motivation and Program Goal

As technological interaction and electronics artifacts integrate ever more tightly in our lives, it is disquieting to note that engineering enrollments continue to drop throughout the United States (Vegso, 2006). Even more alarming is that women participate in dismally low numbers in fields such as computer science and engi-

¹Much of the research reported in this chapter is excerpted from papers (Hamner et al., 2008a,b, 2010) co-written with Emily Hamner, Debra Bernstein, Kristen Stubbs, and Carl DiSalvo. Their permission was received before reprinting here.

neering, whereas virtually all science and business fields show significant improvement in terms of female participation (Vegso, 2005).

One popular movement to stem the current tide evolves out of a recognition that the pipeline is both the source of today's trends and the strategic place for leveraging real change: improve the technology literacy of students at the primary and secondary level, and the statistics of the subsequent decade may finally turn around (Adams, 2007; Arsenault et al., 2005; Cannon et al., 2007; Doerschuk et al., 2007; Frost, 2007; Hylton and Otoupal, 2005; Morris and Lee, 2004).

Robotics has served as a popular vehicle for such pipeline-based technology literacy programs because of its ability to attract and inspire the imagination of students who are often unmotivated by conventional classroom curricula (Druin and Hendler, 2000). National contests include US First, BEST and Botball, programs that have jointly engaged more than 75,000 students (Botball, 2009; FIRST, 2009a). There is no doubt that some of the students have found the contest-driven problem-solving experience to be transformative. However, these existing pipeline-focused technology literacy programs share a number of features that may limit participant diversity: they are short-term, high-intensity, competition-driven and technology-focused.

In response, we propose a complementary class of activities that we believe can engage and retain the participation of secondary-level students who will not be attracted to the currently available pipeline interventions (Buechley and Eisenberg, 2007; Rusk et al., 2008; Kim et al., 2007). A major aim of Robot Diaries is to increase the technological fluency of participating students. By technological fluency, we mean the ability to manipulate technology creatively and for one's own use. We believe that our focus on fluency-building activities, which encourage creativity and personal adaptation of technology, will engage a more diverse student population with technology and engineering. We additionally hope to demonstrate the characteristics of such a program that will allow it to be broadly disseminated.

The primary goal of the Robot Diaries program is to develop and disseminate a technology education experience that attracts students who are not interested in current technology education programs while still providing an experience which results in students who are both more interested in and more competent with technology.

5.2 Timeline of the Project

The idea of robot diaries was formulated in November 2005 with additional conversations during winter of 2005-2006 clarifying the initial concept. We held a focus group to test some of our initial ideas, and after this focus group engaged in a num-

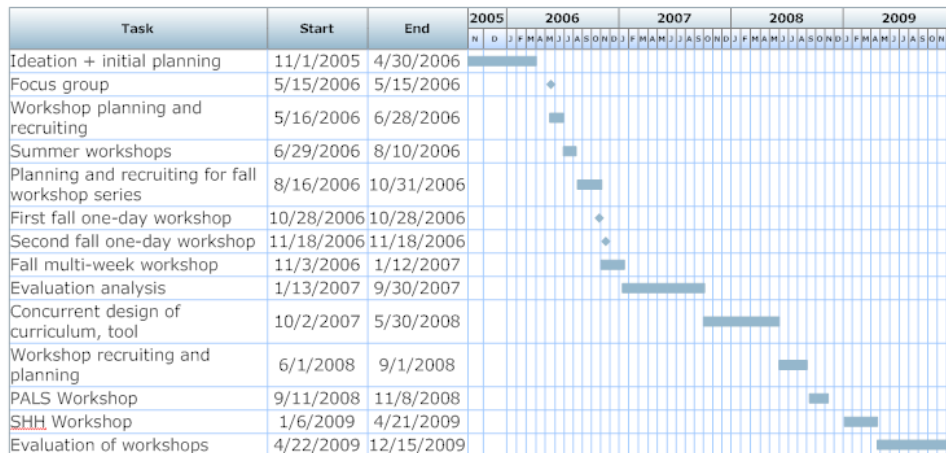


Figure 5.1: Timeline of the Robot Diaries project

ber of participatory design workshops. We held a multi-week workshop in summer 2006, followed by two one-day workshops in fall 2006 that were essentially distillations of the summer curriculum. We had another multi-week workshop in the fall and winter of 2006. The analysis of this workshop was done in spring and summer of 2007, after which we began a concurrent design of a new curriculum and materials. We began recruiting and training teachers in summer of 2008 and held two pilots with the new curriculum and tool in the fall of 2008 and the winter/spring of 2008-2009. The analysis of these workshops was carried out in the summer, fall, and winter of 2009.

5.2.1 My Role in the Project

I have been a member of the Robot Diaries team since the inception of the project. I was primarily responsible for creating and selecting the robotic elements of the kit of parts, including designing a new robot controller for the program. In addition, I played a major role in the design of the various curricula, in piloting the curriculum, and to a lesser extent in formulating and carrying out the evaluations of the pilots.

5.3 Ideation

In late 2005 the Community Robotics, Education, and Technology Empowerment (CREATE) lab was heavily involved in the development of a new robotic technology, the Tele-presence Robot Kit (or TeRK) (Nourbakhsh et al., 2007). TeRK

5. Robot Diaries

robots relied on a new and highly capable robot controller, the Qwerk, which was partially developed in the lab. As the controller was completed it was apparent that the potential applications were vast, and so we were faced with a challenge of finding interesting applications. Two members of the research team initially came up with the idea of an alternative to competition-centered robotics activities that would nevertheless inspire and excite participants to continue studying STEM topics. The two researchers, both women, came up with a description of an activity that was generated mostly based on what they thought they would have enjoyed when they were twelve years old. This description was included in the grant proposal that launched Robot Diaries, and it is instructive to quote it here:

A Robot Diary is a customizable robot designed to serve as a means of expression for middle school girls. The robot itself will be responsive to the personal diary entries of its user. Groups of middle school friends will create and use Robot Diaries as a unique means of exploring, expressing, and sharing emotion. Girls will keep daily private diaries about their lives with entries that can be interpreted and expressed by the robots. Girls will be able to share their diary entries and robot expressions with their friends through a web-based diary community. A girl can see how her friend is feeling by playing the friend's entry on her own robot. The actual content of the diary remains secret, but the emotional gist is shared publicly with the friend group.

In many ways, this description was a stab in the dark - it was our best guess of a technology that would be appealing to middle school girls, but it was based almost entirely on intuition. It would have been an enormous and fairly wasteful effort had we used this description as the basis for an engineering design process to create a robotic system or kit capable of expressing diary entries. Instead, we began our process by arranging a focus group of middle school girls to determine if the idea, in whole or in part, was appealing to girls in this age range.

5.3.1 Focus Group

The focus group consisted of seven girls, ages 11 to 14, and was conducted for two hours on a Monday evening in May 2006. In selecting this group of girls, the emphasis was to choose girls who were talkative and excited about science and technology and who we could safely assume would be eager to participate in early-project brainstorming. In this sense, the girls were not representative of the average middle school girl but were already somewhat biased towards STEM activities.

We began by surveying the girls about computer use in their homes, finding that:

- All of the girls had two or more computers in their homes

- All of the girls had high-speed Internet access at home (4 had wireless access)
- Five out of the seven girls said their family had a webpage
- Three out of the seven girls said they use MySpace or some type of “friends site”
- None of the girls contributed to an online blog or journal, nor did they keep paper-based diaries

We also engaged the girls in conversations about their communications and computer use habits. These conversations led us to determine that girls prefer non-disruptive to disruptive forms of communication; the girls disliked cell phones and chat applications, generally preferring email because you can “say more things”, because emails can be saved, and because email allows for photo and document attachments. The general pattern of email use for these girls is to check their email once or twice a day, soon after returning home or in the early evening after dinner. Girls would use the telephone, but only in situations in which a message was urgent and had to be received by a friend before the next school day.

A final portion of the focus group included reading three stories about sample prototypes of a robot diary. The prototypes were a clock, a turtle, or a flower - each of which could be actuated to express diary entries. Girls guessed right away that the stories were about some kind of robot or technology device. Even so, the girls were able to voice some likes and dislikes about the robot diary stories.

Things they liked:

- They had questions about the practicality of some of the items, but made some positive comments: “I like the clock... It’s nice to know what time it is,” or “Is it wireless wherever you go? Cause then I would get it cause I like to check my email.”
- Girls liked the animals, such as the turtle. They also liked the fact that it could capture emotion and share it with others. “I liked how you can talk to your friends and show your emotions a little bit.” Or “The fact that it can be done through a little technological flower, like the whole concept is kind of like neat.”

Things they disliked:

- The girls seemed to want useful or practical things. When asked what they would call the item they had just read about, one girl responded “SPA - stupid, pointless appliance.”

5. Robot Diaries

- Girls especially disliked the random spontaneity of the robot, such as the jumping up and down, shaking, and shining a light on you when you least expected it, because they thought it would be very annoying: “My little brother already does it.” This agrees with their general dislike of disruptive forms of communication.
- The girls also thought this robot sounded expensive, and this reason, as well as the lack of usefulness, deterred them from wanting to buy the robot. They estimated that it would cost 5 cents to make, but would range from \$50 to \$200 to buy.

Conducting the focus group made apparent that diaries, in the traditional sense in which we had thought of them, are not the appropriate avenues to engage these middle school girls. Nor do the girls seem to be that invested in real-time communication with one another when they are physically separate.

However, this also suggested an opportunity, first to re-think and re-pitch the diary. If a cloth-bound notebook is not where and how girls are recording their thoughts and plotting their actions, then where and how are they doing it? Second, this suggested an opportunity to think about and create a different kind of communication tool, one that is asynchronous and that does not need to support a significant amount of content.

An interesting distinction emerged between the notion of “communication” and “expression.” Expression, specifically personal expression, seems to be more salient than communication for these girls. Expression and communication seem to be bundled together in collaboration. When the girls work on expressive tasks together or when they work on them separately but then later share these expressions (such as stories) with one another, they are collaborating.

It’s important to emphasize that because the sample size was small, and because the girls were chosen precisely because they were extremely ambitious, they may not have been representative and generalizations are not possible. But the focus group was extremely successful and useful in providing initial insight into this demographic and the specific challenges and opportunities of Robot Diaries.

In addition to challenging our initial assumptions about the communication habits of middle-school girls, the focus group was effective in introducing the research team to the experience of working with middle-school girls. Unanimously, we were astonished by the girl’s high energy-levels and the effect this had on us as researchers. We found working with the girls for two hours to be tremendously exciting and also exhausting. In addition, we quickly came to realize how imperative it was to keep the girls on-task; tangential comments and momentary digressions quickly spiraled into consuming conversations. This realization had significant impact on our planning of summer workshop sessions. To facilitate our design and

learning objectives we realized we would need to provide a highly-directed (but flexible) agenda and a low participant to researcher ratio.

To sum up, the focus group was critical to our design trajectory. Our initial ideas revolved around robotic artifacts that would link to diary entries and that would react to the emotional content of the entry, or of linked friends' entries. The girls shot down this specific idea but did so in a way that caused us to believe that there was potential in the more general idea of a robotics activity built on creating expressive robots. The focus group also reinforced for us the importance of approaching the design problem through participatory design; our goals and initial idea were very broad, and so we next embarked on a number of participatory design workshops with the aim of narrowing the idea into a define-able educational activity.

5.4 Participatory Design Sequence

In 2006, we conducted two major participatory design workshops, one over the course of seven weeks in the summer, and the other over the course of ten weeks in the fall and winter. Each workshop taught our team important lessons about working with middle school girls, markedly affected the evolution of our curriculum, and brought us closer to a final technology and curricular design that was appealing to the girls. These workshops had a dual nature. They supported an iterative design cycle in which we could test software, curricular exercises, and the attractiveness of our technology designs with girls. They also were meant to serve as experiences which would both engender learning and motivate the girls to further study in science, technology, engineering and math (STEM) fields. Specifically, our program goals in conducting these workshops were:

Appropriate Exercises. Our program requires students to understand advanced technical concepts such as programming, prototyping, and connecting hardware. We sought to develop exercises which introduced these concepts in a non-threatening way, while providing the appropriate scaffolding to allow students to quickly use the concepts creatively.

Attractive Technology Framework. Determining the kit of parts from which students can build robots was crucial, as this kit defines the space of possible robot designs. Through the workshops we hoped to gain an understanding of what to place and what not to place in the kit, as well as an understanding of how much of the initial robot design should be provided.

Software Development. As part of our program, we needed software interfaces to allow for iconic programming of robots, as well as a messaging client which allows the sending of “roboticons”, or expressive sequences of robot mo-

5. Robot Diaries

tions.

Student Development. We wished to ensure that our student design partners got as much or more out of these activities as we did, and so one of our goals was to engender student growth through this experience. Specifically, we wanted students to leave the workshop understanding new technical concepts, with improved feelings of self-efficacy and confidence with regards to technology, and motivated to continue studying STEM topics.

Evaluation Scheme. In order to measure student development, we needed to develop a novel evaluation scheme to determine if students gain motivation, confidence, and content knowledge through participation in the workshops.

As participatory design workshops, we were not creating a design with learning goals, instruction, assessment, and tool all aligned prior to the workshop and then testing this design in the workshop. Instead, we were using these workshops to work with the girls to help us learn the background knowledge about the girls' interests and capabilities to allow us to create an aligned design later in the program (see section 5.5). As such, we had very loosely defined learning goals for these workshops: we wished the girls to show improved confidence and motivation with respect to further study in STEM subjects, and we wanted them to display improved knowledge of the robot building elements and skill at building (and in the case of the fall workshop, programming).

5.4.1 Curriculum Progression

We designed the Robot Diaries curriculum to follow an arc from simple to complex, familiar to new. We followed this guideline for the introduction of individual components, throughout the course of a workshop series, and for ourselves as we progressed from one series of workshops to another. We introduced new robotic components with a brief example followed by a free exploration time for the students. We then presented the students with a small design challenge to use the component in an expressive manner. Later the students combined the parts to create whole robots that could express emotions and communicate.

We learned from the girls as well. In the summer workshop we focused exclusively on mechanical design, with actuation performed by connecting motors and lights to batteries through switches. We learned which design challenges worked as effective learning experiences, and what materials worked well for ease of building, expressive features, and fun robot designs. This workshop was heavily focused on the goals of developing appropriate exercises and selecting materials. Then in the more extensive fall workshop we were able to introduce more complex technology and progress beyond mechanical design to introduce aspects of robot programming. Thus the fall workshop was primarily focused on the goals of developing a

software framework, creating an evaluation scheme, and evaluating student development to ensure that our approach was valid.

5.4.2 Summer Workshop

The primary purpose of the summer workshop was to engage a small group of girls in a series of participatory design activities that would lead toward the development of a working prototype of a “Robot Diary” for use in a more structured fall study. The summer workshop allowed the research team to work closely with a group of middle school girls over an extended period of time in direct, “hands-on” cooperative exploration of robotic technology. This, in turn, provided four important opportunities:

1. To experiment with a variety of participatory design activities and discover which were most effective and compelling for middle-school girls.
2. To develop research themes and observational measures.
3. To progress the concept (both form and function) of a “Robot Diary”.
4. To test evaluation instruments and collect initial qualitative data from interviews, observations, and the participatory design activities.

The Summer workshops took place at a public library in the girls’ community, one evening a week for two hours. In total there were six sessions over a seven-week period, with a one-week break between sessions two and three due to a holiday falling during that particular week. Participants were solicited via directed recruiting and word-of-mouth. The workshop was titled “Creative Expression with Robots” and the advertising pitch stated that “Participating girls will explore different types of technology and, working in groups of friends, will use those technologies to create short performances like skits, dances, and musical concerts.” The girls were thus a self-selecting group who already expressed an interest in technology.

While the attendance to the workshop was less than we hoped for, we were able to gather a small but consistent group of girls to attend the sessions and developed excellent rapport with them over the course of the workshops. A group of seven girls signed up for the summer workshops. The girls were all between 11 and 12 years old and middle-class. Four girls were signed up by the time the first session was held; two pairs of friends. One girl was ill so three girls attended the first session. After this session, one of the girls’ mothers offered to recruit additional friends who she believed would be interested. Seven girls attended session two. One girl attended session three. Three girls attended the remaining sessions. The

5. Robot Diaries

same three girls attended regularly. The research team who interacted with the girls consisted of four women and two men, four or five of whom attended any given session.

Session Arc

Each session followed approximately the same general arc. Broken into stages, the session arc was as follows (times are approximate):

1. review of previous week's homework (10 minutes)
2. bridge from the previous week's homework to this week's topic (5 minutes)
3. acting-games exploring the week's topic (20 minutes)
4. introduction to the week's featured technology (10 minutes)
5. open exploration with the week's technology (20 minutes)
6. directed design activities with the week's technology (20 minutes)
7. discussion, brainstorming, and review (10 minutes)
8. assignment of homework (10 minutes)

Each week we would describe a new concept centered around a core technology - for example, we would bring up light as made by LEDs, or motion as made by electric motors. We would begin with an acting game wherein girls would attempt to be expressive with a very constrained set of actions. We found acting games early in the evening was a successful way to engage girls in topical discussions and foster and promote collaboration among the girls. The games were physical in nature, requiring the girls to perform, usually with props. The games were designed to introduce the girls to general concepts without technology that would be addressed in more detail later in the session through technology.

After the acting games, the girls were introduced to the featured technology of the week. This usually took the form of two or more of the research team demonstrating the use of a given technology, such as motors or LEDs. The staging was an important aspect of this introduction. The girls would crowd around a table along with the researchers, as the demo took place. The tables themselves were usually strewn with wires, batteries, and various other technical supplies. This extremely informal staging was intended to promote a playfulness with the materials.

Immediately following the introduction to the technology, the girls were given time to explore and play with the featured technology without any restrictions or

expectations. Throughout this time, the researchers would work with the girls, answering questions and facilitating their play, but also allowing the girls space to truly experiment, to discover features and capabilities of the technology on their own, and to even allow them to make mistakes (for example, wiring motors incorrectly) in order to foster in the girls a sense of play and confidence in their ability to manipulate the technology.

After a break the girls would be presented with a directed design activity that would marry the general concepts of the acting game with the featured technology of the week. Like the free play, the researchers were present to support the girls in their process of discovery, invention, and design. In addition to working with the technology, during this time the girls were encouraged to work with the art supplies. Again, staging becomes important, so we made every effort to make the art materials ready at hand and compelling to use.



Figure 5.2: Robots made from craft materials

Materials

We used craft materials to construct the robot forms because these materials are familiar and approachable; see Figure 5.2 for some examples of such robots. The students used cardboard and foam board to build the structure of their robots. To this structure they attached aesthetic components like markers, felt, beads, bells and other items from a local craft store to transform the plain cardboard and give the robots personality.

Beneath the craft materials we used a variety of technical components to move

5. Robot Diaries

the robots. In the summer workshop students constructed simple circuits from alligator clips, AA battery packs, and switches to drive lights (LEDs) or motors. They also used radio transmitters designed for model airplanes to operate servo motors.

Evaluation Methods

We designed several evaluation instruments for the summer workshop: specifically, pre- and post-interviews with the girls and brief surveys after each workshop session. We also met immediately after each session to discuss the evening's events and record our insights from that session. Although we had too few regularly attending students in the summer workshop to provide evaluation results, the workshop afforded us a first opportunity to test some of these instruments, and both the evaluation instruments and post-session discussion aided us in formulating a set of lessons learned.

Lessons Learned

Our experiences running the workshops pointed to a number of design issues relevant to the overall project. These are described below, along with recommended modifications that were made in the fall workshop.

Providing specific and directed tasks is imperative It was imperative to provide participants with specific and directed tasks. These can be preceded by open play and exploration. But providing them with a specific and directed task was required to have them produce something. For example, experimenting with making sound is good, but then student should be directed to produce a specific kind of sound, or a sound to illustrate something, or to integrate sound into something else.

There is a tendency to work with what is “at-hand” The girls seemed to work almost exclusively with whatever art and craft materials were directly at hand. Never did one of the girls go randomly exploring through the arts and craft materials. Whatever materials were on top or in front of them were the materials they worked with. The only times they sifted through the materials was when searching for something specific.

If one of the goals is for the girls to use a diversity of arts and craft materials for their projects, then tactics should be employed to make as many materials as possible at-hand and to more vigorously promote their use. One obvious tactic would be to lay out the materials so they are all visible. In addition to making all

of the materials immediately visible, it would probably be helpful to introduce the materials, to actually go over each and every one of them. When new materials are brought in, (e.g. plastic balls for mounting lights) they too should be introduced. When this happened, as in the case of the plastic ball, they were more likely to be used.

Recommendations:

- Identify and describe the qualities of the different arts and craft materials.
- Create demonstrations that exhibit how different arts and craft materials can be used in conjunction with the different technical components.

Prototyping is a difficult concept and task The girls seem to have difficulty with the concept of prototyping. That is, they seem to have difficulty with presenting something and asking us to imagine it as something else. They want to build things as close as possible to how they want them to be, in both form and function. They seem reluctant to say “it’s supposed to do this, but for now it just does that.” This is a challenge because the design activities that we involve them with are essentially prototyping and not production activities. But the assumption that prototyping would come naturally to the participant was a flawed assumption on our part. Prototyping is an activity of abstraction that requires some knowledge of what the final system could be like - knowledge that the girls most likely did not possess.

Prototyping was additionally complicated by physical and dimensional requirements. Motors required mounting, long wires became tangled, and the construction of forms was difficult. The girls also had difficulty conceptualizing and constructing mechanisms for motion and support. This is to be expected. Mechanical engineering and industrial design are difficult. Recommendations:

- Introduce and develop the concept of prototyping through example.
- Structure the arc through the lessons so the girls witness the refinement of their concepts from rough sketches to finished products.
- Create a set of basic working mechanisms for motion.
- Create a set of basic support structures.

There is a predisposition to anthropomorphic and zoomorphic forms The girls seemed interested in developing fairly stereotypical robot forms: humanoids and dogs. The most successful forms, in terms both of the girls’ enjoyment in

5. Robot Diaries

making them and functionally, were a dog's head and a humanoid. This is not an issue so much as it raises a set of interesting questions, specifically:

- Are anthropomorphic and zoomorphic forms necessary (or at least extremely valuable) hooks for the girls to use and build on?
- Do these common forms limit their creativity?

Our concern with the predisposition to anthropomorphic and zoomorphic forms is that by immediately turning towards these forms, do the girls limit their range of possible exploration? That is, we struggle with the question of how creative is it to imitate the common forms of robots from the media? However, we also want to be honest and reflective and admit that it is our interest in more abstract and ambient robotics that is the source of these questions. If anthropomorphic and zoomorphic forms are those which are most compelling to the girls, they should be wholly supported.

Recommendations:

- Provide examples from art and design of alternate forms as well as a broad range of anthropomorphic and zoomorphic forms.
- Encourage the girls to personalize and thus differentiate their anthropomorphic and zoomorphic forms.

Not all materials are equally valuable As this was our first time working with the girls, we did not know which materials would capture their interest and which would not. We provided them with a large variety of arts and crafts materials, and gave them lessons on using small and large motors, servos, ways to make physical sound, and LEDs. We found that girls used some materials much more frequently when creating their robots, specifically:

- Servo motors were conceptually easier to understand and physically easier to use than continuous rotation motors. Small, fast DC motors were almost useless, as they move too quickly and don't have enough torque to move significant masses. Larger continuous rotation motors did get used, but were harder for the girls to install and use than servos.
- Vibration motors were a big hit, especially when combined with small bells to make sounds.
- Larger bells and chimes were generally not used because it was difficult to design a mechanism to cause them to emit sound.

- LEDs were popular, and useful for expressing emotion (glowing eyes that could be red for angry or green for happy). We provided some materials for optical effects, such as wax, colored transparencies, and mirrored surfaces - these were not used.
- The girls like hot glue so much that they began to use it even in situations where other fasteners could be placed faster and work better. We strongly recommend that hot glue be incorporated, but that action be taken if the girls become overly dependent on it as a building material.
- Fake fur was the most popular “skin” for the robots, although this may be due to the large number of zoomorphic projects.

The importance of community involvement School-based educators learned long ago that family involvement was an important part of a child’s education, and we believe the same to be true for informal learning experiences (Crowley and Jacobs, 2002). Over the course of the summer, we observed numerous instances of girls inviting friends and family into their Robot Diaries experience. This included girls showing off their creations when their parents came to pick them up, or even showing off our creations (one girl was very excited about showing her father the TeRK robot we had brought in). In one instance, a girl’s friend was in the library on the same day as our session, so she invited her friend in to see what she had made.

We believe these instances of community involvement are important learning moments, as they allow children both to learn from others and to share what they have learned. This sharing of knowledge helps children to solidify their own position as a knowledgeable individual in the community.

As creators of an informal learning experience, we should embrace any and all opportunities for community involvement. Over the summer, we did this by inviting the girls’ families to the last session, to give the girls a chance to show off their creations. One child invited her parents and three siblings to attend, and then proudly demonstrated both her robot and her knowledge of the technology she had been using over the summer.

Recommendations:

- Involve family and friends in Robot Diaries activities.
- Provide girls with structured opportunities to involve members of their community (e.g., activities that require them to work with a family member).

5. Robot Diaries

Making room for ‘Girl Culture’ This lesson has to do with connecting our activities to the cultural world that middle school girls inhabit. As a number of our research team learned while running a previous, unrelated project with this age group, middle school girls respond well to educational experiences that reflect their cultural interests. This is why we sometimes asked girls to bring in pop-culture products to use in session activities. For example, in order to introduce the girls to LEDs, we set up an activity where they could choreograph the LEDs to music and invited them to bring in their own music. One of the girls brought in a Madonna CD, which we played while they were creating their light show. The girls enjoyed listening to the music while working, and created a light show that was relatively well synchronized to the familiar music. In the future, we should continue to make room in our session activities for the things that interest girls.

5.4.3 One-Day Workshops

We distilled the activities developed during the summer for two one-day workshops in the early fall. We held these workshops in collaboration with C-MITES, a university-affiliated organization that provides educational programming for academically talented elementary and secondary school students. Through C-MITES, we obtained significantly higher attendance numbers than during the summer - 15 girls attended the beginner workshop and 12 attended the intermediate. These workshops served as a chance to observe a larger audience of girls using craft materials to create communicative robots.

Beginner Workshop

The beginner workshop was a direct distillation of the summer curriculum. We took the lessons learned from the summer, especially regarding which materials and activities worked, and shrunk the workshop into a six-hour session focused on the design of an expressive robot. We began by introducing the goal of the workshop, and then quickly taught girls how to integrate motors, servos, and LEDs into their robots. As in the summer we allowed some time for free exploration with the materials, before turning most of the afternoon over to the directed design activity.

Intermediate Workshop

In the intermediate workshop we sought to test some activities we were working on to introduce programming in our ten-week fall workshop. For this session, we introduced the same set of basic robotic components (LEDs, motors, servos), but

also introduced a microcontroller and programming environment. We followed the same basic structure as the beginner workshop (introducing components, allowing for free play, and then ending with a directed design task). The intermediate workshop drew girls from the beginner workshop as well as girls with other prior robotics experience.

5.4.4 Fall Workshop

Our most extensive workshop was held from early November 2006 to mid-January 2007 with a group of 8 girls from a private, university-affiliated middle school. Sessions were two hours long and held immediately after school. Referring back to the five goals stated at the beginning of the Participatory Design section, the fall workshop was focused on software development, adding programmability to the technology framework, creating and testing an evaluation scheme, and using this scheme to measure student development and ensure that our approach was working.

Curriculum Progression

We began the workshop in much the same way as our summer workshops, gradually introducing the students to important robotic technologies over the first four sessions. A major difference was the early introduction of the Qwerk (Nourbakhsh et al., 2007), a controller which allows the girls to create programs which actuate motors, servos, and LEDs. In addition, we began to introduce the girls to “Doodlechat” (see Figure 5.3), a combined chat and collaborative doodling program that was designed by our group and later used to control the robots. Within Doodlechat girls could discuss in real time from home as well as draw on a community window pane - thus they could collaboratively draw a robot idea. Doodlechat enabled the students and researchers to communicate as part of a private, informal, online community between workshop sessions.

Once students had the foundational robotics knowledge to make cogent design decisions, we began a series of participatory design exercises which yielded a final robot design in session six. This design was selected by the girls from a set of five after a group discussion. The girls then each constructed a variant on the design with the same underlying morphology but widely varying cosmetic touches (see Figure 5.4). Once the final robots were constructed, the girls took their robots home each week to experiment with programming the robots in a novel software framework which was refined weekly based on the girls’ feedback.

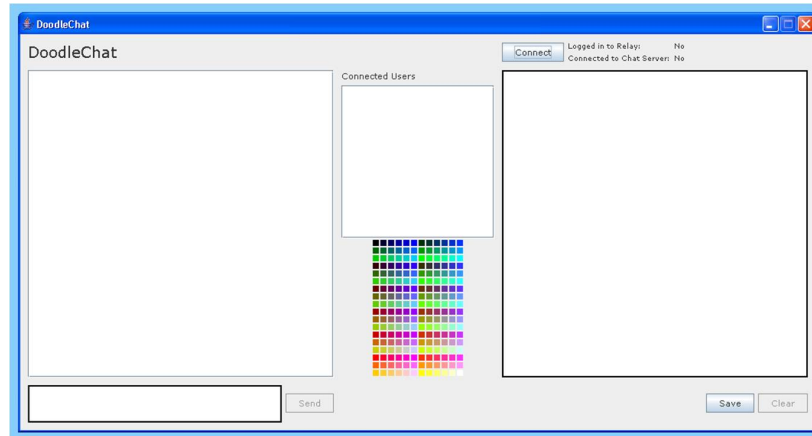


Figure 5.3: The Doodlechat interface

Materials and Software

Building materials were very similar to the summer workshop: cardboard and foam board for constructing structures, a wide range of arts and crafts materials to create aesthetic designs as well as sounds and motion of small components (like eye-brows, noses, etc), and LEDs, motors, vibration motors, servos for movement. Unlike the summer workshop, students were also introduced to the Qwerk (Figure 5.5) to allow them to program their robots.

The Qwerk was a commercially available product that was selected primarily because it had a high upper limit of supported components and because the research team had experience using and developing for it. It was also capable of being wirelessly tethered to a computer - a mode in which programs execute on a host computer, with commands being sent wirelessly to the Qwerk. We ran the Qwerk in this mode, allowing us to build a software environment for the girls to program their robots that existed entirely on a host computer. Doing so allowed us to develop the software much more rapidly, which was important as we were adjusting the software in real-time based on feedback from the girls.

The software itself consisted of three separate programs; the RuR (see Figure 5.6), the Express-O-Matic (see Figure 5.7), and the Roboticon Messenger (see Figure 5.8). Together, these programs enabled programming of the robots in an easy-to-learn format and sharing of finished programs. The RuR allowed direct remote control of the robot through on-screen sliders and buttons (for example, one could set the speed of a motor with a slider). It was also possible to save the state of the sliders and buttons into an *Expression*. The Express-O-Matic allowed

5.4. Participatory Design Sequence



Figure 5.4: The chosen design and girls' instantiations of the design

one to chain together expressions, with a customizable delay between expressions. We referred to these chains of expressions as *Sequences*. A sequence, essentially, is a script of motions for the robot to execute - thus it is possible to animate the robots through a series of expressions. Finally, the Roboticon Messenger allowed sharing of both expressions and sequences. Like Doodlechat, Roboticon Messenger included a chat window. Additionally, girls were able to post sequences or expressions to a public space, or send them to individual girls. We referred to publicly posted sequences or expressions as *Roboticons*, a take-off on the emoticons that symbolize mood in online chat conversations. Girls could play Roboticons, or incorporate them into their own programs, thus mixing together elements of their own programs with those of other's.

The software was designed and written by the research team during the fall workshop with feedback from the girls. We introduced each of the three programs during the pedagogically appropriate week - so the girls used the RuR for a week to

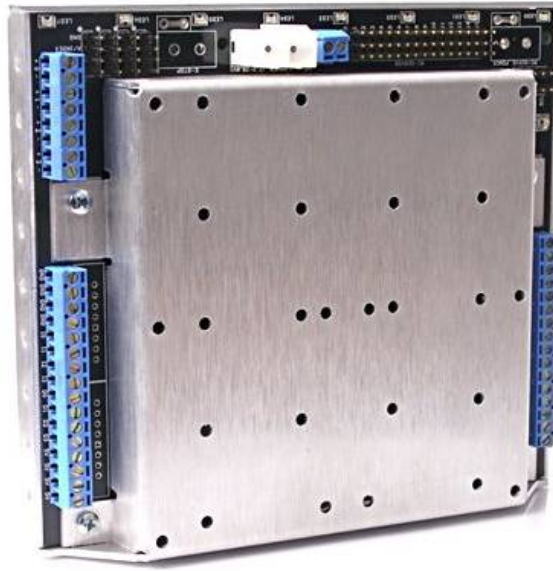


Figure 5.5: The Qwerk microcontroller

allow them to move their robots. Then they learned to use Express-O-Matic so that they could script sequences of motions, and finally they used the Roboticon Messenger to share sequences. The girls spent a week with each program beta-testing the program during the workshop and at home. They would then return the next week and provide comments to us. In some cases we were able to take their comments and revise the software program to provide them with an improved version by the next week. In this way, we were able to quickly improve the programming environment in a way that was driven by feedback from the girls.

5.4. Participatory Design Sequence

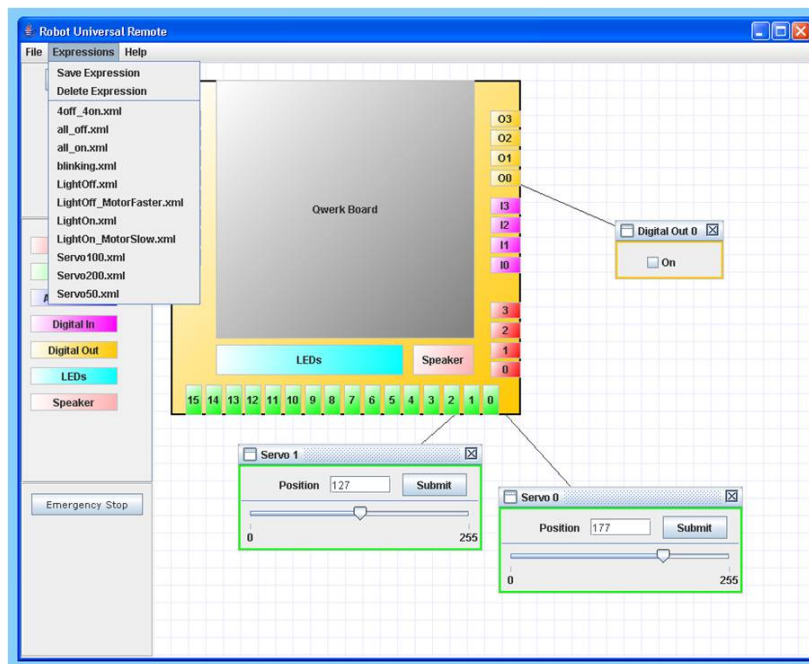


Figure 5.6: The RuR software program

5. Robot Diaries

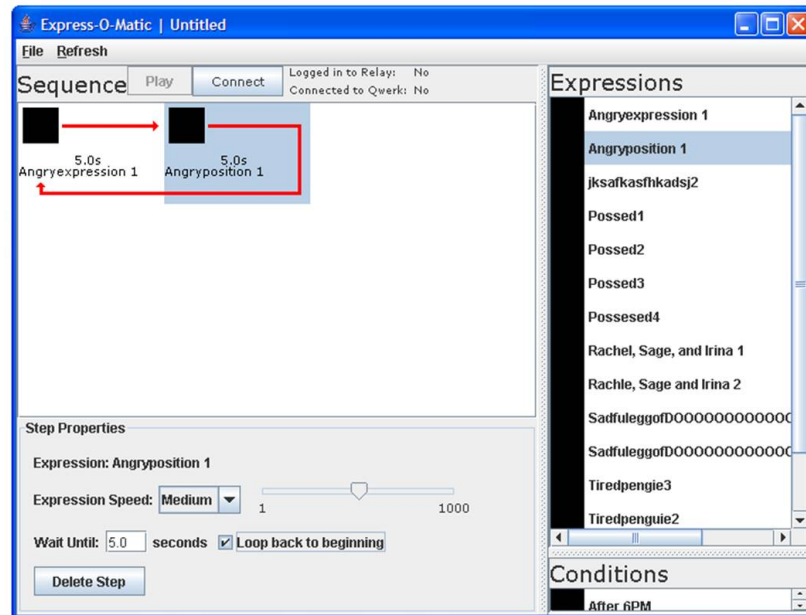


Figure 5.7: The Express-O-Matic software program

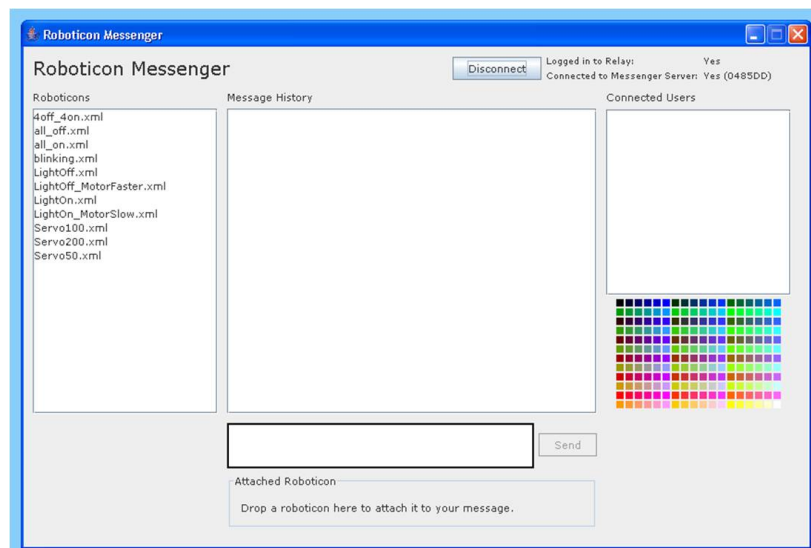


Figure 5.8: The Roboticon Messenger software program

Evaluation Methods

Our research used methods drawn from the learning sciences and interaction design. Collected data included interviews with participants and their parents and electronic activity logs.

Participants were interviewed individually at the beginning of the workshop (pre), and again at the end of the workshop (post). Interviews included questions about relevant declarative knowledge (e.g., identify and provide a definition for relevant parts, such as sensors and motors) and designed systems (e.g., examine an electronic toy and describe its components/how it works). Participants were also asked to imagine how they might build a new system (an alarm) using a fixed set of components (a battery pack, alligator clips, switch, LED, servo, and sensor). Pre-interviews ranged in length from 16 to 32 minutes. Post-interviews ranged in length from 21 to 45 minutes.

Parents were interviewed in their homes at the beginning of the workshop and again after the workshop was completed. In the pre-interview, parents were asked about their child's previous experience with robotics and related technologies and about the family's activities related to science and technology. Post-interviews mainly focused on parents' impressions of the workshop and what their child gained from participation.

Electronic activity logs were derived from girls' contributions to the Doodlechat and Roboticon Messenger online communities. From these logs we could see how frequently girls were logging in, contributing through chat, and contributing through doodling or posting Roboticons.

Evaluation Results

Girl-Focused Parent interviews conducted at the start of the workshop revealed that children in the workshop group were generally interested in using and/or exploring technology. A subset had attempted to participate in other technology workshops, but these experiences were not always positive. One parent described her daughter's experience in the following way:

She has been fascinated by robotics for a long time every time we sign up for one of those [technology] camps we'll get there on the first day and it's all obnoxious little boys and she just goes, 'never mind'.

Another parent provided the following explanation for why she thought her daughter would enjoy Robot Diaries:

The problem with some of those [technology workshops] was that

5. Robot Diaries

there were often more boys there than girls, and so she didn't feel quite as comfortable. So that's why this [program] looked more interesting.

As comments from this small sample of parents suggest, existing resources may not be fully serving the needs of middle school girls interested in technology exploration. These parents point to the male-dominated culture of these activities as being particularly problematic for their daughters. One of the girls echoed this sentiment when she commented that her school's competition-oriented FIRST LEGO League team, which she had joined briefly, was "more geared towards boys."

Engagement In Robot Diaries, we tried to engage participants using a 'social narrative' approach, which enabled girls to engage with their robots in a narrative way. A quick look at the robots created during the workshop suggests that this occurred for most participants. Six out of the eight participants named their robots. Nearly all of the participants personalized their robots through decoration, and a few created additional narrative elements such as accessories for the robot (for example, one student made an 'iPod' out of foam for her robot and another made a guitar). Another participant created a back-story to explain her robot's appearance:

Dear old elderly professor Bob suffered from a head injury when he ran into an Eskimo so now he has a band-aid on his head. And he's a professor so he has to dress up. The tie. And he has certain vision problems so he wears a 'monocule' [monocle].

An examination of the electronic activity logs showed that all eight students participated to some degree in the Robot Diaries online community from home. Each girl posted messages to the custom messaging program used during the workshop. Half of the students posted robot programs, or Roboticons, to share with the Robot Diaries community. An example of one such Roboticon was a program expressing sadness. The robot's eyes were lit by green LEDs as the robot's arms rose to cover the eyes and then slowly lowered. Network problems contributed to at least some of the remaining girls' inability to share Roboticons from home.

Lastly, informal observations of the girls' behavior at the workshops showed them to be engaged by the workshop content. Frequently one or more girls would not leave when the workshop ended. On one occasion one of the girls stayed a full hour after the two-hour session officially ended. It was not uncommon for girls to stay 15-30 minutes after the session to continue working on their robots.

Learning Learning was formally measured through analysis of pre- and post-workshop interviews with participants. Analysis consisted of coding interview responses for correctness, comprehensiveness, and level of sophistication. In order to determine the reliability of our coding scheme, two raters coded three of the 16 transcripts. Inter-rater reliability was calculated at over 83%. Coding disagreements were resolved through discussion.

Two main types of knowledge were assessed: declarative knowledge and knowledge of technical systems.

The majority of participants showed gains in declarative knowledge. On average, participants were able to identify and correctly label four (SD = 1.31) out of six robotic components at pre-test and 5.9 (SD = 0.35) at post-test. A paired t-test indicates this increase is statistically significant, $t(7) = -4.26$, $p < 0.05$. Additionally, there was a significant increase in the comprehensiveness and accuracy of participants' descriptions of a sensor and electric motor, as indicated by sign tests. Six out of eight participants showed improvement in their descriptions of a sensor at post-test (the other two participants were already knowledgeable at the start of the workshop), $p < 0.05$. Seven out of eight participants showed improvement in their descriptions of an electric motor, $p < 0.05$.

The American Association for the Advancement of Science (AAAS) recognizes knowledge of technical systems as an important component of scientific literacy for children in grades 6 through 8. The following benchmark is included in their Atlas of Scientific Literacy (AAAS, 2007), page 57:

“Analyze simple mechanical devices and describe what the various parts are for; estimate what the effect of making a change in one part of a device would have on the device as a whole.”

We believe that this type of knowledge may be critically useful for engagement in the design process, and movement towards technological fluency. We assessed knowledge of technical systems in two ways. First, we presented participants with an electronic toy (a Furby, a Meowchi, or an iDog), and asked them to explain what parts were inside the toy, and how it worked. At post-test, all seven children were able to identify parts from the workshop (e.g., servos or LED's) in the electronic toys. Additionally, six out of seven children were able to provide more sophisticated explanations of how the toy worked at post-test (one participant showed no change). The increase in sophistication of explanation was significant, $p < 0.05$ by sign test.

Parent Viewpoints One of the most important reasons to work on technical literacy in informal contexts is that we have an opportunity to change how parents

5. Robot Diaries

see their children. This is crucial to long-term success of any technology enrichment program. We know from prior work in museums that even the most educated parents often fall back on traditional stereotypes when they think about what girls might be interested in learning about (Crowley et al. 2001). Because parents play such a large role in orchestrating the educational enrichment experiences of middle school students, effective programs should change how parents view their children's interests, competence, and potential for success in technology. Our preliminary evidence suggests that we did just that.

In interviews conducted after the conclusion of the workshop, a number of parents commented on changes they had observed in their daughters. Two parents commented that the workshop led to gains in interest and engagement in robotics and technology for their daughters. For example:

I think [she] learned a lot from and just got much more adept and fluent at doing that kind of thing and I think she just sort of just enjoyed the - the whole concept of doing it, and she got bitten by a bug, I mean she wants - she wants to do more and so now we're sort of trying to figure out how you do that - what else there is besides LEGO Mindstorms.

Two parents commented that the workshop helped to broaden their daughter's perspective about technology. For example, when asked what he felt his daughter gained from the workshop, one parent responded, "I think she probably got an awareness that computers and technology are in more parts of her life than she realized."

Additionally, three parents commented that the workshop helped to increase their daughters' comfort and confidence with technology.

I would say there was a - a pulling together of things that she probably already knew, but hadn't really combined all the skills in the way that she did, and - just a higher comfort level. In some of the skills that she probably already had, I mean she'd downloaded things before, and she'd - you know, interacted with websites and whatever before, but she's just much, much faster at it now, and less afraid when she hits a glitch of her - that's - the big thing is that when she hits a glitch, she's like not afraid to kinda, fix it as opposed to 'eek!'

Finally, six parents commented that the workshop and associated activities led to knowledge gains for their daughters.

Well I think she got a lot out of it. I know she really enjoyed it, and I know she learned, I mean I know before this she had no idea about,

you know, all the things she would mention, words that I have no idea about. You know, like how to built the robot, how to pr- control it via the computer, how to enhance the robot. Like she was very excited, when she'd come home with the little - the little white and black [servo] boxes. And be like, I'm adding a new one to this, 'cause I want it to do that.

In summary, parents commented on changes in their daughters' knowledge, confidence, interest, and perspectives on technology. Each of these areas of change represents an important part of girls' movement towards fluent use of technology. As girls come to see technology as more relevant to their lives or think critically about technology more often, they take an important first step towards technology literacy and a deeper engagement with technology.

5.4.5 Validation of the Approach

Our evaluation and analysis of the 2006 pilots provided us with some preliminary data to validate our curricular approach. Specifically, we were able to document participants' engagement in the program, their knowledge gains, and their parents' viewpoints about how they had changed as a result of participating in the workshop. We also found that some of our curricular activities were scalable, as demonstrated by the fact that C-Mites has continued teaching one of our one-day workshops without our support.

In summary, Robot Diaries appeared to have a positive impact on its participants. The eight girls enrolled in the fall workshop were engaged by the social narrative approach to robotics and actively participated in the community formed by the workshop. Participants also gained valuable technical knowledge. This evidence suggests that the Robot Diaries program is moving participants in a promising direction. However, given the pilot nature of the first phase of the program and the small number of participants, the findings are incomplete. With the approach validated by our participant results, we began a design process oriented ultimately towards creating disseminable curriculum, materials, and software.

5.5 Designing for Dissemination

The goal of the Robot Diaries program is the creation of an activity that is a viable alternative to robotics competitions. This goal requires that the activity can be led by educators who are not members of the research team. At the beginning of the second design phase of the Robot Diaries program we shifted our focus to creating a program that could be taught by external educators. This goal suggested two

5. Robot Diaries

other requirements: a documented curriculum, and a set of supporting hardware and software tools that did not require a highly technical background to use.

We approached the design problem of creating a new program through the principle of alignment. As we were creating an entirely new educational activity, we began by specifying the learning goals of the activity, and then derived a program theme, assessments, and curriculum from those goals. While designing the curriculum, we also created new hardware and software tools that aligned with the goals and curriculum of the program. The rest of this chapter is concerned with this design, pilot, and evaluation of the first revision of this curriculum and associated tools.

5.5.1 Learning Goals

At the outset of our redesign of the curriculum and tools we identified the educational goals that we were aiming for in the Robot Diaries program. We created a core learning goal that we felt would support our program goals. *The ultimate learning goal of Robot Diaries is to enable girls to engage with, change, customize or otherwise become fluent with the technology in their lives.* From this goal, we derived a number of others - the full list of which can be seen in Table 5.1. We began the process of identifying goals by deciding on a central theme for our curriculum.

Curricular Theme

We thought it important to base our curriculum around a core theme to provide consistency to the instructional sequence. We rooted our curriculum on the notion of the engineering design process; the steps of planning, prototyping, testing & evaluating, troubleshooting, and documenting. We also stressed the importance of iterating through these steps multiple times. Based on our earlier workshops, we felt that this process could best be explored by students through the design of a robot focused on communication and expressiveness. The idea for the theme came out of our experience with how girls engaged in design in the participatory workshops. Girls engaged in what we labeled “organic design” - they would build, often with little planning, and when a component of their creation did not work, they added to their creation to try to work around this failure. We rarely saw a girl remove a non-working part from their robot, and there seemed to be a general resistance to “starting over”. Hence, robots would grow in an organic, almost stream-of-consciousness way.

We discussed within the group whether or not the theme could align well with our stated overarching learning goal. The design process is inherently about creat-

ing new technological artifacts, and our central goal is to enable girls to become capable of changing or customizing the technology in their lives. To us it seemed like teaching design process aligned with this goal very well, as customizing technology would rely on many of the same steps (prototyping, evaluating, troubleshooting) as design. Once the theme was selected, it became fairly easy to derive a number of lower-level learning goals; we began this process of derivation by creating three categories into which goals could be placed.

lower-level Learning Goals

We used a categorization system from Understanding by Design (Wiggins and McTighe, 2005), an instructional design textbook, to help us structure and discover learning goals. This system divides goals into dispositional, knowledge, and skills:

- **Dispositional goals.** The dispositional goals of the program are to help girls see technology as interesting and deeply relevant to their lives, to help motivate their continued engagement and exploration of technology, and to provide them with confidence in their own ability to create with, modify, or troubleshoot the technology in their lives.
- **Skills goals.** The central skills goals are focused on design and creation; although some of these goals are specific to the Robot Diaries technological context, many of them are in line with the ITEA Standards for Technological Literacy (ITEA, 2000). With respect to design, girls will understand and be able to engage in an iterative design process, including prototyping, evaluating, troubleshooting, and documenting. They will understand the idea of trade-offs and constraints and be able to identify them for specific designs they create. In terms of creation, girls will receive a detailed understanding of the robotic and structural kit components to allow them to properly identify components which meet their needs. Girls will also learn to use a number of tools to construct their designs, as well as a graphical programming language to animate their creations.
- **Knowledge goals.** Our knowledge goals were derived mostly from our desired skills - for example, we wished for girls to engage in the design process (a skill), and so one of our goals is to teach the steps of the design process. We also formulated knowledge goals around facts that are important for girls to know in order to create with the specific media of the program (LEDs, motors, servos, a microcontroller).

5. Robot Diaries

We created a list of lower-level learning goals to align with specific curricular activities and placed this list (see Table 5.1) at the beginning of each part of the curriculum. In many cases, goals are only indirectly supporting this process. For example, many of the Technical Knowledge Application goals are necessary for girls to learn in order to work with the specific materials that are used in Robot Diaries; it is important that they learn these skills, because without them they would be unable to learn more directly supportive goals.

The final categorization system in Table 5.1 has been modified somewhat to be more specific to the curriculum's theme. Even so, these categories still categorize goals by knowledge and skills: Technical Knowledge Application (indirect skills), Design Process Knowledge (direct knowledge), and Design Process Application (direct skills).

Dispositional Goals Dispositional goals are not present in Table 5.1 because we did not align specific curricular activities to these goals. Instead, we viewed dispositional goals as learning goals that should be attained through completion of the entire program. The dispositional learning goals we identified were:

- **Engagement and Motivation.** The program will help girls see technology as interesting and deeply relevant to their lives, and help motivate their continued engagement and exploration of technology.
- **Confidence.** The program will help girls to develop confidence in their own ability to create with, modify, or troubleshoot the technology in their lives.
- **Worldview.** 1. Girls will have incorporated the idea that there is no one right answer to solving many real-world problems. 2. Girls will see that there are many ways to use technology (including in situations which do not appear to be 'technological'), and that some of these constitute creative ways to use and think about technology.

5.5.2 Curriculum

We began a curriculum design once we had completed the first draft of our learning goals. The curriculum, the entirety of which is in appendix A, was influenced by the learning and program goals, as well as by a number of lessons learned from our previous participatory design workshops and from speaking with potential partners. We begin by discussing these initial considerations, and move on from there to discuss the theme of the curriculum and its structure.

Technical knowledge application
Demonstrate the use of hand tools.
Attach servos, vibration motors, and LEDs to a controller.
Create short programs using a graphical programming language.
Design process knowledge
Describe all of the steps in the design process.
Explain why design is an important part of the creative process.
Explain why design is an iterative/experimental process.
Design process application
<i>Planning</i>
Create or modify a technological design to meet a need.
Formulate design goals.
Identify design constraints.
Identify available resources.
Identify the limitations of an available resource.
Draw a sketch or diagram of a technological design.
Create a design which utilizes mechanisms X, Y, and/or Z
Modify a design or implementation based on design constraints.
<i>Prototyping</i>
Assemble the resources needed in order to implement a design.
Construct a physical prototype of a technological design.
<i>Evaluating designs</i>
Assess how well a technological design meets a need.
Judge the benefits and drawbacks of a design modification with respect to design constraints and goals.
<i>Troubleshooting</i>
Identify problems with a technological design.
Devise solutions to the problems with a technological design.
<i>Documenting designs</i>
Describe a technological design using writing or photography.

Table 5.1: Learning goals

Considerations

Teachable by Partners A major program goal for the new curriculum was that it should be teachable by educators that we train and with whom we partner. The curriculum we had developed for our earlier participatory design workshops were designed to allow us and the participants to jointly explore and discover a com-

5. Robot Diaries

elling robotics experience for middle school girls. This earlier curriculum focused on answering questions such as “What type of robot do middle school girls want to create?”, “What form should the robot take?”, and “What materials are the easiest to use and result in the most compelling robots?” This participatory design, discovery-based workshop curriculum was both labor- and materials-intensive, as well as being very open-ended and flexible. Unfortunately, these characteristics made the participatory design curriculum poorly suited to dissemination and training of external educators.

Materials It became apparent during our numerous participatory design sessions that the materials provided can impact the students’ creativity. Both the specific variety of materials made available and the layout and arrangement of materials are very important. Students have a strong tendency to use the materials that are ‘at hand’. Student creativity can be increased by providing a variety of arts and crafts materials and making the variety of materials readily apparent.

Pacing of Activities A number of our lessons learned concern the timing and pacing of curriculum activities. For example, we learned that middle school students require significantly more time than adults to perform manual skills and have tried to pace our activities accordingly (e.g., attaching components to a microcontroller using a screwdriver took students roughly four times longer than adults).

Modularity Based on our conversations with several potential partner venues, we learned that the amount of time dedicated to any given program varied from site to site. Girl Scouts might spend roughly six hours working on a badge. Groups such as the People Always Learning Something (PALS) home school group and Sarah Heinz House Boys and Girls Club have ten- to twelve-week classes. In order to make our curriculum applicable to as many groups as possible, we decided to take a modular approach. We developed a core set of activities as well as optional activities. In our curriculum we specify and study which learning goals are targeted by each activity so that teachers can see which learning goals they can accomplish given their own time requirements.

Instructional Sequence

We designed a three-part curriculum: the first part’s goals were to educate girls how to use the materials of Robot Diaries (both robotic and craft) and the purpose of the program (to build an expressive robot). The second part was an introduction to the design process, and a run-through of an initial, constrained design cycle.

The third part, which lasted longer than both previous parts combined, was a set of iterative design cycles followed by use of the robots in communication-focused group activities. We briefly run through the activities of each part here.

Activity	Estimated Time
Session 1	
Introduction to Arts & Bots	10 minutes
Survey	15 minutes
Group Activity: Expressive Charades	20 minutes
Introduction to Motors, LEDs, and Circuits	10 minutes
Group Activity: Hypnotic Eyes	10 minutes
Introduction to Hummingbird and Software	25 minutes
Group Activity: Make a noise	15 minutes
Session 2	
Home Activity 1: Scavenger Hunt	5 minutes
Home Activity Follow-Up: Scavenger Hunt	10 minutes
Introduction to Express-O-Matic	20 minutes
Group Activity: Foley Artist	40 minutes
Choose a User Name	15 minutes

Table 5.2: Sequence of activities for first curricular module

Part One: Introduction Table 5.2 shows the sequence of activities for the first part of the curriculum. These tables have been taken directly from the curriculum documents provided to educators. We broke up the activities into sessions lasting roughly two hours. This was a somewhat arbitrary choice, as in our pilots sessions were anywhere from 1 to 3 hours long. Session length has minimal effects on the timing of the activities, with the exception of take-home activities, which were most logically described at the end of a session.

We began the first session with an introduction to Arts & Bots, the name that we gave this iteration of the Robot Diaries program. In the introduction we discussed the goal of the program from the student’s perspective - that is that Arts & Bots is about designing robots to be expressive and communicative. We then allocated some time for a research survey of the students to establish baseline interest, motivation, and confidence. The next activity was a game of expressive charades, in which girls pretended that they could only move in certain ways (to demonstrate the difficulty of making a mechanical robot with limited movement into an expressive device). This pattern of introducing a concept and then highlighting or practicing the concept in an activity was heavily used in the curriculum.

5. Robot Diaries

After charades, we taught girls about basic circuitry, LEDs, and motors. They were then given an activity in which they made “hypnotic eyes”, or spinning, patterned discs mounted on motors. They directly hooked up switches, motors, and LEDs to batteries to make these creations. We then introduced the Hummingbird, a microcontroller created as part of the concurrent design, that they would use to control their robots. We described how to hook up LEDs, motors, and servos to the Hummingbird, and also showed a software environment, the RuR, for controlling the Hummingbird. The RuR is a screen with sliders for each of the ports on the Hummingbird, such that each port can be directly adjusted.

The previous activity provided a strong rhetorical reason for using the Hummingbird: by directly creating circuits girls quickly became aware how little control they had over the motors and LEDs when all they could do is turn them on or off with a switch. With the Hummingbird, we demonstrated that you could speed up or slow down motors, rotate them backwards, and fade LEDs. We had another group activity in which the girls were tasked to use the Hummingbird, software, and attached motors or servos to make a noise (bells and other materials were provided). Girls were then given a scavenger hunt to complete at home, with the task to find objects in their home that have one or more of the following: buttons/switches, motors, LEDs, something programmable, and something robotic.

In session two, we began by discussing the results of girls’ scavenger hunts and then immediately launched into the Express-O-Matic, a software environment we created to allow chaining together of “expressions” or robot states. An expression is a state of ports defined by the RuR. For example, if in the RuR two motors are set to full forward and two LEDs are half on, this can be saved as one expression. Another expression could be two motors turned off and two LEDs full on. The Express-O-Matic would allow the chaining of one expression to the next, with a customizable delay between them, so as to create a simple sequential program. Any number of expressions can be chained together, and a loop can be made by linking the last expression to the first. Once again we limited the time spent on explanation and instead focused on providing girls with an activity to practice with the Express-O-Matic. In this case they were tasked to create mechanical devices that could do the job of a foley artist. Each girl was given a 20-second clip from Harry Potter and was asked to re-imagine the sound effects by building a robot capable of making noise. The use of the Express-O-Matic was in ensuring that robot noises occurred at the correct time in the clip.

After this activity, girls were introduced to a website tailored to allow documentation of their experiences and of their robot designs. They were asked to choose a user name for the website.

The first part of the curriculum does not yet involve the robot design that will be the main theme of the rest of the curriculum. It does, however, provide girls

5.5. Designing for Dissemination

with experience working with the hardware and software of the program, and it provides some initial experience designing and building small robots.

Activity	Estimated Time
Session 3	
The Design Process	5 minutes
Design / Plan	15 minutes
Build / Implement Designs	60 minutes
Group Activity: Robot Dance Party	30 minutes
Documenting Designs	10 minutes
Session 4	
Documenting Designs: Sharing Your Design Journey	30 minutes
Group Activity: Show and Tell	20 minutes
Group Activity: Improv	10 minutes
Group Activity: Charades	30 minutes
Introduction to Messenger	15 minutes
Individual Activity 2: Share a Story	5 minutes

Table 5.3: Sequence of activities for part two

Part Two: Design We began part two of the curriculum by explicitly explaining the steps of the design process and provided some examples of how engineers do design. We then gave girls fifteen minutes to design an expressive robot using the materials they had learned about in the first two sessions. They were not allowed to begin building during this planning phase. One important choice in shaping the Robot Diaries experience was how to guide and constrain the initial robot design task. How directed or open should the design task be in order to maximize student creativity, learning, engagement, and enjoyment? We chose to make the task fairly directed but allow room for flexibility. The following excerpt is from the initial design task instructions:

“Today you are going to build an expressive robot that has a body, **2 servos, and 2 LEDs**. The servos can control arms, legs, ears, antennae, wings, fins, heads, tails, and so on. The LEDs can be used in the eyes, ears, nose, antennae, or some other part of your robot.”

Terms like body, arms, and eyes set the tone for the robot to be modeled after living creatures. This choice is in line with the types of robots that the girls in our participatory design programs found appealing. A living creature is also a good model for making a robot that can express emotions and moods. We hope that these instructions along with sample robot pictures will help students think outside

5. Robot Diaries

the often common robots-as-cars model.

Another advantage to this approach over a more open approach is that it gives direction to the first design and building task by narrowing the scope of possibilities. This decreases the amount of time necessary to settle on and implement a design. By setting boundaries on the initial design task, we assert that students can more quickly reach a functioning robot which can be programmed. Thus we walk participants through the complete design cycle, including using and evaluating their designs, before having them iterate on the designs.

We allotted 60 minutes to building the initial robot, although we know from experience that students could happily spend more time building and decorating their robots. We made a conscious choice to limit the amount of personalization students are able to do to their early robot designs. While we value the sense of attachment students are able to gain from personalizing their robots, we do not want students to view their first robot design as a final product. We then walk students through a group activity (the Robot Dance Party) in which they are asked to program their robots to music. This task, as well as the Show and Tell, robot Improv, and Charades are designed to make girls program their robots and evaluate the robots expressiveness. To spur reflection and critical evaluation, we also ask students to document their designs extensively. Students were given ten minutes to write a reflection at the end of each design session, and also began all design sessions by explicitly planning and drawing their proposed designs.

In addition to the robot activities in session four, we also introduce messenger, the third component of the Robot Diaries software environment. With messenger, girls can chat with others and share the programs they write using Express-O-Matic with some or all of the other girls. This sharing allows girls to run programs on one another's robots and to effectively collaborate in some of the future group activities. At the end of session four students are ready to revise and expand their designs and are given both a longer time to design and build as well as more freedom (in terms of constraints on parts) in the design direction.

Part Three: Iteration Part three is designed to drive home the importance of iteration, and so we begin with a story of the design process an engineer went through to create a robotic, emotive, desk lamp. We then ask girls to reflect on how to iterate on their robot, and also provide them with additional components (more LEDs, motors, and servos). The next few sessions follow a familiar pattern: much of the time is set aside for rebuilding and iterating on robots, followed by activities that are designed to highlight the expressiveness of the robots. We also introduce two new types of components into the mix: speakers and the ability of the robot to speak and play audio files, and simple light and distance sensors, which can be

5.5. Designing for Dissemination

Activity	Estimated Time
Session 5	
Individual activity follow-up	10 minutes
Iteration Example: Robotic Desk Lamp	10 minutes
Design Reflection	5 minutes
Making Changes 1: Brainstorming and Building	60 minutes
Introduction to speakers	5 minutes
Group activity: Robot Mad Libs	25 minutes
Document designs + reflection	10 minutes
Individual activity 3: Robot meets world	5 minutes
Session 6	
Individual activity follow-up	10 minutes
Making Changes 2: Design reflection and brainstorming	15 minutes
Building (adding new components to robot)	50 minutes
Prepare for show and tell	15 minutes
Group show and tell and feedback	20 minutes
Document designs + reflection	10 minutes
Individual activity 4: Pull the Chain	5 minutes
Session 7	
Individual activity follow-up	10 minutes
Introduction to sensors	20 minutes
Introduce idea of Play	5 minutes
Planning for Play	30 minutes
Design/re-design and building for play	50 minutes
Documenting designs + reflection	10 minutes
Individual activity 5: Silly messages	5 minutes
Session 8	
Individual activity follow-up	10 minutes
Continue working on play	75 minutes
Group Activity: Robot makeover	30 minutes
Survey	15 minutes
Session 9	
Group Activity: Play Presentation	30-60 minutes

Table 5.4: Sequence of activities for part three

5. Robot Diaries

used to affect program operation (for example, the robot can be programmed to trigger a sequence when a bright light is shone on it).

Towards the end of the curriculum, the work becomes focused on the play, a final project in which the robots are programmed to use their new range of emotional and communicative expression as actors in a theatrical context. In sessions seven and eight, girls are engaged in a multi-faceted design, working on both the theatrical production and on building their robots to have the right capabilities for this production. In some ways, the play is our answer to the experience of the robotics competition: parents and friends are encouraged to attend, and it allows for conversations to occur about robots between the girls and their parents/friends on many different levels. Of course, there is a stark difference as well: the play is a highly collaborative event, in which the play's quality is dependent in many ways on how well girls worked together.

Teacher Professional Development

We created a teacher training curriculum in addition to the curriculum designed to aid the research team in training. This training curriculum was a crash course version of parts 1 and 2 of the curriculum. In much the same way as the teachers would teach the girls, we trained the teachers by holding a mini-workshop (roughly six hours in length) where they learned about the building and robotics materials, programming, and the design process. In addition to these learning-by-doing tasks, we also went through the entire curriculum, describing each activity, emphasizing the design process theme, and discussing the learning goals.

5.5.3 Assessment Approach

This version of the Robot Diaries curriculum was designed as an after-school activity, and so our approach was to provide opportunities for guided self-assessment instead of creating an environment in which participants would receive a summative grade or critical statement at the end of the course. Feedback was provided through group activities after each design cycle that were designed to engage the robots' expressive abilities. These activities would allow the girls to test out their current designs and determine for themselves how closely those designs met the goal of an expressive, communicative robot. After each of these activities, girls were asked to document their current design and reflect on its capabilities in writing, with specific probing questions prompting critical reflection. In addition to self-assessment, we also included a couple of show-and-tell activities, in which girls would describe their robots and receive critical feedback from the group.

5.5.4 Tools

Our experiences with Robot Diaries have made us thoughtful about the ways in which technology can enable fluency experiences or serve as a barrier. After the completion of the participatory design sequences, we set out to redesign the Robot Diaries technology. The central goals of the redesign were to maintain functionality and features useful to Robot Diaries, while drastically reducing cost and improving user-friendliness. To the extent that the technology is transparent and easy-to-use, it can greatly enhance a participant's experience in the program; conversely, non-functioning technology can frustrate students and lead to disengagement.

During the fall 2006 workshop, the girls used the Qwerk as their central robot controller. A Qwerk is a powerful controller capable of controlling large robots, communicating over wireless networks, and sending video over a webcam. Although the Qwerk was expensive, we used it in our initial pilots because it had a very large set of features, providing us with the flexibility to experiment with features and determine if they were valuable to retain. In our pilot iteration of Robot Diaries we relied on wireless networks in participants' homes to control the robots. While this approach worked well in some homes, others experienced connectivity problems that led to decreased robot use.

The Qwerk's poor user-friendliness (at least where middle schoolers are concerned) and high cost suggested to us that it could not form the basis of a broadly disseminable design. Furthermore, the Qwerk was far more capable than necessary for our program's needs - using it was akin to using a supercomputer to play checkers: possible, but insulting to the computer. The Qwerk's large feature set was very useful in our participatory design sequence; it allowed us to experiment with many types of robot components - sensors, sound, LEDs, servos, and many types of motors.

This experimentation, as well as an analysis of the proposed curricular activities and learning goals, led us to determine a set of features for the new Robot Diaries controller, dubbed "Hummingbird" (see Figure 5.9). We decided to retain the ability to control servos, LEDs, motors, vibration motors, and a speaker, and our device would need to be directly connected to a computer over USB to simplify the setup process for home use. Furthermore, even though we had not found a direct use for them at the time that we were specifying our feature set, we felt it important to leave open the ability to use sensors; this was an easy design decision to make from a technological point of view, as it required no engineering trade-offs. Had it not been possible to do without sacrificing another feature, we would not have done it.

Fortunately, none of the required features require the use of complex or expensive technology, and so we were able to design a new controller with the necessary

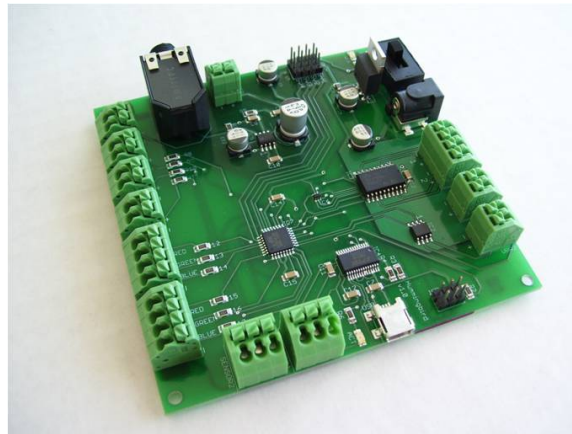


Figure 5.9: The Hummingbird microcontroller

capabilities. Whereas the Qwerk is \$349 retail, the Hummingbird could be sold for less than \$60.

Reducing cost was one goal of the redesign, but ensuring that the Hummingbird was easy to use was paramount. Connecting the Hummingbird to the computer over USB (and thus running the programs on the computer) eliminated the wireless tethering issues students had in the participatory design workshops. The Hummingbird was designed around the notion that it is better to have well-labeled ports that support a single feature than multi-purpose ports that can support sensors inputs, digital outputs, or motors. Multi-purpose ports are certainly desirable for some users as they increase the flexibility of the controller, but they are not helpful to novices. The Hummingbird has four ports for single-color LEDs, two ports for multi-color LEDs, two ports for regular motors, two ports for vibration motors, a speaker port, four servo ports, and two sensor ports. In most cases the port connectors are a very easy to use type where a connection is made simply by depressing a tab and sticking in a wire; there is no need to use screw drivers or other tools.

We also developed a software programming and sharing environment, pictured in Figure 5.10. We made two major changes to the software compared to the 2006 software interfaces: First, we integrated the three interfaces used in 2006 (RuR, Express-O-Matic, and Roboticon Messenger) into a single environment, thus eliminating a number of user interface issues that students in 2006 had. Second, we updated the software to work with the Hummingbird, and customized the RuR so that it reflected the ports available on the Hummingbird. Girls were very quick to pick up programming in the 2006 pilots using the expressions to sequences programming method, and so we did not change the way in which robots are programmed.

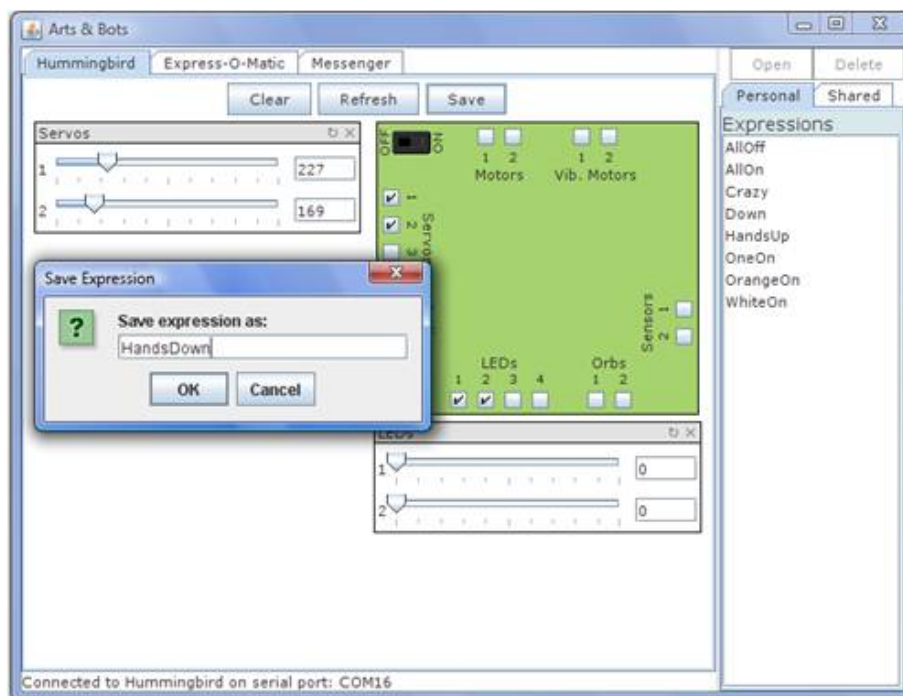


Figure 5.10: The Arts&Bots software environment

5.5.5 Evaluation Strategy

We developed an evaluation strategy prior to the program pilots to determine if the new iteration of the robot diaries curriculum was successfully meeting the technology fluency learning goal and to inform any future designs of the program. We were planning to keep our pilots fairly small, with between five and fifteen participants per pilot and only two or three pilots in total. Thus, we would be below the number of participants required to achieve statistically reliable quantitative results. Instead, we developed a strategy to capture as much information about the pilots as possible. We employed a number of fairly standard methods, notably:

Pre/Post Interviews. Each girl was interviewed before the workshop began and after it ended. Interviews directly asked girls about their interest in robotics, science and technology, and attempted to detect change in our dispositional learning goals. Interviews also included questions about relevant declarative knowledge (e.g., identify and provide a definition for relevant parts, such as sensors and servos) and designed systems (e.g., examine an electronic toy and describe how it

5. Robot Diaries

works). We also asked participants to imagine how they might build a new system (an alarm) using a fixed set of components (a battery pack, alligator clips, switch, LED, servo, and sensor).

Research Observations. One or more researchers attended every workshop session, logging girls' reactions to new activities, documenting moments of frustration or achievement, and determining how closely the teachers followed the curriculum. The workshops were also video-recorded to allow the entire research team to view moments of specific interest.

Teacher Interviews. We interviewed teachers to get their impressions on how the girls proceeded through the program, as well as gather information about what was difficult in implementing the curriculum.

Parent Interviews. We interviewed parents at the beginning and end of the workshop. Interviews focused on the child's previous experience with robotics and related technologies, family activities related to science and technology, and parent impressions of the workshop and what their child gained from participation.

These methods were sufficient to provide us with information relating to the dispositional learning goals. We received data regarding girls' dispositions towards science and technology from the girls themselves, from our own observations, and from the teachers and parents. They also provided some information related to the knowledge and skills goals. However, we also wanted to assess girls' change in creativity and design skills as well as technical skills such as debugging. While the teacher interviews and our own observations provided some subjective information on this, we felt it necessary to create two new methods that explored change in these learning goals in a less subjective and finer-grained way. These methods were:

Debugging Task. (Hamner et al., 2010) To assess whether girls were able to work with and troubleshoot the hardware and software after the workshop concluded, we gave them a robot made from the same materials and components that was broken in five distinct ways and asked them to fix it.

Creative Design Exercise. To assess change in how girls thought about designing and attempt to measure transfer of design ability, at least within the context of expressive electronic devices, we asked them both before the workshop started and after it concluded to sketch and model a design of a device that could solve a problem.

5.5.6 Fluency Moments - an Analysis Methodology

All of our evaluation methods, and especially the debugging task and creative design exercise were designed to create data to allow us to understand the extent to which participating in Robot Diaries provides girls with the knowledge, dispositions, and other tools they need to move towards technological fluency.

One way to measure movement towards technological fluency is by examining participant experiences captured using the above evaluation methods for expressions of fluency, which we have called “fluency moments” (Bernstein, 2010). These are moments in which participants’ knowledge, motivation, interest, and confidence come together to support their exploration and development of new technological solutions to problems they have identified. Our analysis of the data from the fall 2006 participatory design workshop led us to one such fluency moment, as described below.

The following excerpt was captured on a Robot Diaries participant’s digital video camera. In the excerpt, a participant named Rosemary [not her real name] explains how, during her winter break, she used a servo from the workshop to modify the robot she had created (a penguin):

During break, I decided to do something with the extra servo I got. So, I added a servo to the nose. And it’s kind of being difficult now, because I need a replacement nose. But, here, let me spin it (penguins nose moves back and forth on its own she is controlling to from her computer). It basically goes back and forth. And what I did is, I unscrewed the inside part of the servo (she removes the cardboard nose, and points to the motor shaft on the servo), and I broke a toothpick and stuck it in [the motor shaft], and I stuck my beak then into it (demonstrates with the beak), and it was able to move for some time. But then as the hole got bigger it wouldn’t move so much, so once I get the superglue, I’m going to superglue it together, so I can just put it in and out, and that will allow me to have maybe multiple beakswwhich would be pretty neat.

By analyzing these types of examples, we can begin to examine the characteristics that move individuals and groups towards fluency. In this excerpt, Rosemary identified a problem on her robot: the lack of a movable beak. She then applied some of her knowledge of servos as well as other tools (toothpicks, superglue) to develop a feasible solution on her own. This is an example of a self-generated adaptation. Rosemary chose to add this component to her robot during her vacation, indicating a certain level of engagement with the technology. We might also

5. Robot Diaries

assume a certain level of confidence on Rosemary's part, as she was able to undertake and successfully carry out the desired modification on her robot. We can speculate that Rosemary's knowledge, engagement, motivation, and ability to carry out the task outside of the workshop setting all set her up well to engage in future fluency activities.

5.5.7 Piloting the Design

We began to look for community partners to pilot our new curriculum in the spring of 2008. Our community partners provided a vital link between us and the students we are trying to reach: they take on the responsibilities of recruiting children, provide computers and a place for groups to meet, and provide instructors to lead the Robot Diaries sessions. In order to maintain the integrity of the program and to work with community partners as effectively as possible, we developed a set of suggested program guidelines. First, students in the workshop should be regular, committed participants in the program. We did not feel that casual, drop-in participation would provide enough continuity to make the program effective. Regarding facilities and staffing, we recommended that workshop sites have at least one Internet-connected computer per three participating students and one instructor for roughly every six girls. Although we provide training in the curriculum, robot hardware, and software, instructors were required to have basic computer literacy skills, comfort using simple hand tools, and familiarity with fields such as art, design, or drama, to effectively lead the workshops. The creative building focus of Robot Diaries means that students will require ample space to explore materials and build their robots. Lastly, we were interested in pilots of our entire curriculum, which required a time commitment of approximately 15 to 18 hours with students to complete. After discussing the program with a number of groups, we decided to work with two community groups, the People Always Learning Something (PALS) home school group and the Sarah Heinz House. We describe our experiences with each group next.

PALS

PALS offers classes to home school students on a wide variety of topics. Classes are often taught by parent volunteers. Seven girls participated in the pilot, ranging in age from 9 to 14. The girls came from middle to upper-middle class backgrounds, and started with a fair amount of enthusiasm for science and engineering activities. Classes were taught by two parents, both of whom had a child in the course. The classes were held on Saturday afternoons in one of the instructors' basements. There were six classes held over seven weeks (one weekend was

skipped due to scheduling conflicts). Classes were three to four hours in length.

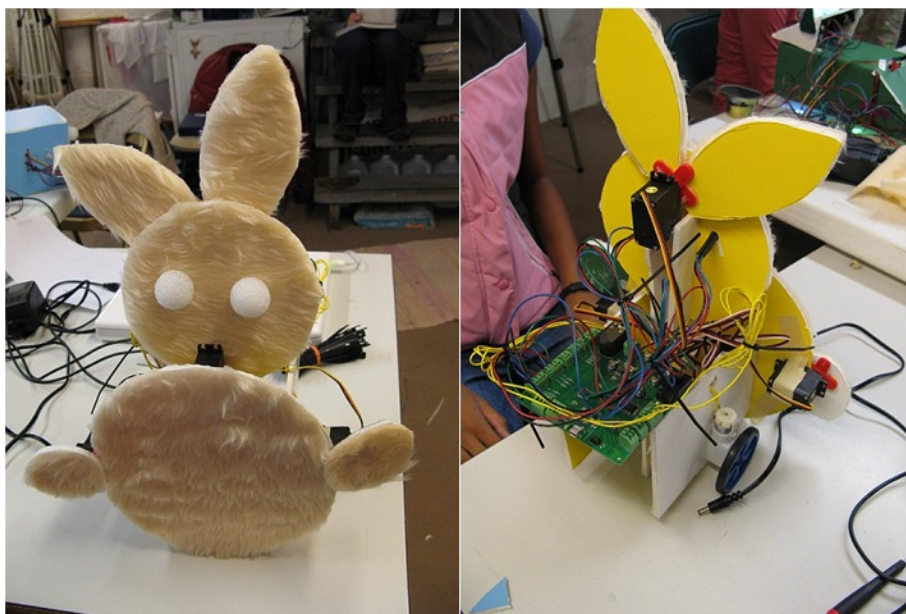


Figure 5.11: One girl's final robot from the PALS workshop

Sarah Heinz House

The Sarah Heinz House offers after-school programs much like a boys and girls club, however unlike most boys and girls clubs, students are required to attend several sessions a week in order to maintain their club member status. Ten girls participated in the pilot, ranging in age from 11 to 14. The girls came from all socioeconomic strata and had none to extensive experience with science/technology activities. The class was held at the House once per week for sixteen weeks by two instructors; one instructor specialized in technology activities at the house, and the other specialized in the arts. Classes were one hour in length.

5.5.8 Evaluation

The analysis of the evaluation data from the PALS and Sarah Heinz House pilots has not been completed. Portions of the data from PALS have been analyzed and are summarized here; a full analysis is available in Bernstein (2010). Specifically, we have analyzed the data from pre-post surveys on measures of confidence and interest with respect to robots, robot knowledge, and creativity.

Confidence and Interest

All seven workshop participants showed a greater degree of confidence in their ability to work with robots at the post-interview. More generally, three of seven girls showed greater confidence in working with computers, one showed less confidence, and the other three stayed the same. On questions that sought to examine confidence with everyday technology, there was no net change. This suggests that the workshop improved confidence levels for working with robots and brought these up to the levels exhibited for other technologies used by the girls. The group of girls at PALS was self-selected, and reported fairly high confidence in using everyday and computer technologies in the pre-interview.

Girls did not show a global change in interest with respect to any category of technology (every-day, computers, or robots). However, unlike with confidence, girls started the workshop with higher interest ratings in the robots category than for computers or every-day technologies; two of the seven girls hit the measure's ceiling on interest in robots in both pre-and post-interviews.

Robot Knowledge

We measured each girl's knowledge in five categories before and after the workshop: identifying robot parts, sensor operation, electric motor operation, systems reasoning, and system components. As there were seven girls, there are 35 total opportunities to witness change in a category. Of these 35, girls' knowledge was already at ceiling in the pre-workshop measures on sixteen of the opportunities. Of the remaining 19, girls showed positive improvement in 14, no change in four, and negative change in one. Positive knowledge change was most marked around the theme of computation; girls generally had a better understanding at post of the importance of a computational object like a microcontroller or computer to the operation of robots. In sum, it would appear that girls' overall knowledge, while high to begin with, was positively affected by the workshop. Effects may have been stronger if either girls had lower initial knowledge or if our measures had had a higher ceiling.

Creativity

Creativity was measured primarily by the design exercise, an activity which was given to girls both before and after the workshop. We attempted to measure divergent thinking; that is, the number of potentially valid solutions to a problem that girls could come up with in a given time frame. Unfortunately, we found no net effect of the workshop on these measures, and so no evidence that the workshop improved girls' creativity in the robot design context.

Anecdotal Observations

In addition to the preliminary results, we have also collected a list of representative quotes about various aspects of the program from girls involved in the PALS pilot.

Several children mentioned the ability to enact their own ideas on the robots as a positive aspect of the program:

I really liked that we use a lot of robotics, we're like really into it instead of just having a robot and learning how to program it. We actually get to make our own thing. [PALS participant, age 14]

Getting to build the robot. I think that was the coolest part I liked having like all - like any supplies you wanted to. You could do anything you wanted to. You could make it as big or as little as you wanted to. You could do like anything basically. [PALS participant, age 14]

Some participants expressed pleasure at the complexity of their robotic creations, and at the general level of engagement the program enabled:

You can make it do so many different things with just vibration motors, servos and LED's and a speaker. Like when I made it be mad, I had the vibration motors and then there was a siren and then red lights and that was very funny. And it just, and you could just really tell that it was mad. [PALS participant, age 11]

Well I didn't think it would be so advanced. Like I didn't think you guys would actually have websites for all of us and have a Hummingbird built for all of us. I thought it was going to be like kind of like the robots you get in the kits that you buy at like science stores. [PALS participant, age 14]

Additionally, participants' comments indicate that Robot Diaries has the potential to broaden children's ideas about the accessibility of robotics:

If you had told me like maybe 6 months ago that I was going to be building a robot that could move and sense its environment I would like, I probably wouldn't believe it. I can build that stuff? And then it was like once you learned it was like really easy. [PALS participant, age 10]

5.6 Next Steps

The program developed to date meets some of the criteria for disseminability - it is capable of being taught by external educators who do not have a background in engineering or robotics, and it produces qualitatively positive outcomes among participants. Our efforts in the next few years will revolve around adapting the program to different educational settings and creating a broadly disseminable program.

5.6.1 Next Steps: Adapting Robot Diaries to Formal Educational Settings

The Robot Diaries program is being expanded through a grant from the National Science Foundation to formal educational settings. This three-year grant will begin in 2010 with an effort to work with teachers at one local public school and one magnet science and technology school to adapt the curricular approach, activities, and materials created to date through robot diaries to the classroom. We will run the curriculum with these partner teachers in fall 2010, revise it based on that experience in spring 2011, and run it again in fall 2011. After revising the curriculum in spring of 2012, we will focus on disseminating the completed, adapted curriculum to a network of local school teachers.

5.6.2 Improving Disseminability

Discussing dissemination with our partners at PALS and the Sarah Heinz House, we determined that in order to disseminate the program we would need two things:

1. Remote Training. It will not be possible to have members of the research team train individual educators who live outside the Pittsburgh region, and while existing teachers can train new ones, this process will be too slow for widespread dissemination. Thus, it will be necessary to create materials which allow us to train interested educators remotely, potentially through an online course or video of a training workshop supplemented by online interaction with the research team or educators who have already run the Robot Diaries workshop.
2. A Commercial Source for Materials. The distribution of Robot Diaries hardware must be done commercially so that purchasing and repairs are handled professionally. The CREATE Lab has a history of open-source releases combined with commercialization of necessary hardware in cases such as the CMUcam vision systems (CMUcam 2008) and the Telepresence Robot Kit

Qwerk processor (Qwerk 2008). In the case of Robot Diaries, the ideal commercial partner is an educational technology or curriculum company that has an established brand in the learning community.

5.7 Summary

The core theme of Robot Diaries is unchanged: We are developing a program to provide an alternative robotics experience for students not well-served by competitive robots. However, our approach has undergone several major modifications driven by our experiences working with students. Though we always planned to include a design element to the program, later incarnations of the program have become steadily more focused on having students create their entire robot, and on leveraging this creative process as a way of teaching engineering design. We backed off from some of our initial ideas after the focus group, and focused in on engineering design after seeing the girls in our participatory design workshops engage in “organic design”. We believe that the curriculum and current tools created are a well-aligned design for an after-school workshop. We look forward to aligning the tools and activities to in-school environments.

5. Robot Diaries

Chapter 6

CSbots

6.1 Motivation and Program Goal

In the context of steeply declining enrollments in Computer Science (Vegso, 2005), the CSbots program focused on developing curricular modules for introduction to Computer Science (CS1) classes in which robots are used as educational tools to motivate students about applications of Computing. Since 2000, enrollment declines in computer science classes have led to a wide number of approaches to motivate students to study computer science: Alice (Cooper et al., 2003), Scratch (Maloney et al., 2008), and CS Unplugged (Bell and Fellows, 2010) are all efforts resulting in part from this crisis.

Robots, too, have been used by a number of Computer Science educators, both in introductory Computer Science courses (CS1) (Fagin and Merkle, 2003; Blank et al., 2007), and higher level courses (McNally, 2006). The results to date have been mixed, with a very large study using Lego Mindstorms robots in CS1 (Fagin and Merkle, 2003) finding two critical weaknesses of using robots: Firstly, robots are typically too expensive for student ownership, and so students must work on robot programming assignments in labs with limited hours. Secondly, feedback is delayed due to the need to observe a program running in the physical environment, and so students must devote more time to tedious debugging, and less to developing solutions. These weaknesses can be seen as failures of alignment of the tool to the ecosystem of the introductory Computer Science course, a topic that we will be expanded on later in this chapter.

The aim of CSbots, then, was to create a robot and accompanying curriculum that had the advantages of robots seen in other educational contexts (namely improved motivation and interest in studying STEM (Melchior et al., 2005)), while eliminating the weaknesses found in previous work.

19

© 2000 Blackwell Science Ltd *Journal of Internal Medicine* 247: 111–117

velopment of which began in spring of 2008. The new robot, dubbed Finch, was used in pilots at a local community college in spring and fall of 2009, as well as by a number of high school teachers. We also established a lending program to allow educators to borrow Finches for short term exploration. The analysis of the 2009 community college and high school pilots concludes this chapter.

6.2.1 My Role in the Project

I led the CSbots design team. As part of my responsibilities, I recruited and was the primary point of contact with partner educators. I participated in all of the pilot studies described and created the evaluation scheme for these studies; I also performed the analysis of the evaluation data. I did the majority of the design work for the robots used in the pilots, and contributed to the design of the software environment (especially to the high-level API that students use). I also designed a number of the assignments used during the pilots.

6.3 Approach

Our approach to creating the CSbots program rests on four methods; an initial evaluation of the field, feedback and evaluation driven iteration of design, deep partnerships with educators working in the Computer Science field, and alignment of the robot and curriculum with learning goals and audience of the CS1 class.

6.3.1 Initial Evaluation

As none of the members of the research team had taught a CS1 course, we felt it necessary to do pre-design evaluations to determine the learning goals of the course, the logistics and constraints of running such a course at different institutions, and the receptiveness among educators to using robotics in their class.

6.3.2 Iterative Design

We engaged in an iterative design process with extensive participation from computer science educators at two and four year schools. The process was composed of the following steps:

Design. The design step involved the creation of a robotic platform and associated software, and the development of assignments and course outline.

Pilot. The pilot step involved the test of the designed curriculum and technology in a CS1 course.

Evaluation. Learning, motivation, and retention were tracked with standard exams, weekly student surveys, and comparisons of drop-out rates. These data were used to inform the next design step.

6.3.3 Partnerships

Given our own lack of experience in teaching Computer Science, we decided to engage with educators of CS1. Our intention was to develop a partnership that was more than simply advisory, with a two-way sharing of domain knowledge and skill. Our partners would be involved from the start of the design process, allowing them to test and comment on early versions of the robot and software, while we would have access to their existing curricula and their crucial understanding of what works with students in the CS1 class. While the details of the robot design were left up to us, we worked together to design the interface of the software framework, and the curricular activities.

Given the length and nature of this program, we developed partnerships with multiple educators, some of which were temporary and some of which are still on-going. We began with a focus on community colleges, as student retention is especially critical in these institutions. We partnered with three community college educators after the initial evaluation but prior to any design work. After an initial robot was created, we also partnered with five high school teachers. These partnerships have taken effort for both parties to develop and maintain.

6.3.4 Alignment

As previously mentioned in this thesis, alignment is a powerful concept that can be extended to the design of educational technology. The features of a technological system provide a fourth design node, feeding back on and aligning with the three traditional nodes of learning goals, instructional methods, and assessments. Among the three programs discussed in the thesis, CSbots exists in the most constrained design space. As we will see in later sections, the learning goals, many of the assessments, and some of the instruction is unchangeable. As such, the major design elements that can be adjusted to align with these unchangeable elements are the robot's feature sets, and the instructional elements that use the robot.

6.4 Ideation

Much like Robot Diaries, the initial idea came from a more general survey of ways in which robotics technologies could be applied to education; we brainstormed a number of topics where robots could be used as tools, and eventually looked more

closely at Computer Science. Unlike Robot Diaries, we were guided by previous work, especially the work by Fagin and others in using robots in Computer Science. We felt that the basic idea (robots used as tools in computer science classes) was sound but that the approach taken in past studies had caused disruptions to the way Computer Science was naturally taught that caused the negative effects reported by these studies.

6.5 Initial Evaluation

To ensure that our designs were grounded in the realities of Computer Science education we engaged in extensive pre-design evaluations in fall 2005. These evaluations included a textbook survey that sought to discover unifying themes and learning objectives and a survey of educators to characterize curricular design constraints at different institutions as well as receptiveness to using robotics in the CS1 class.

6.5.1 Textbook Survey

In spring of 2006 we performed an analysis of ten major CS1 textbooks to help us develop a conception of the standard CS1 curriculum and discover the major learning objectives of the class. This analysis sought to answer a number of fundamental curricular questions:

- Does the textbook assume a specific Integrated Development Environment (IDE)?
- Does the textbook focus on the science of computing or is it a trade book focused on teaching students how to program?
- Are the problem sets cumulative or modular?
- Does the textbook require the use of additional hardware or software?
- What is the order and list of topics covered?

We used three sources to determine popular introductory CS texts. The first was a survey of web sites of CS1 courses at four-year universities across the United States. From a group of about 60 CS1 instructors, about 30 had accessible course web sites. The next measure of popularity was to look at the textbooks used at top universities such as MIT, Stanford, Harvard, etc. Lastly we emailed the CS1 instructors and the assistant dean of undergraduate education in the CS department at Carnegie Mellon University (CMU) to ask them which textbooks they thought

6. CSbots

were the most popular and widely used. Table 6.1 presents the top ten textbooks identified using these methods.

Textbook	IDE	Problem Sets	Additional Resources
<i>Head First Java</i> , (Sierra and Bates, 2005)	none	mostly modular	no
<i>Java How To Program</i> , (Deitel and Deitel, 2005)	none	modular	no
<i>Big Java</i> , (Horstmann, 2006)	BlueJ suggested	mostly modular	internet connection
<i>Objects First with Java</i> , (Barnes and Kolling, 2005)	BlueJ required	cumulative	no
<i>Java Concepts</i> , (Horstmann, 2005)	BlueJ suggested	mostly modular	internet connection
<i>C++ Programming</i> , (Malik, 2004)	none	mostly modular	no
<i>Java Software Solutions</i> , (Lewis and Loftus, 2005)	none	modular	no
<i>Computer Science</i> , (Forouzan and Gilberg, 2001)	none	modular	no
<i>The Art and Science of C</i> , (Roberts, 1995)	none	modular	additional libraries
<i>Absolute Java</i> , (Savitch, 2006)	TextPad suggested	modular	no

Table 6.1: Summary of key textbook attributes

Results. From Table 6.1, only one book assumed that students were using a specific IDE, although several others suggested and even came with IDEs. All of the books were centered on teaching the skill of programming in a specific language, not on high-level computing concepts, and so they were all classified as trade books. Nine of the ten books had problem sets that were mostly modular; exercises from one chapter were not based on code written in a previous chapter. Most textbooks required no additional hardware or software, although two books assumed an internet connection was available, and one came with non-standard software libraries for use in some of the book assignments.

We analyzed the order of six topics that were covered in most of the textbooks and that have potential for self-contained robotics modules. Those six topics are:

V: Variables

I: Simple input/output (I/O) (screen I/O with characters or strings)

F: Flow (conditionals, loops, relational operators)

D: Arrays

E: Exceptions and errors

A: Advanced I/O (file I/O)

The most common ordering of these topics was: variables, simple I/O, flow control, arrays, exceptions, and advanced I/O. The topics were found in this order in four of the ten books. Variables and simple I/O were introduced first in all books but one, with simple I/O coming before variables in only three cases. Flow control was the third topic in all but two books. The fourth topic was arrays in six of the books. In the other four books, arrays was the fifth topic. Exceptions came after arrays in all but two cases (once it was directly before arrays, and once exceptions was missing entirely). Advanced I/O was the last topic in seven of the books. (See Table 6.2)

Number of Books	Topic Order	Programming Languages
4	V I F D E A	Java, Java, Java, Java
1	I V F D E A	Java
1	V I F E D A	C
1	I V F A D E	Java
1	V I F A D -	C
1	V I A F D E	C++
1	I F V D E A	Java

Table 6.2: Topic orderings. The topics are variables (V), simple I/O (I), flow control (F), arrays (D), exceptions and errors (E), and advanced I/O (A).

Implications and Conclusions.

All of the books focused on programming skills and not computing concepts, and most presented a modular approach. This analysis implies that a broadly applicable curriculum should be composed of unrelated modules covering specific programming concepts. These modules could be picked up by educators and placed into their curriculum, regardless of their course schedule or textbook used. Based on our textbook analysis, modules with the following topic dependencies would fit into many CS1 courses:

Module 1: VI - Variables and simple I/O

Module 2: VIF - Flow control (with variables and simple I/O as prerequisites)

Module 3: VIF D - Arrays (with variables, simple I/O, and flow control as prerequisites)

Module 4: VIF E - Exceptions and error handling (with variables, simple I/O, and flow control as prerequisites)

6.5.2 Survey of Educators

We embarked on an extensive survey of educators at two- and four-year institutions (Lauwers and Nourbakhsh, 2007) during the 2005-2006 school year. We were inspired by the Taulbee survey (Zweben, 2005) and the McCauley and Manaris (McCauley and Manaris, 2002) studies, but our aims differed from these; instead of a broad-based, largely quantitative analysis of the state of Computer Science education, we were interested in analyzing the attitudes, opinions, and challenges faced specifically by CS1 educators. Our study sought to answer questions that were best asked in the context of a personal interview, and best analyzed through conceptual code-based qualitative metrics. The foundational questions that we sought to answer included:

- How do CS1 instructors feel about the effectiveness of their curricula, both in teaching students and in motivating them?
- To what degree are instructors able to make curricular changes?
- What are the typical dynamics and logistics of a CS1 course?
- What tools and programming languages do instructors currently use and what is their relative popularity?
- Are instructors interested in using robotics as a teaching tool?
- How do the classroom realities exposed by the previous questions inform methods for introducing new educational tools?

120 educators were identified as currently teaching CS1 at four year institutions within the United States. Of these 120, 33 responded to an email request to participate in the survey and were interviewed. In addition, we interviewed four educators at community colleges across the United States. The geographic distribution of all respondents is mapped in Figure 6.2. The educators came from a wide range of public and private, small and large institutions and were themselves diverse with respect to age, gender, and professional standing. Our results provided

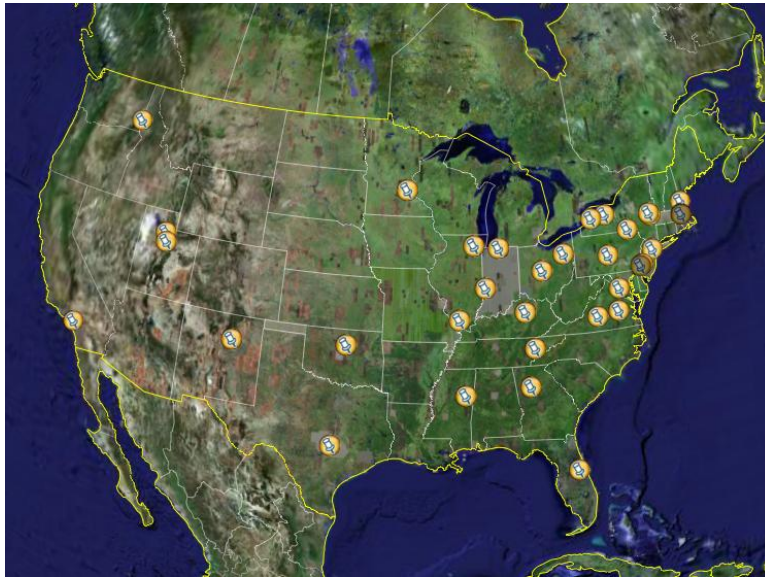


Figure 6.2: Distribution of survey respondents

tentative answers to the foundational questions and feed into an answer to our primary question: What methods and strategies should we employ in our attempt to introduce a new educational tool into the CS1 curriculum? Our conclusions can be split into those general to introducing a new educational tool and those specific to introducing robots.

Conclusions Applicable to any New CS1 Program

- A major curricular change is difficult to implement, requiring buy-in from the department faculty, and sometimes from other departments. Tools should therefore be introduced with a curriculum that is similar to that already in use.
- Tools should not be tied to a new or relatively rare programming language; not only does this require a major curricular change but few instructors are planning to change the programming language they use. Support for either Java or C++ is required for widespread adoption.
- Any new educational tool must integrate into the class such that it does not significantly increase student workloads.
- Given that most educators use textbooks as guides when embarking on a

significantly different curriculum, it may be useful to either develop or adopt an accompanying textbook and lab manual for the educational tool.

Conclusions Specific to Using Robots in CS1

- Students must be able to work on their out-of-class assignments at home. This is not an impossible goal for a robotic tool. This challenge can be met with a number of methods: the use of a simulator, remote or tele-present access to physical robots, or by providing each student with a very low cost robot.
- Developing for Java provides access to the largest and most enthusiastic base of potential robotics adopters. Educators using C/C++ were less interested in adopting robotics.
- Materials cost to students is widely perceived as more important than departmental costs. We do not believe that a robotics class should require students to purchase a robot unless it is in lieu of a similarly priced textbook.

The purpose of the educator survey was both to discover the realities of the group for whom we were developing the technology, and to ensure that members of this group were interested in our proposed technology. In these senses, the survey was successful at identifying key logistical difficulties of introducing robots in CS1, as well as establishing that a significant fraction of educators were willing and able to try such an intervention.

6.5.3 Partners' Prior Curricula

In addition to the formal textbook and educator surveys, we also worked with our partner educators to detail the learning objectives, instructional activities, and assessments in their specific courses. This was important not just as a pre-design evaluation method, but also because it was our intention to pilot the revised course with our partners in such a way as to not change the learning goals. In one case, our intentions were realized; we went through every exam, assignment, and prepared lecture used by our partner in the previous year. It was our goal to create a curriculum that would mirror this class at both the macro and micro levels: the learning goals of the class were to remain the same, as would the complexity and learning goals of the individual weekly assignments. Similarly, the assessments would cover the same material at the same point in the course schedule. By keeping constant much else of the class, it was easier to measure the effect of the robotics activities on the students. It also simplified the alignment process, as it meant that

we could not modify the learning goals or assessments. All that was necessary was to ensure that the instructional activities aligned with these two.

6.6 Initial Design

We used the results of our initial evaluation and analysis to simultaneously design a curriculum, robot platform, and software for the CS1 classes of our partner educators. Design occurred in 2006 and early 2007.

6.6.1 Learning Goals

The typical CS1 course is about programming, specifically, becoming literate in a programming language so as to be capable of writing programs of moderate complexity in that language. There is some controversy about whether this should be the goal of CS1; many in the computer science education community call for other approaches. Some computer scientists desire a more fundamental introduction into computing, beginning with the high level theorems of computer science and computational mathematics (Dijkstra, 1988). Others desire a broader introductory course that merges the history of computer science, current fields of interest like graphics, computational science, and robotics, and introduces programming as one of several tools used in this field.

Despite the potential advantages of these other approaches, the results of our survey strongly suggested to us that broad acceptance of our curriculum would only be possible if we mirrored the learning goals of the current CS1 course. The learning goals of the CS1 course were identified primarily to us through the textbook survey and our conversations with our partners. In a rough chronological order, these goals are:

- Compile a simple program that outputs text to the screen in the Integrated Development Environment (IDE) used in the class.
- Understand simple variable types, and how to assign values to variables.
- Learn how to use pre-defined objects and methods.
- Learn about conditional statements.
- Read in user input to the program.
- Learn about looping.
- Create methods (functions in C) outside of the main method.

- Create classes that are independent of the main program.
- Learn about arrays of simple variables and of objects.
- Learn to use one of the graphics packages to make graphical representations.
- Read data in from files and write out to files.
- Create classes that inherit characteristics from other classes.
- Learn to handle exceptions and create methods that throw exceptions.

6.6.2 Curriculum

Curriculum design was guided by several conclusions from our survey. Educators teaching Java were most enthusiastic about using robots, most educators wanted a new curriculum to be linked to a textbook, and robots could be introduced into a class with minimal administrative paperwork so long as the programming language did not change and the course schedule was minimally affected. We focused on Java and chose a specific textbook, “A Guide to Programming in Java” (Brown, 2007), to link to our curriculum. We took care to ensure that each assignment developed was focused on teaching CS concepts; we used the prior assignments from our design partners as a baseline and analyzed these to ensure our assignments were similar conceptually. Table 6.3 contains the tentative schedule for a semester-long CS1 course with robot-based assignments.

Prior to testing our assignments in a pilot, we ensured that they were at an appropriate level for our intended audience. To do this, we had a high school student with one semester of Java experience complete solutions for each assignment. Although most assignments were found to be appropriate to the level of our intended learners, this early testing did result in several changes made to the later assignments to reduce tangential complexity and focus more heavily on core concepts.

Week	Goals	Assignment
1	Learn how to compile/debug programs and use variables.	Write a program that causes the robot to speak.
2	Use simple arithmetic/boolean operators and print variables. Use documentation.	Debug a provided program.
3	Learn about classes by using two exemplars; the String class and the Robot class.	Make robot weather forecasters that say the forecast and move based on weather conditions.
4	Continue classes.	Move in a regular polygon.
5	Learn to allow input from screen and review of Boolean/logical expressions.	Move and turn from user input.
6	Use conditional statements like <i>if</i> and <i>switch</i> .	Have the robot react to a weather forecast based on a randomly generated mood.
7	Use looping statements like <i>for</i> , <i>while</i> , and <i>do-while</i> .	Wander around the room and avoid obstacles.
8	Write non-main methods.	Create a story-generator composed of random sentences.
9	Write custom classes.	Write a simple simulator.
10	Create graphical programs and handle button events.	Create a tele-op interface.
11	Learn to read text/string inputs from a graphical interface.	Improve the tele-op interface with a text input box.
12	Use arrays of primitive data types.	Script motion by reading bumper presses and then play a sequence of motions based on those presses.
13	Use arrays of objects.	Write a color-tracking program.
14	Introduce exceptions and exception handling.	Combine the wandering program with the tele-op program so that the robot throws an exception when it hits an obstacle and moves to tele-op control.
15	Review.	No assignment.

Table 6.3: Schedule for a CS1 robotics course

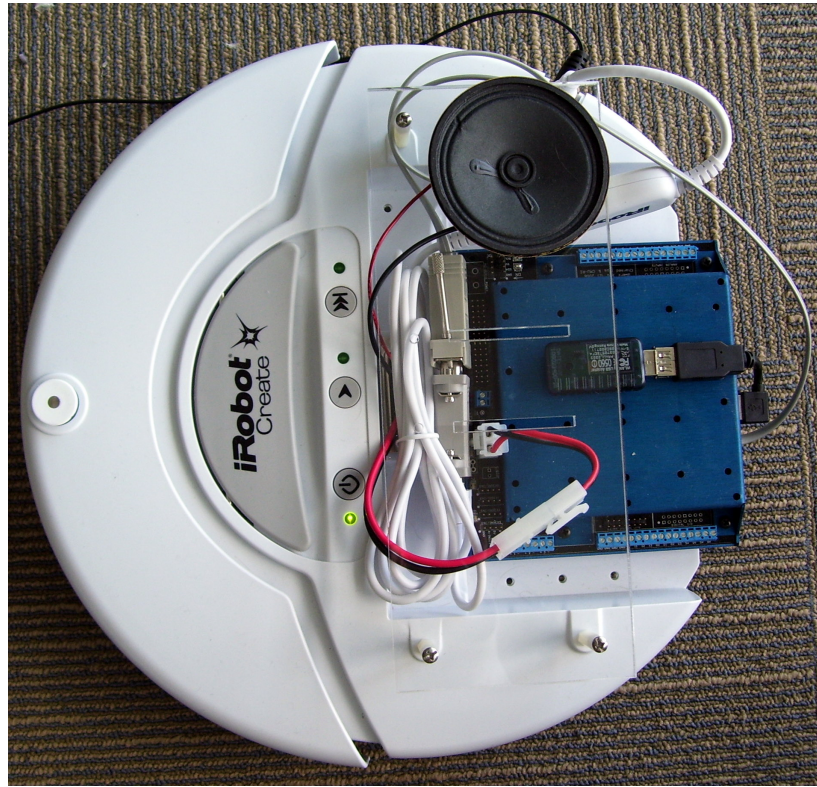


Figure 6.3: The first year robot platform

6.6.3 Robot Platform

The end goal of our technology development process was to create a robot platform and accompanying software that had the correct sensing and actuating interactions for the introduction to computer science class while simultaneously being sufficiently low cost to be used in such classes. In line with these goals, during our first design cycle we focused on maximizing the number of testable sensing and actuation interactions included on the platform while keeping the cost per platform below \$1000 so as to allow sizable initial pilots. This strategy allowed us to test individual features during our pilots and determine an optimal feature set to guide our second robot design.

The platform that we used in the first design cycle is an iRobot Create¹ combined with a Qwerk controller² (Figure 6.3). The iRobot Create is a robotic plat-

¹<http://www.irobot.com/sp.cfm?pageid=305>

²http://www.charmedlabs.com/index.php?option=com_content&task=view&id=29

form based on the popular Roomba line of robotic vacuum cleaners. The Qwerk, developed by Charmed Labs and the CREATE lab, contains circuitry for low level robot control, as well as firmware allowing instant tele-operation over a wifi network with the addition of a USB wireless device and USB webcam. The Qwerk was accompanied by the TeRK software environment³, an extensive open-source software base for developers. Although not the lowest cost solution, the combined platform, dubbed the “alpha design” had the advantage of using off-the-shelf technology and maximizing feature richness. The combined platform had the following capabilities:

- A wireless tether, allowing programs to reside on a student’s computer
- Bumpers for simple obstacle detection
- Encoders to track distance traveled and maintain equivalent wheel speeds
- Vision via a USB webcam
- Audio, including the playing of audio wav files and the generation of speech from text
- An LED array as well as a color and intensity variable LED
- Access to real-time RSS feeds, enabling programs to respond to internet events

6.6.4 Software

The software environment we created was heavily influenced by the feedback we received from our partners as we were designing the environment and the constraints imposed by using the TeRK software environment. Unlike standard small Java programs which require little or no additional software libraries to run, our student-written Java programs would rest on top of several layers of increasingly advanced code; this code was necessary for connecting over the wireless network to the computer, for sending appropriately ‘phrased’ commands to the robot, and for supporting the graphical interface which was part of the software package.

One of the major themes in our conversations with our partner educators was to simplify as much as possible the student-visible level of computer code. Our first attempt to create a top-level file that students could edit is shown in Figure 6.4; our partners objected to this file on the grounds that it was much too complicated and had too many unexplainable sections for beginning CS students. Several months

³<http://www.terk.ri.cmu.edu/software/index.php>

6. CSbots

```
import javax.swing.SwingUtilities;
import edu.cmu.ri.mrpl.TeRK.client.components.easyclient.EasyClientBase;

public class EasyClient extends EasyClientBase
{
    /** The application name (appears in the title bar) */
    private static final String APPLICATION_NAME = "Easy Client";

    /** Properties file use to setup Ice for this application */
    private static final String ICE_PROPERTIES_FILE = "/EasyClient.ice.properties";

    public static void main(final String[] args)
    {
        // Schedule a job to show the GUI
        SwingUtilities.invokeLater(
            new Runnable()
            {
                public void run()
                {
                    new EasyClient();
                }
            }
        );
    }

    private EasyClient() {
        {
            super(APPLICATION_NAME, ICE_PROPERTIES_FILE);
        }
    }

    public void ExecuteUponStart() {
        clearMessageArea();
        appendMessage("Hello World!");
    }
}
```

Figure 6.4: Initial skeleton file for software framework

of refinement followed, eventually producing the starter file in Figure 6.5. Figure 6.5 shows a comparison between a program which causes the robot to say 'Hello World' and the standard 'Hello World' print program, and highlights the additional code required to run our version. The software libraries required were packaged into a single .jar file which could be used by a program with a single 'import' statement. The four lines of extra code in the program are fairly easy to explain to a novice user. One specifies the IP address of the robot to connect to (a number which is written on the robot's bumper in large bold print), one specifies the name of the program, one instantiates the robot object, and the last initializes the robot. Although students are required to use object oriented programming methods to call robot based methods before objects are theoretically explained, this did not prove to be a source of confusion, as *myRobot.saySomething("Hello");* and *System.out.println("Hello");* are equivalently complex and mysterious to the beginning programmer.

As we were developing the software environment, special attention was di-


```
import RobotClient.CreateClient;

public class SimpleSpeech
{
    public static void main(String[] args)
    {
        // Sets the name of the application
        String applicationName = "Speechifying";
        // Set the IP address to connect to
        String ipAddress = "192.168.0.10";
        // Instantiate the robot and GUI
        CreateClient myRobot = new CreateClient(applicationName, ipAddress);
        // Initialize the robot before other commands can be sent to it
        myRobot.initialize();
        // Say something friendly
        myRobot.saySomething("Hello World");
    }
}

public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello world");
    }
}
```

Figure 6.5: Programs that make the robot say “hello world” (top) and print “Hello World” to screen (bottom)

rected at ensuring that all method calls to the robot were relatively clear, such that reading the name of the method gives a proper indication of what that method will do. Students were provided with both a full Javadoc style listing of all robot class methods, as well as with a quick reference that covered the most important methods.

6.7 Pilots and Evaluation

Pilots were conducted at community colleges (Lauwers et al., 2009) and high schools (Lauwers et al., 2010) in the 2007-2008 school year. The community college pilots were intended to be trial runs of our robot and accompanying curriculum, with a formal evaluation system in place. The high school pilots were closer to a participatory design exercise, to determine if the system developed for the community colleges could be adapted to the high school environment. Combining the results of both of these pilots, we were able to execute a broadly applicable redesign during our next design phase.

6.7.1 CCAC Summer 2007

We conducted a brief week-long pilot during the regular Community College of Allegheny County (CCAC) Introduction to Computer Science course taught by our partner. During this period, two of the newly designed weekly assignments were tested with the students; specifically we gave the students assignments 7 and 12 from Table 6.3. Assignment 7 deals with looping and conditionals, a topic which the summer class had already covered, but which our partner suggested might be useful to repeat. Assignment 12 covers arrays, the topic that the class was in the process of covering during our trial. Roughly ten students completed each assignment.

Evaluation Methods

The goals of our evaluation of the summer pilot were to determine if the assignments we provided to the students were compelling, as well as to ensure that our survey materials were capturing the information we desired.

We surveyed students at the beginning of the course (pre-survey) to determine their age, grade point average, prior programming knowledge, reasons for taking the class, and interest in using robots in the course. We also surveyed students at the completion of each robotics assignment to determine how interesting they thought the assignments were compared to other assignments, how confident they felt with the subject material covered by the assignment, and how easy they felt the assignment was. Lastly, we offered a very small amount of extra credit on assignment 12 for a reasonably difficult extension to the assignment to determine if students were interested enough in using the robot to do extra work with minimal grade-related motivation.

Evaluation Results

Our pre-survey had 16 respondents, while the surveys regarding the assignments had 9 and 12 respondents each; this disparity mostly reflects students dropping the course before our pilot began. The start survey provided validation of our questions regarding confidence with computing, prior programming knowledge, and reasons for taking the course. There was one notable and unexpected result: 75% of students thought programming a robot would be more interesting than programming a computer.

The results from our assignment surveys showed our approach to be promising and worth testing in a larger pilot in the fall. For both surveys, students were asked to rate on a 1-5 scale how confident they were with the concepts, how interested they were in the assignment, whether this was their favorite assignment of the

class, and how easy or difficult the assignment was. Table 6.4 shows the averages for these results for assignments 7 and 12. As you can see from the table, both

Assignment	Topic	Confidence	Interest	Difficulty	Favorite
7	Loops	4.22	4.22	2.67	7 yes, 0 no
12	Arrays	3.58	4.08	4.08	7 yes, 3 no

Table 6.4: Confidence and interest ratings for summer robotics assignments

assignments got high marks for being interesting, and were considered the favorite assignments in the class by most students. Assignment 7 was listed as relatively easy, likely because it was considered a review assignment to repeat some earlier course material. Assignment 12 shows that even with new subject matter, the students enjoyed the assignment. Ten of the twelve students in the class completed assignment 12 successfully; of the other two, one was unable to complete the assignment due to a medical emergency. Furthermore, six of the ten students who successfully completed assignment 12 finished the extra credit portion of the assignment. Given these favorable results, we prepared to fully pilot the curriculum at two community colleges in the fall: CCAC and Ohlone Community College.

6.7.2 Ohlone fall 2007

The Ohlone pilot was conducted from early September through mid-December 2007. Fifteen students signed up for the class, but only four stayed in the class with a passing grade; although low, this is not an unusual retention rate for many computer science classes at both Ohlone and CCAC. The pilot was not framed as an introduction to computer science course, but as a robotics and AI course; even so, it provided us with an opportunity to test several of our activities, as well as our software framework and hardware platform. The class was held on Friday evenings for four hours, and so most of the students had day-time work and were interested in professional advancement. Students were only able to work with the robots during this limited class time.

Although we attempted to run a number of assignments in the course, technical problems prevented the robots from working reliably. After most of the semester had run its course, we discovered the source of the problem; the on-campus wireless network was occasionally and at random intervals throttling the wireless signals from our robots. This was an extremely difficult problem to diagnose, as the problem would come and go without warning, and so the robots would work for some time and then stop working. After a few weeks of these difficulties, our partner at Ohlone decided to de-emphasize the robotics portion of the class.

6. CSbots

A number of valuable lessons were learned through the experience of the Ohlone pilot. Firstly, our technology was vulnerable to issues beyond our immediate control; the Ohlone college wireless network was not configurable by our partner, and so it was difficult for him to experiment to discover the cause of the problem. Secondly, physical distance from the research team caused support to be much more limited in comparison to the support provided to CCAC. It is now clear that during the initial pilot the amount of support required was greater than our ability to provide that support.

6.7.3 CCAC fall 2007



Figure 6.6: Students in the CCAC robot lab

The CCAC pilot was held from late August through early December 2007. 72 students in four sections began the class; 25 students completed the course with a grade, and 23 of those passed. Three of the sections occurred during the day, and consisted of bi-weekly lectures of two hours each. The fourth section was held in the evening for four hours once per week. The evening course attracted a different audience than the daytime sections; these students were more likely to be working and using the course for professional development, while students in the daytime sections were more likely to use the class for fulfilling degree requirements.

In addition to lecture times, students were able to access the robots for testing and demonstrating assignments. Figure 6.6 shows some of the students in the robotics space. These open labs were staffed by the researcher, and were open from

noon to six on Wednesdays and Thursdays. These times were arranged with the students beforehand and represented the optimal amount of available time for all students. As many of the evening section students were unavailable to work during the day time, the first hour of their scheduled lecture time was assigned to testing with the robots.

Subject	Description
1. First program	Print a short statement to screen. This assignment did not use a robot.
2. Formatting output	Print a formatted block of text. This assignment did not use a robot.
3. Talking, dancing robots	Make the robot talk and make at minimum three distinct motions.
4. Conditional control structure	Have the robot sense bumper hits and condition based on those hits.
5. Looping control structure	Make the robot draw a polygon of 3-10 equal sides; the user inputs the number of sides.
6. Methods	The robot draws a shape, but each shape is a different method in the program.
7. Classes - Robot Tango	The program instantiates two robot objects representing different robots, and has to choreograph a dance between the two.
8. Classes - Story Teller	Students are provided with a main program which calls a class that they need to write. The class has to provide story snippets that the main program assembles into stories the robot tells/acts out.
9. Arrays - Record a Dance	Students create a program that uses bumper hits to store a sequence of directions. Playback of the sequence is also programmed by the students.
10. Applets	Students create a simple applet with graphics that appropriately resize as the applet window is stretched. This assignment did not use the robot.
11. GUIs	Students develop a simple interface to drive the robot around remotely.

Table 6.5: Listing of assignments used in CCAC pilot

Students were shown the robot during the first couple of class sessions, but because of the logistics involved in arranging a lab space and finding a time to hold the lab, the first assignment which used the robot platform was the third week's

assignment. It was also felt by the partner educator at the time that it might help the students to do a couple of traditional assignments before starting on the robot to reinforce some of the details of compiling programs and syntax.

About a week before each assignment was distributed, the principal researcher and partner educator would meet to discuss the upcoming assignment. We would use a previously developed assignment as a starting point and decide whether it was appropriate given our experience with the students to that point. In this way, we responded to our conceptions of students' abilities and aligned our upcoming assignments with those conceptions. Table 6.5 presents brief descriptions of the assignments that students were given; most of these assignments were entirely or heavily based on the previously developed curriculum; the assignments that were newly developed are designated with bold text.

Evaluation methods

The CCAC pilot was formally evaluated along a number of metrics to inform our second-stage design. We were especially interested in student motivation and retention, but we also assessed learning. We tracked retention of students after every assignment and compared it to the retention of students in courses offered by the same educator from fall 2003 to fall 2006. The retention data are rich. They show exactly which week in the class individual students stopped completing assignments, and so we can compare the piloted course to older courses not just on a gross scale, but on a fine-grained time-scale. Additionally, we tracked student interest in our assignments with short surveys that were completed after each assignment, and we also compared interest in CS as measured by pre and post surveys. To assess learning, we compared performance on traditional CS exams of the students in the pilot course to performance by the students in four previous fall semesters; these exams are changed superficially year over year to prevent cheating, but cover the same content. As the conceptual progression of the curriculum is the same as in the previous non-pilot courses, the exams are well-aligned with what students have learned.

Evaluation Results

We now present the results from the evaluation and analysis of the pre/post surveys, post-assignment surveys, and comparisons between pilot and prior year retention rates and grade performance.

Pre-Post survey results We asked students to complete surveys at the beginning and end of the course to gauge student prior experience, their reasons for taking

the class, and their interest in and confidence with computers and programming.

To gauge prior experience, we asked students to state which programming language they knew best, and if they had one, to rate their experience level with that language on a 1-5 scale (from novice to expert). Table 6.6 presents the results for pre and post surveys. Although 40 students took the pre survey and 20 the post survey, only 10 students took both; therefore we present comparisons between both the larger set, which includes data in the pre survey from students who dropped the class, as well as comparisons between the ten students who took both surveys. Not surprisingly, many students became more familiar with Java and no students answered 'none' for the post survey. Experience ratings did not increase because many students with no experience (and thus no rating) became beginners after taking the class.

Language	All Pre (n=41)	All Post (n=20)	Matched Pre (n=10)	Matched Post (n=10)
Java	17.07%	55.00%	20.00%	70.00%
C/C++	14.63%	15.00%	20.00%	20.00%
Other	17.07%	25.00%	20.00%	10.00%
None	51.22%	0.00%	40.00%	0.00%
Experience Rating	2.35	2.58	2.17	2.1

Table 6.6: Percentage of students who reported each category as their best-known programming language

We asked students to choose from a list of reasons why they were taking the CS1 class. Students could select multiple reasons: required for degree, plan to transfer credit to a 4 year school, want to program as a career, professional advancement, interested in programming and other. Table 6.7 presents the reasons students gave for taking the class; there was no statistically significant difference between pre and post surveys, but these data provides some insight into the diversity of needs the incoming students of a community college CS1 class have. It is interesting that the reasons changed somewhat among the matched set - this may represent a shifting focus over the course of the semester to more degree based and academic goals.

We gave students a number of statements intended to measure their interest in and confidence with computers and programming. Students were asked to rate the statements on a scale from 1 to 5 with 1 being strongly disagree, and 5 being strongly agree. Table 6.8 presents the results of these surveys. In both the matched and unmatched sets some of the measures of confidence and interest increased, although not by statistically significant amounts.

6. CSbots

Reason	All Pre (n=41)	All Post (n=20)	Matched Pre (n=10)	Matched Post (n=10)
Degree	41.46%	40.00%	30.00%	50.00%
Transfer	34.15%	40.00%	20.00%	50.00%
Career	31.71%	25.00%	50.00%	20.00%
Advancement	21.95%	40.00%	30.00%	30.00%
Interest	48.78%	50.00%	50.00%	50.00%
Other	12.20%	25.00%	10.00%	20.00%

Table 6.7: Student reasons for taking course

Statement	All Pre (n=41)	All Post (n=20)	Matched Pre (n=10)	Matched Post (n=10)
I am familiar with computers.	4.67	4.47	4.90	4.80
I am familiar with computer programming.	3.00	3.73	3.44	3.50
I am confident I could program a computer if I needed to.	2.95	3.21	2.60	3.10
Programming is easy for me.	3.08	3.37	3.22	3.10
I like computer programming.	3.68	4.21	3.80	4.10
I would like to study computer programming as my major.	3.20	3.72	3.33	3.80
I would like to program computers as a career.	3.37	3.74	3.56	3.80

Table 6.8: Student ratings of interest in and confidence with computing

Retention Rates We compared the retention rates of the fall 2007 course to courses taught by our partner in fall semester 2003-2006. We compared the overall retention rate, and found no significant difference between our pilot and prior years (Figure 6.7). We also examined the retention rate at the week-by-week level (Figure 6.8), by determining when a student last completed an assignment for non-zero credit. Note that this is different than when a student officially drops a course, but in some ways is more accurate, as it represents when the student stopped trying in

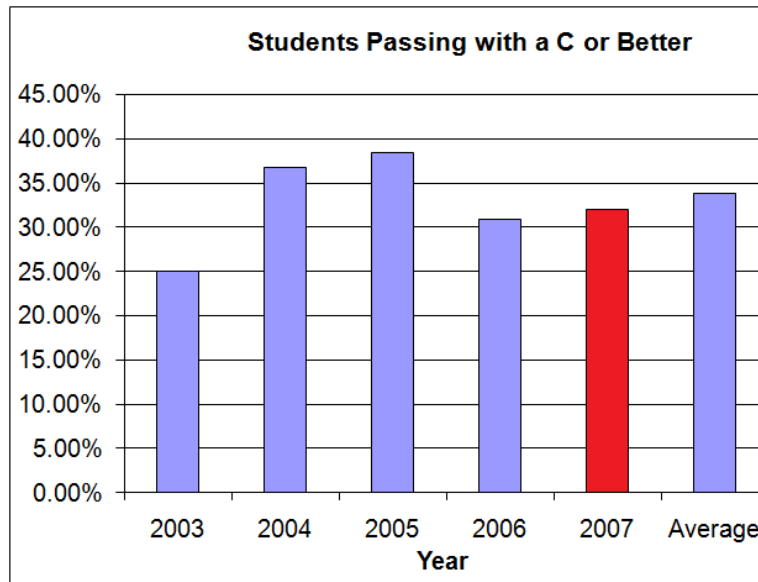


Figure 6.7: Overall retention rates in the fall semester CS1 course at CCAC

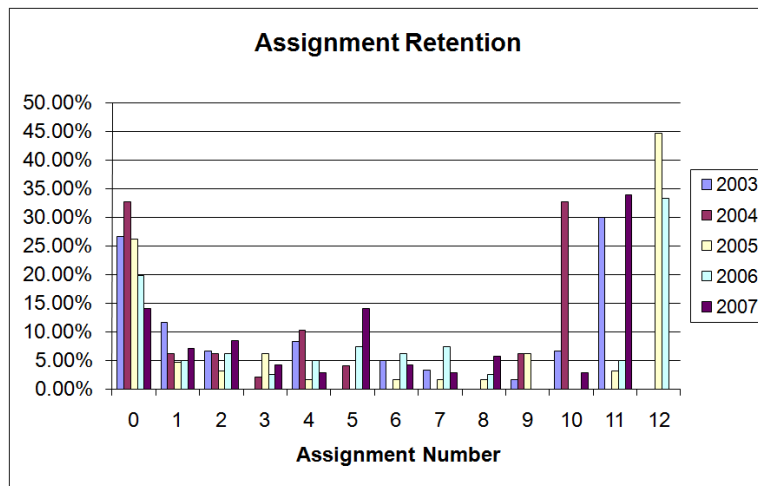


Figure 6.8: Week-by-week retention rates in the fall semester CS1 course at CCAC

the class. At the more detailed level, we also saw no real differences in the pattern of drops between the pilot and prior years. However, two important patterns are visible in the detailed data: First, on average about one-third of all students drop out before assignment 3, that is, before any robotics assignments had been handed

6. CSbots

out in the pilot. Second, there is usually a small spike in drops in the middle of the course (between assignments 4 and 7); this spike is caused by students dropping after receiving poor results on their first exam.

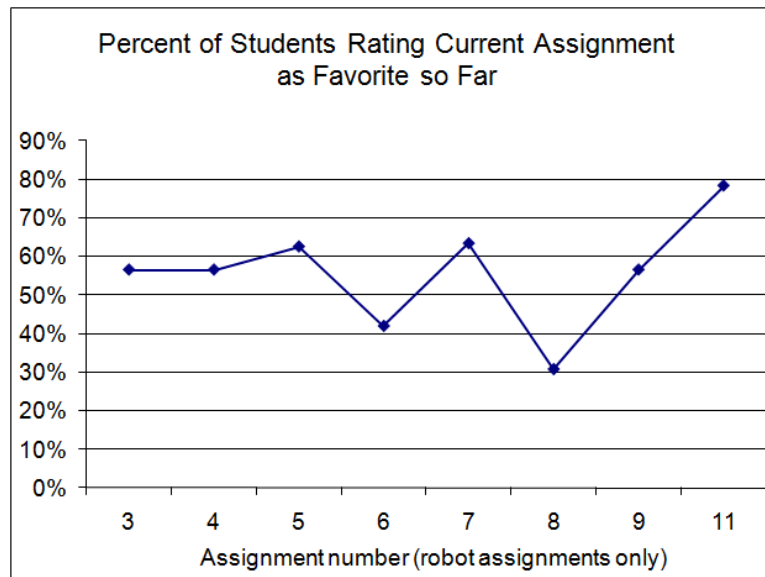


Figure 6.9: Percent of students who listed the current assignment as their favorite to date

Interest We measured student interest in the robot assignments by comparing their assignment completion rates to prior years and by directly asking them in post-assignment surveys. Student reports of the assignments via these surveys were generally favorable. Figure 6.9 shows the percentage of students who, after completing an assignment, rated that assignment as their favorite assignment to that point in the class. One would expect that the trend line in this figure would drop over time, as students have more assignments to choose from. Instead we see a flat to marginally increasing trend line, indicating that students who did not drop the course stayed engaged throughout. Although this response can be explained partially by recency effects, in the post survey a number of students wrote that assignments grew increasingly interesting, and that each new assignment surpassed the last in interest.

Confidence We measured student confidence in their performance on the robotics assignments by asking them what grade they thought they would receive on the

assignment they had just completed. Figure 6.10 shows the results on an assignment by assignment basis. Generally student confidence improved slightly over time; this may be due to stronger students staying in the course while weaker ones dropped out. There was a strong dip in confidence for assignment six. Assignment six was widely seen as a difficult assignment and marked the introduction of the programming concept of non-main methods.

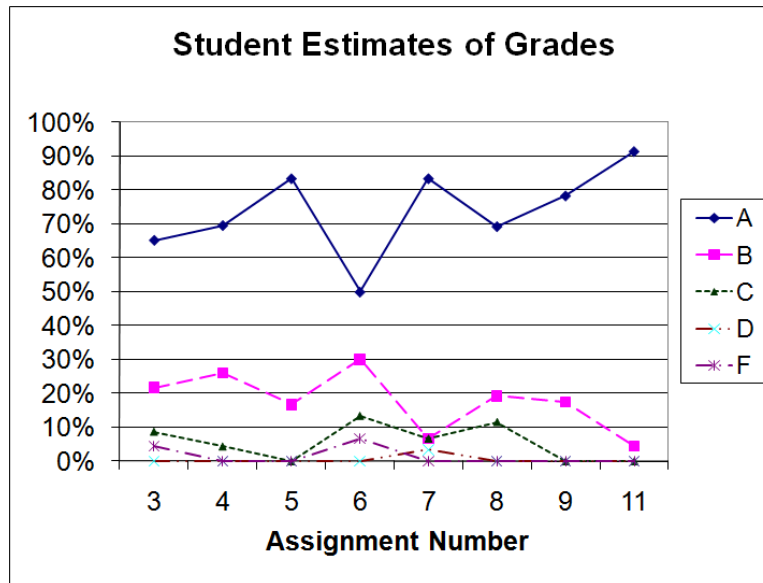


Figure 6.10: Student estimates of their grade on each assignment

Frustration We attempted to deduce potential ‘sticking points’ that would make the pilot assignments less engaging or more difficult by asking students to tell us the most difficult part of the just-completed assignment. We coded these responses into six categories. The percentage of expression of each is shown in Figure 6.11. The codes were:

- **conceptual** - Responses that gave a programming or CS concept as the answer.
- **compile+syntax+IDE** - Responses where students had trouble with syntax, using the IDE, or figuring out compiler error messages.
- **aesthetic** - Responses where the students responded that it was difficult thinking of non-programming creative elements; how the robot should dance, what it should say, etc.

- **lab times+testing** - Responses dealing with the logistics of the lab setup - generally that it was difficult getting to the lab during the open hours, and that there weren't that many hours for testing.
- **framework** - Responses dealing with bugs or usability issues in the robot software framework created for the pilot.
- **nothing difficult** - Responses stating that nothing was difficult about the assignment.

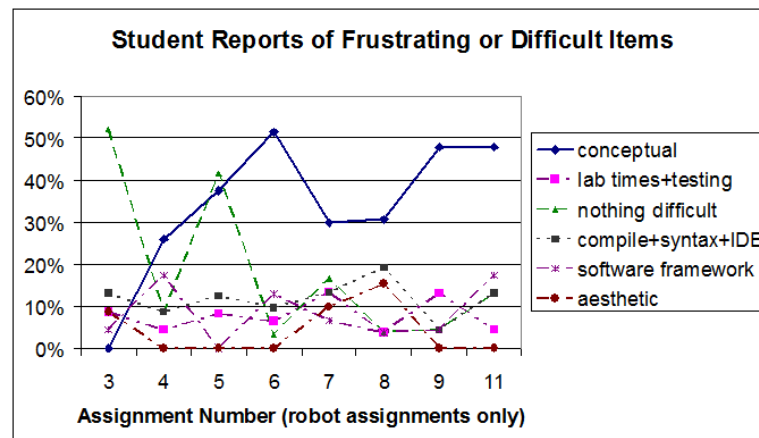


Figure 6.11: Student sources of frustration

Viewing Figure 6.11, it becomes apparent that conceptual issues were often the most difficult part of an assignment. From a pedagogical point of view, we feel this is the desired result; students should spend the majority of their time in a computing class struggling with computing concepts. We were also sensitive to any negative impacts of the robot on the student's ability to complete their assignments. Both the *lab times+testing* and *framework* codes represent difficulties that would not occur in a non-robot CS1 class. Fortunately, these responses were expressed fairly rarely, and combined make up less than 20% of reported difficulties. Also encouraging was the lack of responses indicating a problem with robot hardware; reflecting the fairly robust and failure-free operation of our robot platform.

Relevance We asked students if they saw any relationships or links between the program written for the assignment and the operation of software, computers, or computational devices. We were aiming to create assignments that would be relevant to students' lives so as to be engaging and motivating. Figure 6.12 shows

the percentage of students who linked the assignment content to an item in the real world. Students increasingly saw relationships between their assignments and the real world; this may be due to the increasing complexity of later assignments.

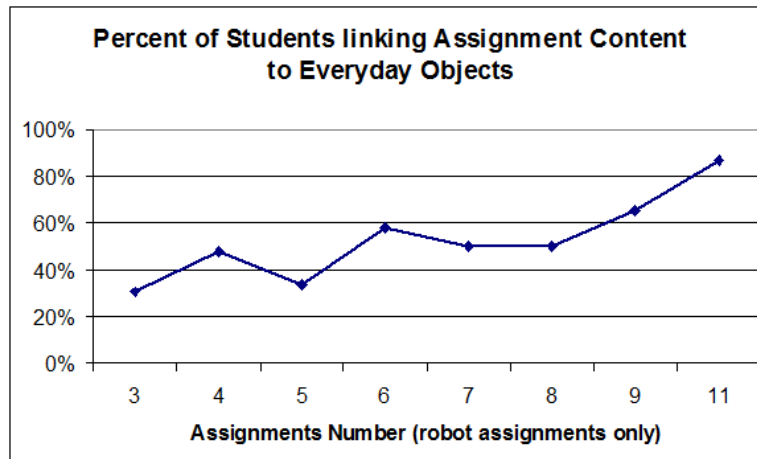


Figure 6.12: Percent of students who linked assignment content to the external world

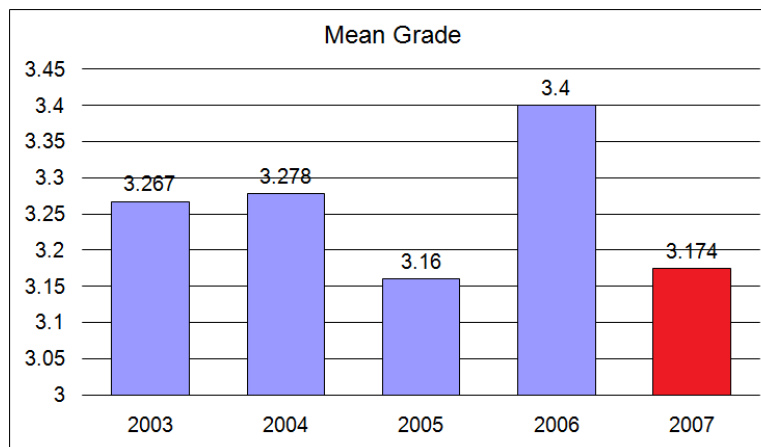


Figure 6.13: Mean grades of passing students, 2003-2007

Grades We compared the grades of students in our pilot to prior years to ensure that our students' learning was not hindered by the robotics assignments. The

grade structure of the class was such that 75% of the grade consisted of exam grades. Exams were modified only superficially (to prevent cheating) between our year and prior years. As such, we consider the performance on these exams and subsequent performance in the class as an adequate measure of comparative student learning. Figure 6.13 details the average grades in the 2007 pilot and four prior years. Passing students in the pilot performed at the same level as the previous years.

6.7.4 High School Pilots

While our initial focus was on introductory computer science courses in community colleges, in 2007 we expanded the program to high schools. Though high school was a somewhat different educational context, the program goal remained the same: to gather information for a redesign of the robot and curriculum that would be well-aligned to the CS1 course. Though the goal was the same, our approach to these pilots was significantly different from the community college pilots. In the case of the community colleges, educators had already provided much input into the initial design of the robot, software, and curriculum; thus the pilots were about collecting information about the impact of using the robot on students. In the high school pilots we decided to focus on working with the teachers to adapt the curriculum, and where possible, the robot hardware (the initial robot had free I/O ports and empty mechanical mount points, and so could be modified). In some ways, our work with high school teachers was similar to our initial engagement with community college partners - we were working with them to create a tool aligned to their curricula. The difference was that to a much greater extent, the initial robot and software API were already designed.

Recruitment

Our goal was to recruit five teachers to actively participate during the 2007-2008 school year. We recruited teachers to work with us by offering a short, one and a half hour session at Carnegie Mellon's CS4HS summer workshop (Blum et al., 2008) for high school teachers. During the workshop, we trained teachers how to use our alpha robot platform and associated software and announced that we were looking to partner with several teachers to develop curricula and inform hardware and software changes to the platform. Of the 35 teachers at the CS4HS session, 22 signed up for additional information about the partner program. We emailed these 22 several weeks later and asked them to complete a survey to help us determine their initial ability to work on the project; eight of the teachers responded to the survey and seven were able to partner with us. We sent robots to all seven, but two

of the seven teachers dropped out of the program after several months due to lack of time, leaving us with exactly the number of partners we initially sought.

Communication and Support

At the beginning of the project we realized that frequent communication between the partner teachers and project team would be necessary. As our partners were distributed over the continental US, face-to-face meetings could not occur. We began the year by mailing each teacher a robot and instructions on how to set up the robot at their school. Four teachers had no problems with setup, while the fifth had technical difficulties due to restrictive school policies on installing new software. These problems were resolved by calling the school's technical support.

Once teachers had received their robots, we kept to a schedule of monthly individual phone calls to get an idea of how each teacher was using the robot and to deal with any problems that might have arisen. These phone calls were crucial in maintaining the relationships between us and the teachers, as they created a series of monthly goals that drove both our group and the teachers to work on the curricula and software API.

In addition to phone calls, we also kept in touch using a wiki. Teachers used the wiki to provide information about their schools, upload pictures of themselves and their students, and later to share activities and assignments they had developed for the robot.

Structure of the Project Year

Teachers were sent robots at the end of September and most began using the robots as soon as they arrived. In addition to the robots, we also sent the assignments developed in Table 6.3. Our goals for the fall semester were to have the teachers set up the robot and attempt the assignments in order to become familiar programming with the alpha robot. Most teachers went beyond this, inviting their students to try some of the assignments and providing feedback regarding the activities, software API, and robot hardware.

December marked the end of the training phase of the program, as by this time everyone had tried a majority of the assignments and were familiar with robot programming. The relationship between the teachers and us became less prescriptive and more cooperative. Teachers began creating additional activities and assignments for the robot. As their comfort with the robot grew, all the teachers involved their students to a great extent. Despite having only a single robot, they were able to integrate robot activities into their spring curricula, and were able to use the robot extensively in out-of-class settings. As part of the purpose of our study was

to discover which hardware features were most useful to teachers and students, we sent supplementary kits of sensors and actuators and asked the teachers to try using them; these kits included light sensors, a distance sensor, and servos with which teachers could build robot arms. The teachers used these supplementary kits, and they, or in some cases their students, upgraded the robot hardware with them.

Summer 2008 Workshop

We held a workshop at the end of the 2007-2008 school year to train new teachers to use the initial robot, software, and curricula developed by us and the teachers who had worked with us over the past year. Although it may seem somewhat counterintuitive to hold a workshop around a robot platform that we acknowledge as an incomplete design, doing so has broadened and deepened our contacts with the high school teacher community, and the comments of these new teachers in the 2008-2009 school year have provided us with further feedback into the redesign of the robot.

The successes that our pilot teachers had with a single robot per classroom supported a belief that the program could be scaled, and we decided that a workshop after the initial project year was the best way to do so. We had support to host a summer 2008 workshop with up to twenty five teachers; as well as to subsidize the cost of robots for all attendees. The workshop was centered on training teachers on the robot used in the first year and on the curricular materials that were generated by the partner teachers and project team during that year.

Recruiting Participants We recruited participants by emailing the SIGCSE and the Collegeboard's AP Computer Science teacher mailing lists. We also worked with the CS4HS organizers to email their list of attending teachers. We co-located with CS4HS and held the workshop directly before CS4HS. This was essential to recruiting an adequate number of participants to a new and unknown workshop; as nearly all of our participants also attended CS4HS.

Twenty-four new teachers attended the workshop, and four of our existing partner teachers attended to share their experiences. Collectively, these teachers teach at least 1500 students per year. We were pleasantly surprised at the wide reach of the workshop which attracted teachers from 12 states and one foreign country (a teacher attended from the American Embassy School in India).

Workshop Composition The workshop took place over one full and one half day near the end of July and was composed of a number of lectures, panel discussions, and directed and open ended robot activities.

Two one-hour lectures were given during the first day. The first lecture was given at the start of the workshop and described the robot's hardware, explained the wireless network configuration, and detailed the software API necessary to write programs. The second lecture concerned the curricular activities and assignments already created, and explained how certain high level CS1 concepts mapped well onto certain features of the robot's hardware (for example, looping works well with reading in sensors).

Two 45 minute panel discussions were held on the first day. The panels consisted of the four teachers who took part in the pilot, allowing peer to peer discussion of common concerns related to teaching CS1 in high schools. The panel topics included:

- How the robot is used effectively at different schools.
- Challenges to setting up the robot in different contexts.
- Involving students out of class.
- How to maximize the limited resource of a single robot
- How the robot was used for community outreach.

Teachers worked on two directed robot activities during the workshop. These activities were relatively brief at 30 minutes each and allowed teachers to become familiar working with the hardware during the morning and early afternoon of the first day. The activities were based on some of the assignments created by our partner teachers during the pilot year.

The last hour and a half of the first day and the entire second day were devoted to allowing teachers to work in small groups to develop robot assignments and activities (and to write solutions to these with the robots present). This time was structured only in so much that a goal was presented to the participants: create one or more potential assignments that involve the robot. These activities produced a long term beneficial result for the community, as all the participants shared their assignments with the group at the end of the workshop. The open-ended structure also allowed groups to focus on areas of personal interest (for example sensing or text-to-speech), to work at their own pace, and to receive individual assistance from us and the experienced teachers on an as-needed basis.

Workshop Evaluation We evaluated the workshop through a post-workshop survey of participants. The goal of the post-workshop survey was to anonymously allow teachers to provide feedback about the workshop. The survey was conducted online, and 16 out of 24 teachers responded to our request to complete it. Our

conclusion from the survey feedback was that the workshop effectively trained the teachers in use of the robot and accompanying materials. Twenty one out of 24 attendees purchased a robot for 200 USD at the end of the workshop, further confirming the workshop’s effectiveness.

We asked teachers to rate a number of aspects of the workshop on a scale from 1 (Poor) to 5 (Excellent). Table 6.9 summarizes the mean ratings for each aspect. Overall, teachers were very positive about the workshop in their ratings, and were fairly consistent as well - almost all teachers rated every aspect of the workshop as either ‘good’ (a rating of 4) or ‘excellent’ (a rating of 5).

Workshop Aspect	Mean Rating
Technical lectures on robot, software, and curricula	4.6
Discussion panels	4.1
Guided hands-on activities	4.8
Open-ended activities/brainstorming	4.4
Overall workshop experience	4.8

Table 6.9: Summary of teacher ratings of workshop

We also asked teachers about our effectiveness at teaching them how to use the robot and curriculum developed for the workshop. Specifically we asked them to agree or disagree with the following statement: “The CSbots workshop gave me the appropriate information to effectively use the Create robot and associated curriculum at my school.” On a scale from strongly disagree to strongly agree, all teachers marked ‘agree’ or ‘strongly agree’.

6.8 Redesign

The evidence at the end of the initial design phase provided support for the notion that robots could be effective educational tools in Computer Science education. Our initial design was lacking in important ways: problems at Ohlone college emphasized the fragility of the hardware setup, the high student:robot ratio and resulting requirement for students to work on assignments on campus is poorly aligned to the CS1 course, and the feature set of the robot had not been optimized for computer science education.

Given these problems, we engaged in a redesign of the robot, software, and curriculum to align with CS1 goals. In this section we discuss the principles behind the new design and details of the design.

6.8.1 Design Constraints

We had many sources of information to consider in our redesign: our initial evaluation and literature review, our experiences with high school teachers testing different peripherals, our CCAC and Ohlone college studies, and much feedback on the initial robot from students and teachers. From these varying pieces of information, we formulated a set of evidence-based design constraints:

- The robot should be sufficiently low cost for individual use. This was borne out both from the original literature, from our own experience running a separate lab in the CCAC study, and from feedback from high schools teachers who were constrained to working with a single robot.
- It should be possible to complete assignments with the robot at home. This suggests three design constraints: Firstly, the robot hardware must be sufficiently robust to survive students' home environments and transport between home and school. Secondly, the robot must be small enough to be carried back and forth easily. Lastly, the robot software must work on the diverse computer and OS platforms students have available at home.
- The robot should be aesthetically appealing. At CCAC, researchers often observed students personalizing robots and giving them anthropomorphic agency when their programs were running on them. We thought this sense of agency important to motivating the students, and felt an aesthetically interesting design would support it.
- The robot should be capable of interesting interaction with students and with the environment. There was no one feature that we had to include based on our earlier experiences with the initial design; instead, a broad suite of inputs and outputs is most important.

6.8.2 Robot

Much of the project time in 2008 was spent formulating the above design constraints and deriving from them an appropriate robot hardware design. At the end of this time we had developed a small, inexpensive, highly interactive robot, which we dubbed the "Finch". The Finch design is unusual for a mobile robot: it must be tethered to a computer at all times. Deciding to tether the Finch provided a number of marked advantages to the design:

- The Finch derives power from the tether, thus there are no batteries to charge and robot behavior can not be affected by on-board power levels.

- The Finch has very little need for on-board processing. Instead, a low cost microcontroller sends and receives commands over USB to the computer. Programs created by the student run entirely on the computer. In this way, the Finch is more of a computer peripheral than an autonomous agent.
- As a USB device, it is relatively easy to create support for the Finch in different programming languages.
- Setting up the Finch is simple. One need only install a USB driver (available for all major operating systems) and use our cross-platform software package.
- Tethering reduces the need for complexity in the Finch, which in turn significantly reduces the cost. We estimate that the Finch can be sold for under \$100 commercially.

Due to its tether, the Finch is not a very capable mobile robot. This is not a reflection of a flaw in the design but of the design constraints themselves - capability as a mobile robot is simply not that important. Tethering provided a simple solution to meeting several of the design constraints simultaneously, but two important constraints are not addressed by the tether: aesthetics and interactivity.

Our group decided to work with an industrial design consultant to create a molded plastic shell for the Finch to both protect the robot electronics and to appeal aesthetically to students. The consultant drew up three shell concepts (see Figure 6.14) and we distributed these to the high school teachers involved in the project. They and their students voted on which they preferred and provided comments, leading to a final concept that borrowed elements from each of the sketches.

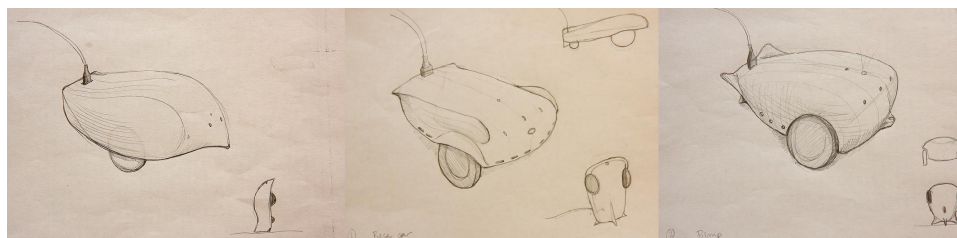


Figure 6.14: Sketches of Finch shell concepts

The final shell pictured in Figure 6.15 intentionally expresses zoomorphic traits, with a beak, two eyes, a nose with two openings, and ear holes. Naming the robot “Finch” is a further effort to carry on this zoomorphic theme. The shell is designed to be easily grasped by students, with the curvature of the shell suggesting that it

can be held with both hands with thumbs resting near the ear holes. Additionally, the rear part of the shell offers a tripod support to allow the Finch to be placed vertically on a flat surface. This is very useful when debugging a program in which the Finch's wheels are moving.



Figure 6.15: Finch shells

As the shell supports the notion of physically interacting with the Finch, the robot's hardware is similarly oriented around interactivity. From our initial pilots we realized that it was not so much a single sensor or actuator that was critical to successful interactions between students and the robot, but the combination of a number of features engaging students through multiple sensory paths. The robot could talk, flash lights, sense bumps, and move about. Successful assignments were those in which students needed to utilize most of these features in a way that also involved their interacting with the robot in some way (for example by bumping the bump sensor, listening to the robot speak and responding, or moving the robot through a GUI designed by the student). As such, we sought to turn the Finch into a highly multi-modal device, one that can interact with students by engaging a combination of students' visual, auditory, and kinesthetic senses. The Finch can express motion through a differential drive system, light through a color-programmable LED, and sound through a beeper. Similarly, it can sense light levels through two photoresistors, temperature through a thermistor, distance traveled through two wheel encoders, obstacles placed in front of it, and its orientation in three dimensional space through an accelerometer (see Figure 6.16 for placement of the Finch's actuators and sensors). In addition to these hardware-based capabilities, the accompanying software allows students to easily have the Finch speak or play songs over computer speakers, read real-time data from internet RSS

6. CSbots

feeds, and react to video from computer webcams.

Roboticists tend to think of mobile robots as creatures of their environment: they move within a space and react to stimuli in that space. Though the Finch is capable of behaving in this manner, the Finch's capabilities provide two other frequently used modes of operation. Firstly, there are programs that use the Finch as an input device to the computer; for example, the accelerometer data can be used to move a cursor on the screen. Secondly, the Finch can be used as a way to convey information from the computer to the user in the physical world; for example, by shivering if it is cold outside or setting an alarm when an earthquake has struck somewhere in the world. Using these three loosely defined and overlapping modes, the Finch expands on the ability of a computer to sense and act in the world. It should be seen as a tool for enhancing a computer's capabilities, providing students with a wider range of programs to write and problems to solve, and ultimately leading, as we have tentatively begun to see, to more interesting and relevant assignments and more engaged students.

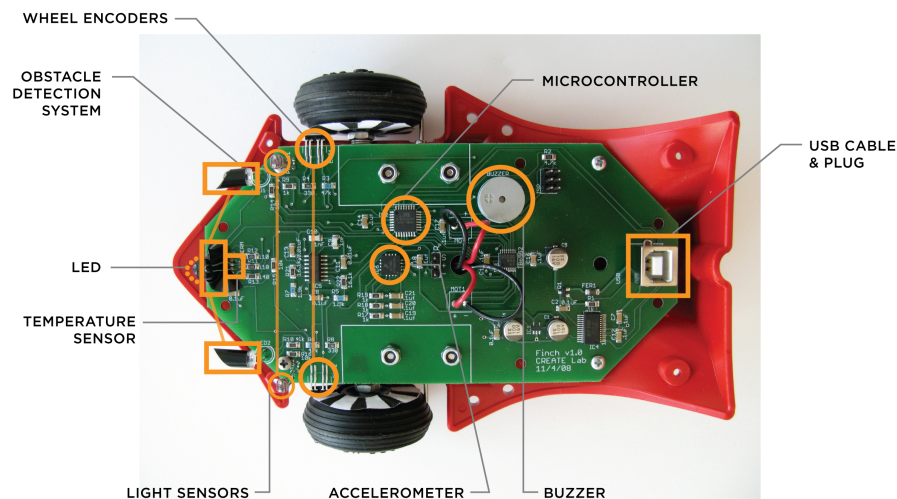


Figure 6.16: The Finch robot's sensors and actuators

6.8.3 Software

Significant changes were made to the software package, however, very little was changed in terms of how the software API was presented to students or how students used the API. Students still had a list of methods called from a single instance of a robot class. As we found that students in the initial pilots had few problems using our software API, many of the method names were the same as those used in the initial pilot. Changes to the software fell in one of two categories: new method calls to support entirely new capabilities of the Finch, and back-end changes to support the different style of tether (wireless vs. USB). The second of these was entirely invisible to students and our partner educators. The first category of changes was made in consultation with educators to name method calls for things like reading light sensors, determining spatial orientation, or setting beak color. Documentation for the software API was included with the software download students received, and as before, presented every method call in javadoc style in addition to the quick reference primer detailing the most important methods.

6.8.4 Curriculum

Unlike our initial design, specific assignments were not created outside the context of a specific course. At CCAC, the Finch was used throughout 2009 in two courses, CIT-111 (a CS1 course) and CIT-130 (a CS2 course); both of these were taught in Java. The Finch was used by students for most of the assignments in each course; however, learning goals did not change compared to prior years. We present the resulting curriculum from each of these courses in tables 6.10 and 6.11.

The Finch was introduced in the fourth week of instruction in CIT-111. The delay was deemed necessary by our partner to allow students to pick up basic Java syntax before beginning with the Finch. In CIT-130, no delay was deemed necessary and the Finch was introduced in week two. In both classes, the first Finch assignment was made somewhat easier to allow students extra time to set up the Finch at home and learn how to create small programs with the Finch. Furthermore, one lecture was dedicated to the Finch software API and how to call methods using the API. These were fairly minor disturbances to the overall curriculum and both courses did cover as much material as prior, non-Finch courses taught by our partner.

Assignment	Goals	Assignment Description
1	Compiling a program; printing to screen.	Write a program that prints to screen.
2	Variable types; simple arithmetic operators and Boolean expressions; printing variables.	Calculate and print miles per gallon given miles and gallons used.
3	Reading in user provided data.	Read in name and age and print remaining life expectancy.
4	Selection structures.	Read the Finch's orientation (flat, beak up, or beak down) and say what it is.
5	Looping	Move around the room and avoid obstacles. Apologize if there was a near collision.
6	Student-written methods.	Students create a note player method that plays notes read in from user input.
7	Student-written classes.	Write a class that tracks the internal 'emotional' state of the Finch and expresses it when the playEmotion method is called.
8	Arrays.	Move the Finch through five points and collect light sensor data at each point; store sensor data in an array and get the average, max, and min.
9	Begin graphics and events.	Create a tele-op interface.
10	Continue graphics.	Create a slider based tele-op interface to control LED.

Table 6.10: Schedule for CCAC's 2009 CIT-111 class

Assignment	Goals	Assignment Description
1	Review variables, operations, user input and screen output	Write a simple calculator program that asks for two integers and performs user-specified arithmetic operations on them.
2	Review using classes in context of Finch	Write a weather forecaster that takes user specified city, and then has the Finch interpret the current weather in that location.
3	Student written methods and looping	Students create a note player method that plays notes read in from user input.
4	Creating, catching, and handling exceptions	Create methods to have the Finch sing and dance to MIDI files; catch/handle exceptions if a specified file is not MIDI-format or doesn't exist at all.
5	Learn about arrays of objects	Create a game of Simon using the Finch - Finch says a position to move the robot to, and then the user must repeat.
6	Class Inheritance	Create a class that inherits the Finch class. This new class is called MoodyFinch and adds emotional state to the Finch.
7	Abstract and interface classes.	Use a provided interface class (the "Robot" class) and create an interface for the Finch with this class.
8	Graphics	Create a user interface for the Finch with movement commands, as well as buttons for several of the internet RSS feeds.
9	Continue Graphics.	Open ended user interface design with the Finch.

Table 6.11: Schedule for CCAC's 2009 CIT-130 course

6.9 Finch Pilots and Evaluation

In January 2009 we built 100 Finch robots and began a lending program with interested educators. To date, Finches have been evaluated by educators and students in high schools, after school activities, community colleges, and four year colleges. Reviews have generally been favorable. In two cases, we have collected formal evaluation data from these pilots; one was a full test at CCAC in which every student was given a robot, the other was a loaning program with nine high school teachers.

6.9.1 CCAC Pilot

Finches were used in the spring and fall 2009 semesters in the CIT-111 and CIT-130 classes (CS1 and CS2) taught at the Community College of Allegheny County by one of our partners. In the spring, three sections of CIT-111 and one section of CIT-130 used the Finch, and in the fall two sections of CIT-111 and one section of CIT-130 used the Finch. Every student in each section was loaned a Finch and could take the robot home with them.

Evaluation Methods

Students were asked to complete a pre survey, a post survey, and a brief survey after every assignment. Pre/post surveys sought to determine characteristics of students who successfully passed the course, as well as interest in using the Finch. Assignment surveys were designed to track student interest and frustration on an assignment by assignment basis.

In addition to these surveys, we also compared the spring and fall 2009 CIT-111 class to previous courses taught by our partner; we compared the average grade of passing students, as well as the retention rate in the class. The fall comparison included our earlier pilot with the alpha robot platform. We did not do a similar comparison for the CIT-130 classes because the low number of participants and intermittent teaching of the course did not provide enough participants to make comparisons statistically valid.

Survey Results

Survey results are provided for the spring and fall CIT-111 course and CIT-130 courses. Students were surveyed at the beginning and end of the course, and were given surveys after every assignment, with two exceptions: students in the spring CIT-130 were not given assignment surveys and insufficient students in the fall

CIT-111 course filled out our post survey, so their responses are not included. Survey completion was a challenge, especially on post surveys and on surveys given after every assignment. Due to IRB restrictions, we could not provide students with any incentive or requirement to complete surveys, and so survey completion rates were lower than desired.

Pre and post surveys sought to capture characteristics of the students in the course: GPA, age, favorite subjects, career interests, and interest in using robots. Assignment surveys sought to capture interest, confidence, and difficulties in completing the weekly assignments.

Participant Characteristics Table 6.12 presents the number of students enrolled in and taking the pre and post surveys in each of the four classes. From this table it is clear that response rates were above 75% for all surveys except for the post survey in fall CIT-111. This survey was supposed to be handed out immediately prior to the final exam, but it seems that students skipped the survey to work on the exam instead.

Class	Students Enrolled	Pre Survey Responses	Students Passing	Post Survey responses
spring CIT-111	59	47	13	12
fall CIT-111	40	33	17	2
spring CIT-130	21	20	11	10
fall CIT-130	14	10	9	7

Table 6.12: Students in the course and responding to pre/post surveys

We asked students their age and observed that in all three courses with pre and post surveys, the average age of students taking post surveys was three to four years older than that of students taking pre surveys. Table 6.13 presents these results, which seem to imply that older students are more likely to pass the course, either due to past experience programming or to a greater degree of responsibility.

To gauge whether prior experience with a language was important to success in the class, we asked students in CIT-111 to state which programming language they knew best. Table 6.14 presents the percentage of all students who selected a given language at pre and post, and the results of a matched cohort who answered this question at both pre and post. From this table, prior experience with programming

6. CSbots

Class	Average age at pre	Average age at post
spring CIT-111	24.43	28.00
spring CIT-130	28.95	32.67
fall CIT-130	23.50	27.00

Table 6.13: Average ages of enrolled and passing students

was not a determinant in success in the class.

Language	All Pre (n=47)	All Post (n=10)	Matched Pre (n=10)	Matched Post (n=10)
Java	9%	70%	0%	70%
C/C++	4%	10%	0%	10%
Other	13%	20%	40%	20%
None	74%	0%	60%	0%
Experience Rating	2.35	2.58	2.17	2.1

Table 6.14: Percent of students in CIT-111 reporting each category as their best-known programming language

Reasons for Taking the Class We asked students two questions related to why they chose to take the course. To get at their motivation for selecting a Computer Science course, we asked them to list their favorite school subjects. To get at what they needed the course to provide them, we asked them their reasons for taking the course.

Subject	Pre (n=80)	Post (n=14)
Computer Science	50%	64%
Engineering	9%	14%
Other Science	25%	14%
Social Science	13%	8%
Health Care	4%	0%
Other	10%	0%

Table 6.15: Percent of students in CIT-111 reporting a given topic as their favorite subject in school

Results from the question regarding students' favorite subject are shown in tables 6.15 and 6.16. The answers to the questions have been split by course but

Subject	Pre (n=30)	Post (n=17)
Computer Science	57%	82%
Engineering	7%	0%
Other Science	18%	12%
Social Science	18%	6%
Health Care	0%	0%
Other	0%	0%

Table 6.16: Percent of students in CIT-130 reporting a given topic as their favorite subject in school

not by semester. While the question was open-ended, answers were coded into one of six categories; computer science, engineering, other science, social science, health care, and other. Some students responded with two or more subjects - in these cases we counted all responses; thus some columns in the table add up to over 100%. In general, students answering the post survey were more likely to be interested in Computer Science and students in the CIT-130 class showed more interest in Computer Science; these differences were not statistically significant. We also tracked changes in responses from students completing both pre to post surveys. Across both classes four students changed their answers from pre to post - all four moved from another category to Computer Science.

Interest in the Finch We sought to determine interest students had in the Finch through three questions. We asked incoming students if they would prefer programming a computer or robot, and asked outgoing students if they had shown the Finch to anyone, and if they had worked on programs for the Finch that were not assigned.

We coded the answers to the programming preference question into three broad categories - robot better, computer better, and not sure. Table 6.17 summarizes the answers to these questions split by class. Generally, roughly 2/3 of incoming students believed they would prefer programming a robot to programming a computer, with the third split between being undecided and believing they would prefer programming a computer. Students were asked to give reasons for their answer; popular reasons for answering computer better were that learning to program a robot was too specialized and might not apply to future classes or careers, or that computers are more useful. Students answering not sure almost always indicated that they had no experience with robots and so could not judge whether programming robots would be more or less interesting. Students answering robot better gave three main reasons: robots are cutting edge or novel, robots provide the stu-

6. CSbots

dent with the ability to have their programs physically act in the world, and robots are fun. There were no significant differences between the semesters or between students in CIT-111 and CIT-130, although there were more students in CIT-130 who believed programming a computer would be preferable. These students almost always cited a worry that programming for the robot would not be useful in later classes or in their career.

Class	Robot Better	Not sure	Computer Better
CIT-111	68%	18%	14%
CIT-130	66%	10%	24%

Table 6.17: Percent of students who prefer to program computers or robots

At the end of the course students were asked two questions about the Finch - whether they had shown it anyone, and whether they had written programs for the Finch that were not assigned. Every student who answered the post survey indicated they had shown the Finch to someone - typically friends and family. Most went on to describe the reactions of the people who saw the Finch; of these reactions, all but one was positive.

Students were less likely to have written programs for the Finch for fun, of twenty eight post survey responses, eight indicated that they had done so. Most of these students wrote small programs for their friends and family; four students gave an estimate of the amount of time they spent out of class on the Finch. Three students spent 3-5 hours on the Finch, and one indicated that she spent 30-40 hours out of class using the Finch.

Interest Interest in the assignments was tracked through surveys after each assignment using the same question as in the 2007 pilot. Students were asked to rate their favorite assignment to date, and we then charted the percentage of students rating the just completed assignment as their favorite. Graph 6.17 shows the trend lines for each of the three courses with assignment surveys (Spring CIT-111, fall CIT-111, and fall CIT-130). The latter half of the spring 2009 semester appears to have been less than compelling to students; and so we retooled our assignments for the fall with positive results. The fall CIT-130 similarly benefited from our experience with the spring course. In all cases, a final assignment mixing graphical user interfaces with the Finch was very popular among students (just as it was in our earlier 2007 pilot).

Confidence Confidence in the assignments was tracked by getting estimates of the grade students expected. Charts 6.18, 6.19, and 6.20 show the general trends

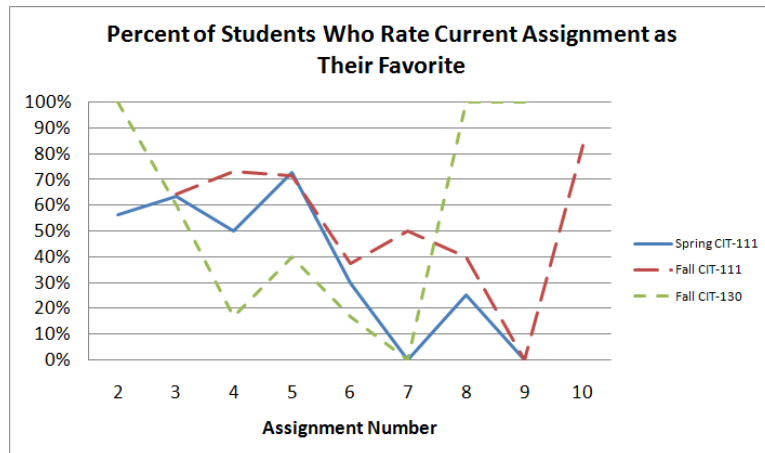


Figure 6.17: Percent of students rating the current assignment as their favorite

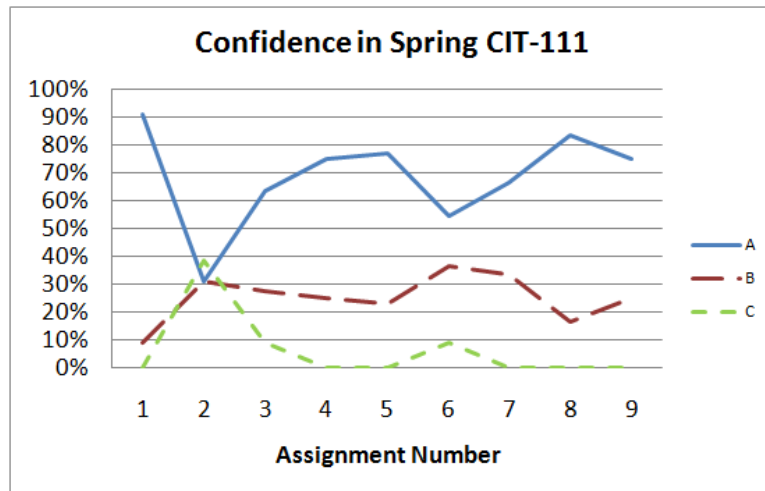


Figure 6.18: Confidence of students in spring 2009 CIT-111

in student reports of their own grades. Student estimates of their grades did not change markedly during the semester, though some assignments appear to have been more difficult, and students in the CIT-130 class estimated lower generally lower grades, possibly because the subject matter of the second level class is more difficult.

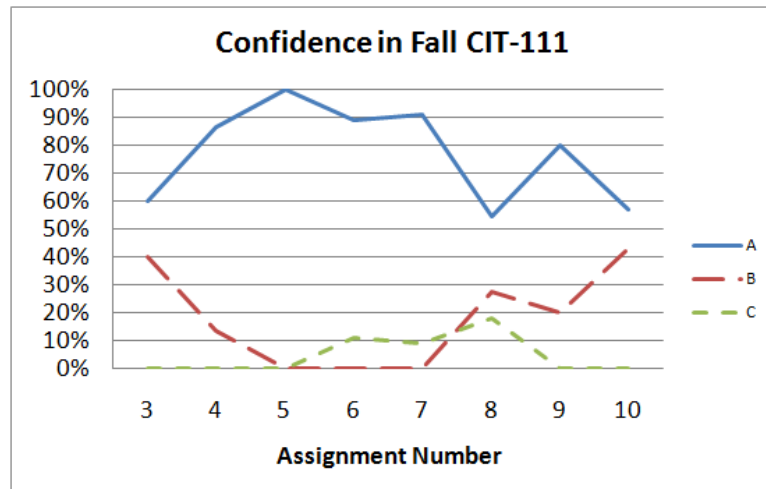


Figure 6.19: Confidence of students in fall 2009 CIT-111

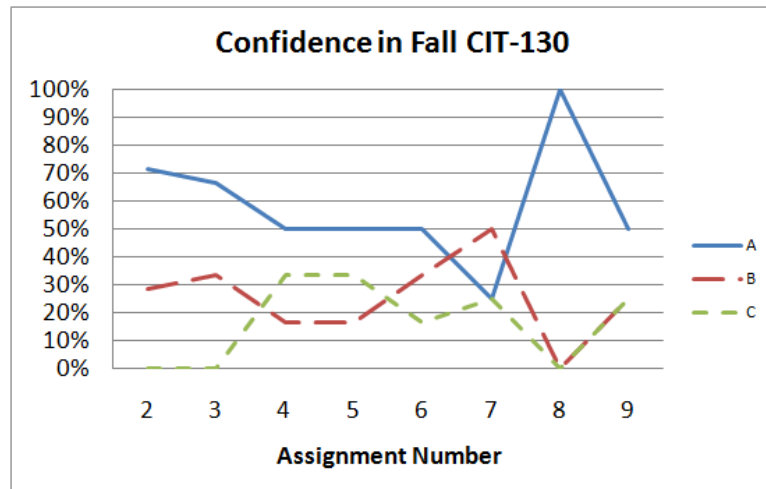


Figure 6.20: Confidence of students in fall 2009 CIT-130

Frustration Students were asked to summarize the most difficult or frustrating problem they encountered for each assignment. We coded these sources of frustration into several categories:

- **Conceptual** - Students mention not fully understanding concepts of programming like looping, arrays, etc.

- **Syntax** - Students report problems getting the program to compile because of syntax errors or program structure.
- **Finch API** - Students have trouble getting the Finch methods documented in the API to work properly.
- **Finch Hardware** - The Finch hardware prevents the written program from being properly executed.
- **Assignment** - Students had trouble understanding what the assignment was asking for.
- **None** - Students reported that nothing was difficult in the assignment.

Ideally, students would have struggle only with conceptual problems or have no problems at all. Struggling with the Finch API is not necessarily negative; many students who had problems with the API actually had problems with the concept of using external custom classes. We sought to minimize problems stemming from the Finch hardware, from the assignment description, and from problems with the API specifically related to confusing documentation or bugs in the Finch software.

We charted the frequency with which each of these sources of possible problems were reported by the students. Figures 6.21, 6.22, and 6.23 display the results.

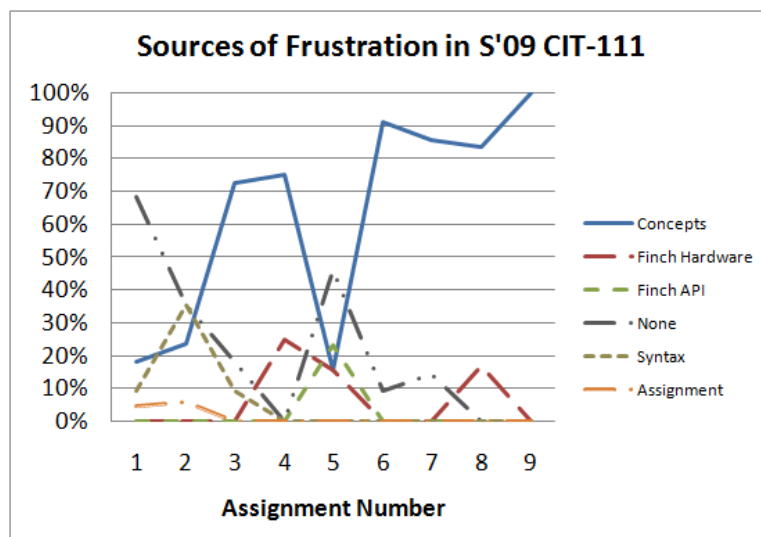


Figure 6.21: Sources of frustration for students in spring 2009 CIT-111

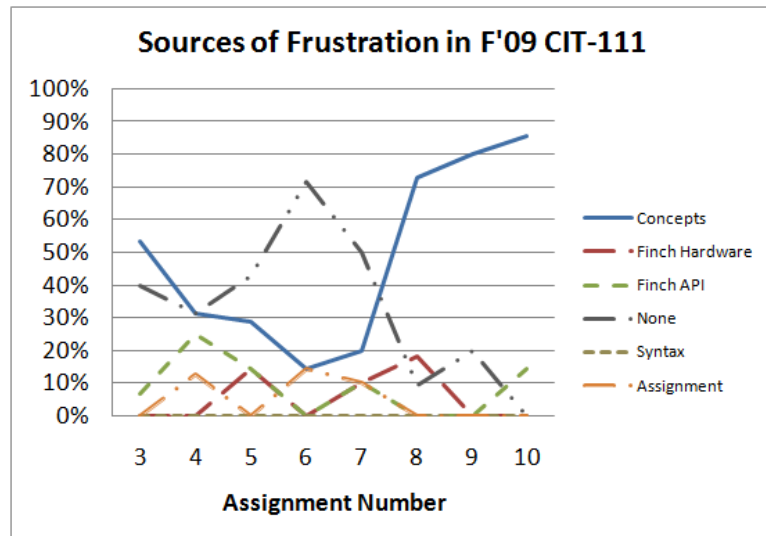


Figure 6.22: Sources of frustration for students in fall 2009 CIT-111

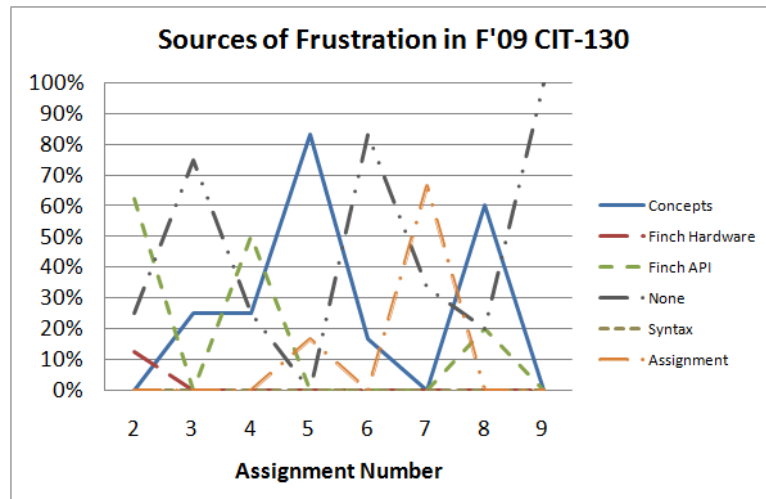


Figure 6.23: Sources of frustration for students in fall 2009 CIT-130

Most students primarily reported conceptual problems throughout the course. However, there is a spike of Finch API problems at the time when the Finch is first introduced, representing problems students have when initially learning to use the Finch. Fortunately, students rarely report Finch API or hardware problems after using the robot for one or two assignments. Finch hardware problems were uniformly

related to the poor drive mechanism of the prototype Finches; this was more a problem of properly setting student expectations than of the hardware. Understanding the assignment description was typically not a problem, with the exception of assignment 7 in CIT-130; this highlights the utility of this question in finding areas of improvement for the course.

Finch Robustness We tracked failures of the Finch hardware over the course of the year. There were very few failures of any type, out of 60 Finches loaned out for the entire year:

- One Finch beak LED did not light up correctly due to poor soldering during assembly.
- On three Finches, screws came loose in the drive mechanism, preventing a wheel from turning.
- One Finch's USB cord was destroyed by a small dog.
- Three Finches were not returned by students.

Of these errors, only the LED failure is irreparable by students, and this was a manufacturing defect and not a result of use. As such, the Finch robot held up very well, meeting our goal of creating a hardware platform that does not introduce hardware problems into a Computer Science class.

Comparison to Prior Courses

We compared the mean grade of passing students and the retention rate in the course to previous years of the CIT-111 class. The CIT-130 class was too small and offered too intermittently to make comparisons to prior classes statistically meaningful. We compared the full year of sections to prior years and also compared the spring and fall sections separately. This is done because our experience in the spring course allowed us to improve the software API and assignments provided to students in the fall course substantially. As such, we believe that the fall course is a better-aligned course generally, and makes better use of the Finch's capability as a tool in this context.

Retention Retention in the spring and fall pilots was compared to four prior years of data from our partner; this also includes our earlier pilot in fall 2007. Figure 6.24 shows the retention rate, defined as the percentage of total enrolled students who passed the course with a C. The three bars for each year represent the spring, fall,

6. CSbots

and total retention rate for that year; blue bars are years without any intervention, the green bar is the earlier pilot, and the red bar represents data from the Finch pilot year. An average retention rate for all non-pilot years is also included. Total retention across all five years was remarkably stable - hovering between 31 and 34%. Overall retention was not significantly different for the fall or spring pilot compared to previous fall or spring courses, nor was the entire pilot year significantly different from any previous year or the average of previous years.

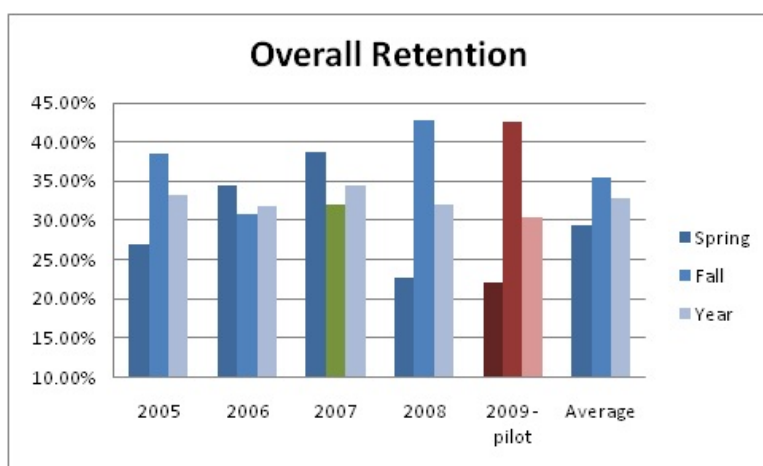


Figure 6.24: Retention of students in CIT-111 in pilot year compared to four prior years

In addition to comparing overall retention, we also investigated the retention rate among students who stayed in the course through the first exam. We did this for two reasons: Firstly, many students who begin the CCAC course drop out in the first few weeks because they had a poor understanding of the goals of the course; many students believe the course is about computer skills. Secondly, the Finch was not introduced until assignment 4, roughly at the same time as the first exam, and so it likely had no impact on retention before the first exam. Comparing the pilot year in this way we see that the spring pilot slightly underperformed compared to the prior four years; the retention rate of 52% was lower than the rate seen in three of the four prior years and was slightly below the four year average of 56%; even so, none of these differences was statistically significant and it should be noted that the retention rate improved compared to the year immediately prior.

The fall course showed a marked improve in retention compared to all other years and the average. The 80% retention rate was significantly better than 2006 and the earlier pilot year of 2007 ($p < 0.01$ for both). Though not statistically

significant, the rate was above the four year average of 57% and above 2005 and 2008 as well. When comparing between the study year and the prior four years, we see no significant differences in the retention rate, although the rate for 2009 is higher than for all other years (65% compared to an average of 56%).

It should be noted that achieving statistically significant results in this study was fairly difficult due to the low number of participants. Fall retention rates needed to be at least 25% higher than prior years to show significance. As such, the fall pilot results are extremely promising and suggest that future studies with more students should be carried out to more clearly quantify retention improvements stemming from Finch use.

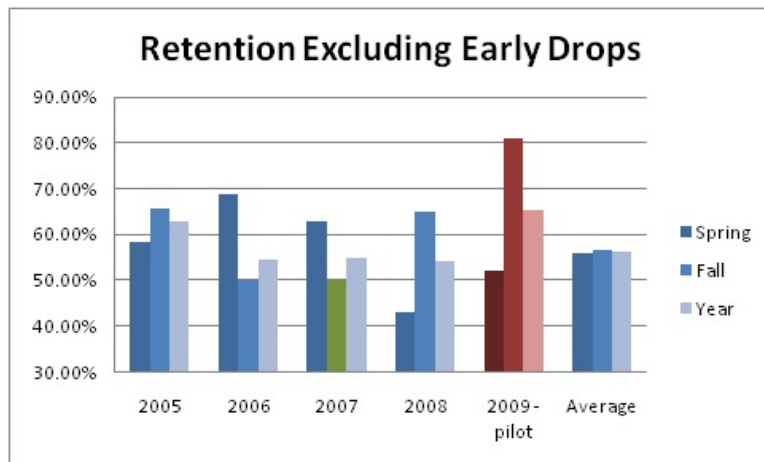


Figure 6.25: Retention rates excluding students who dropped before exam 1

Grades We compared the average grades of passing students in the pilots to prior years. We converted the letter grades assigned to students to numbers using the standard way in which GPA is calculated (i.e., a C is 2.0, a B is 3.0, and an A is 4.0). CCAC does not provide +’s, -’s or percentage grades. As the minimum grade required to pass was defined as a C, the average could fall anywhere between 2.0 and 4.0. The average grade in the class was not significantly different from any previous year, nor were the individual spring or fall semesters different from any previous spring or fall semester. However, the spring semester average grade was lower (though not significantly so) than all prior years. We suspect this may have been due to our inexperience using the Finch. The fall average grade was within the normal range of grades even with a higher overall retention rate, so we suspect that our experience with the spring course and subsequent modifications to

the assignments resulted in a qualitative improvement to the course.

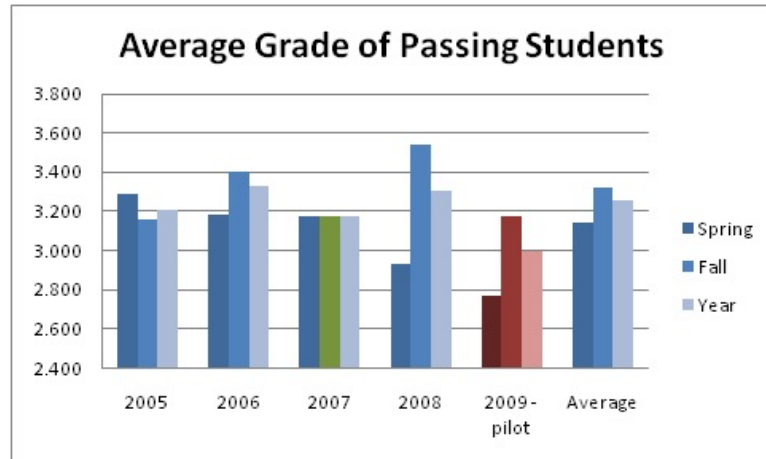


Figure 6.26: Grades of passing students in CIT-111 compared to four prior years

Summary

The 2009 CCAC pilots suggest that our revised design met some of our objectives: retention improved markedly in the fall course as compared to prior years, students continued to struggle primarily with computer science concepts as opposed to robot hardware/software, the Finch robots had very few hardware failures despite extensive use by students at home, and students passing the course demonstrated excitement in the class by showing the Finch to friends and family and by working on programs recreationally. At the same time, the relatively poorer performance of the spring course demonstrates the importance both of properly aligning assignments and activities to make use of the configurable embodied interface, and of past experience teaching with a new tool like the Finch.

6.9.2 High School Pilots

Teachers who completed our workshop in summer of 2008 and had received an alpha robot were given an opportunity to borrow a single Finch robot from us at the start of 2009. Ten teachers responded affirmatively and received Finches, providing us with a way to compare our initial robot design and the Finch in the hands of the same teachers. We asked participating teachers (both with and without a Finch) to complete a survey in June 2009 about how they used their robots to support their curriculum and program.

Fifteen teachers participated in the survey who had alpha robots. We also asked teachers who had been provided with a Finch to complete a separate survey; 8 teachers responded. As before, most teachers used the robot as a recruitment tool. 11 of the 15 teachers mentioned this as a use. Nine of the 15 teachers had students actively using the robots; in a few cases, these students were given full control of the robot for a few weeks or a month to do an extended project. It seems that gifted and advanced students were the main users of the robot. Teachers with a Finch also reported use primarily for recruitment and with small groups of students.

The total number of students using the alpha robot was 37, with most teachers who used the robot with students reporting groups of four to six using the robot. Among teachers who had also received a Finch, a total of 56 students used the Finch, with an average of roughly 10 students per teacher using the Finch in spring 2009. It appears that the Finch, perhaps because it is more portable, was used by a larger number of students for smaller, shorter projects.

Technical problems were somewhat rare and were typically resolved by the teacher. Problems were fairly idiosyncratic, with no teacher reporting the same problem as someone else. The problems reported for the alpha robot were: poor battery life, failed charger (a replacement was sent), setting up the software environment, compatibility problems with Mac OS, struggling to set up the hardware properly, bumpers that didn't respond when hit, and getting wireless networking working. These problems, in all but one case were temporary and resolved by the teacher. Teachers who had a Finch generally reported fewer problems, and no hardware-related problems at all. Three of the eight responding teachers indicated some level of technical difficulty. One had trouble setting up the USB driver on their Mac, and two had trouble using software IDEs with the Finch. All three teachers resolved these problems on their own without contacting us.

To summarize, these preliminary results from teachers compare the Finch favorably to the alpha design. The Finch is one-tenth the price, has fewer points of failure, and was used by a larger number of students in each class.

6.10 Next Steps

Three efforts are underway in 2010 to expand and sustain the CSbots program: new work with Pittsburgh-area charter schools, support for additional programming languages, and commercialization of the Finch.

6.10.1 Charter Schools

We are working with local charter and public schools to adapt the robot, software, and activities to the CS curriculum at each school. Participating teachers will be trained in how to program the Finch, and we will then work with them for a semester to create curricula and software appropriate to their schools. After this training period, each school will receive a classroom set of Finches that will be used during the 2010-2011 school year. As such, this will be the first large scale deployment of Finches at the high school level.

Not all of these schools have traditional CS1 courses, and so for the first time in the CSbots program, we will have the ability to design our software and activities towards learning goals that are somewhat flexible. Additionally, as each teacher will choose how to introduce the Finch, we hope that the range of activities and curricula developed will be fairly broad, reflecting the diversity of the different schools in the program.

6.10.2 Additional Language Support

As the Finch is a tethered computer peripheral, supporting additional programming languages is a fairly easy task. Our eventual goal is to provide support for the Finch in four types of programming languages:

- **Drag and Drop.** Languages like Scratch and Alice provide a graphical interface to allow students to drag and drop elements into a program. Children as young as eight can use these languages to successfully create programs, and they are used in formal education from elementary school through college.
- **Engineering Flowchart.** Engineering flowchart languages like Labview or Matlab Simulink allow representation of a program as an engineering flowchart. This a popular way to program in engineering schools, and for those who have visual learning styles. The Finch has preliminary support in Robolab.
- **Scripting.** Scripting languages like Perl or Python are in some ways as powerful and flexible as many modern traditional languages but are syntactically lighter and have a generally faster learning curve. They are becoming more popular in introductory Computer Science courses.
- **Traditional.** Traditional languages like Java or C are the most powerful of the languages we aim to support, and are still dominant in high school and college education. The Finch supports Java, and can easily support C.

In addition to these languages, we plan to create a program that will allow very young children to use the Finch. This program will allow programming of the Finch through gesture. The program would have two modes: record and play. In record mode, the child could pick up the Finch and tilt it in different directions to “program” different motions. In play mode, the Finch would then move through those motions. We have informally tested this program with kindergarten students and they understood the program operation in less than a minute and greatly enjoyed the program.

If we succeed in providing support for all these language types, we will have provided Finch language support for learners of any age, and for every type of programming method currently used.

6.10.3 Commercialization

In order to grow the program further and establish a sustainable source for the Finch robot, we are creating a company to sell the Finch. As the educational activities and software offerings available increase the potential market and sale-ability of the Finch, this company will have a vested financial interest in both disseminating the created educational activities, fostering a community of educators to create additional activities, and continuously updating and supporting associated software.

6.11 Summary

The CSbots program started with a fairly simple idea: use robots as targets for programming activities in introductory Computer Science education. Thousands of implementations of this idea were possible early on in the design process. We could have focused on producing costly, cutting edge robots that students could only program in closed labs. We could have decided to alter the learning goals of the introductory class to align better with our notions of good pedagogy in Computer Science. We could have decided to have programs run autonomously on the robot, necessitating the use of a customized programming language. All three of these potential facets of the implementation were discussed and decided against because of things that we learned during the initial evaluation and first pilot of the design. Thus, these decisions were made not because of technical constraints but because of educational constraints that were discovered during the design process. In the next chapter I expand on this topic and offer a general way for designers to discover these crucially important educational constraints.

Chapter 7

Summary, Analysis, and Conclusions

The three projects presented in chapters four, five, and six differ greatly in audience, setting, and learning goals. For all these differences, they share an alignment-centered design process concerned with creating a configurable embodied interface, software, and curricula, that are well aligned with a set of learning goals. This section details this design process in the abstract, decomposing it into conceptually different steps. I then describe several efforts that have used design processes similar to alignment-centered design, leading to a discussion of how alignment-centered design modifies a core engineering design process. I conclude by discussing some of the challenges and limitations of alignment-centered design, and provide guidance to future designers.

7.1 Summary of the Design Process

Figure 7.1 details the phases of the alignment-centered design process employed in each of the three previous chapters. I begin with ideation, followed by an initial evaluation, a constraints-finding process, and then one or more iterations of systems alignment cycles consisting of design, pilot, and evaluation steps. At the end of each alignment cycle one decides to either begin a new cycle or disseminate the design. The following subsections describe each of these steps in detail and summarize how each phase was experienced in the case studies.

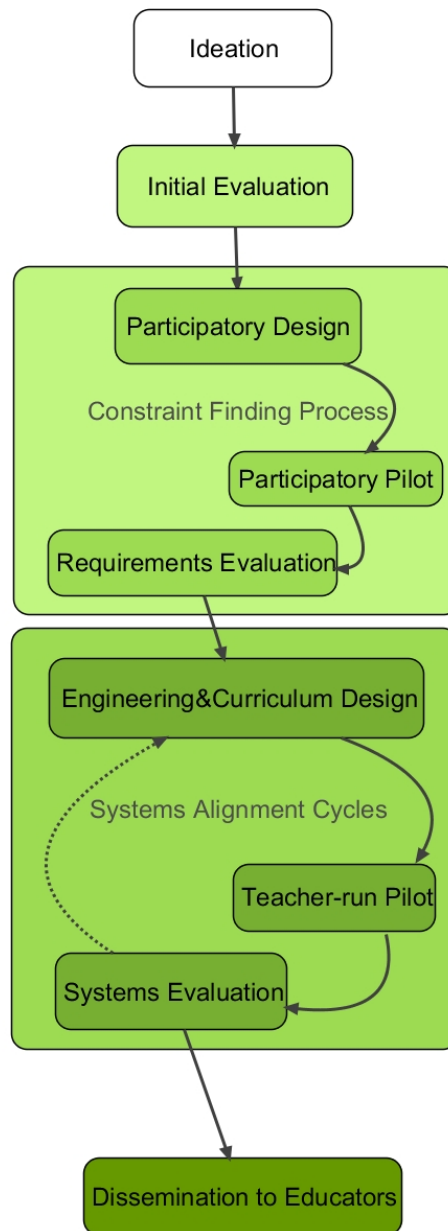


Figure 7.1: Alignment-centered design process steps

7.1.1 Ideation

Ideas spring from numerous sources; they often strike in moments of idle reflection, but generally require an informational trigger to have occurred at some prior time - perhaps a conversation with an educator about the needs of students or the sudden realization that a new technology has enabled a previously infeasible idea.

In the case of the Braille Tutor, the idea originated from a back-and-forth conversation between a designer (Nidhi Kalra) and teachers at the Mathru School for the blind. The conversation took place over email and focused on the challenges that students at the school have; the topic of learning to write eventually came up and Nidhi proposed the idea of a writing tutor. It is important to note that both participants in this conversation contributed to the generation of this idea - Nidhi did not understand the way in which students learned to write Braille, and teachers at Mathru did not know about automated tutors.

In the case of Robot Diaries, the idea was triggered by a conversation between two of the team's members about what kind of robotics activity they would have enjoyed when they were middle schoolers, and the realization that such an activity did not exist.

The idea behind CSbots was less novel - we were aware of other efforts to use robots in Introductory Computer Science. However, we planned to approach the problem in a way that we did not believe had been tried in earlier work - designing a robot from the ground up for the course instead of adapting commercially available platforms. In this case the idea came from external drivers - many resources were being poured into improving Introductory Computer Science due to the retention and enrollment problems of CS education.

7.1.2 Initial Evaluation

Irrespective of source, the first step in each of our projects was to evaluate the idea, typically by discussing it with educators and/or students. There are a number of ways to do this - each of the three projects used a different approach.

The Braille Tutor idea came directly from a conversation between a designer and educator, and so the initial evaluation of the idea was part of the ideation process. As soon as it was proposed, the educator was able to provide feedback to the designer regarding how it may be implemented and whether or not they could see it working in their context.

To evaluate Robot Diaries we assembled a focus group of students and presented our initial ideas to them. Our experience with the focus group led to significant revisions in both our planned activities for the summer and our approach in working with our intended audience.

7. Summary, Analysis, and Conclusions

We conducted a fairly extensive initial evaluation of the CSbots program - interviewing more than thirty CS1 educators around the U.S. and using those interviews to determine the logistics of the course, as well as interest in using robots. This survey led to a number of conclusions that strongly influenced our initial design.

7.1.3 Constraint-Finding Process

Design is, in a broad sense, the collapsing through design choices and found constraints, the design space or set of possibilities until an option that meets all the design constraints in a satisfactory manner can be created. The initial evaluation is the first step in the process of finding constraints to guide the direction of the design. The primary goal of the constraint-finding process, then, is to find additional constraints inherent to the educational context and users; these can be learning goals, required hardware features, instructional activities, or interactions between the hardware or software and the student. The constraint-finding process is composed of three overlapping phases: Participatory design, participatory pilot(s), and requirements evaluation. Though the three phases are not always well delimited in practice (for example, it is possible to still be adjusting elements of a design during a pilot, or evaluating a pilot before it is over), they are separated here to provide an idea of the kind of work done in each phase.

Participatory Design

The interface and supporting curriculum can be designed contemporaneously, as they are components of a unified design. Depending on the context, the interface may be enabling a novel curriculum or augmenting an existing one. In both cases, a participatory design partnership with students and/or teachers in the relevant educational setting is crucial. If creating a new curriculum, it is important to have ready access to partners who can generate ideas, test activities, exercises, or components of the design. For the initial technological design, it may be best to include every feature that appears as though it could reasonably be of use in the curriculum, with the intent that the pilot and evaluation will determine which features correlate well with educational outcomes.

In the context of the Braille Tutor, the initial hardware design was completed prior to the pilot. Portions of the software were completed ahead of the pilot, but the three tutor programs were written by a member of the design team at the beginning of the pilot in India. Educational activities for the Braille Tutor were designed concurrently with the tutor by partner educators.

In the case of Robot Diaries, we held extensive participatory design workshops with students. In Robot Diaries the pilot and design phases of the cycle were nearly

concurrent. The goals of our workshops were to discover how best to present the technical content and themes we wished the Robot Diaries activity to involve. We prepared for the workshops by designing kits of technical and building materials, a software environment for programming robots, and a number of activities. Some of the activities were created to test ways of introducing technical content, while others were specifically made to elicit feedback from the girls and allow for their participation in the design process.

For CSbots, the robot hardware, software, and curriculum were mostly designed concurrently, with feedback from partner educators and from a small pilot study with students at CCAC (see section 6.7.1). Our philosophy with the robot hardware was to develop a maximally instrumented platform coupled with assignments that took advantage of all of these features to allow us to determine what robot capabilities worked in the assignments and which did not. In addition to design work done prior to the pilot, we also revised or rewrote some of our planned assignments during the CCAC pilot to dynamically adjust to student interests and needs. Compared to Robot Diaries or the Braille Tutor, these in-pilot design refinements were very minor; the phases of the CSbots design cycle overlapped much less than the other two.

It is important to note that with Robot Diaries and CSbots, there were elements of the design that existed specifically to allow us to find additional constraints; the activities in the Robot Diaries curriculum to elicit feedback and the overbuilding of robot capabilities in CSbots were integrated into the designed whole with the expectation that a final design would not include these elements.

Participatory Pilot(s)

Pilots were typically held in cooperation with partner educators (though as a brand-new program this was not possible in the case of Robot Diaries). Though partner educators led the educational efforts behind the pilot, members of the design team held critical roles in each pilot, and in all cases directly interacted with students for extended periods of time. As the pilots were oriented towards discovering further design constraints and not to building a definitive case for an intervention's effectiveness, there was no intent to involve enough students to reach statistical power, or to look for control groups to compare the pilot results against. As such, pilots involved no more than twenty five students.

The Braille Tutor pilot occurred over six weeks at a school for the blind in India. A member of the design team traveled to India to help conduct the pilot. The designer worked with teachers to create software for the tutor and then trained the teachers in how to use the tutor. The teachers then used the tutor actively in their second and third grade classes, replacing some of the time allocated to Braille

7. Summary, Analysis, and Conclusions

writing practice with practice on the tutor. In all, twelve students participated in this pilot. At the same time, many other students at the school used the tutor informally.

There were two major pilots of the Robot Diaries project. We developed a summer participatory design workshop lasting six weeks and a fall workshop lasting nine weeks; in both cases sessions were two hours one day per week. Attendance at the summer workshop sessions ranged from one to seven, while eight girls participated in the fall sessions. The summer workshop was focused mostly on evaluating activities and materials for use in a future curriculum. The fall workshop built on lessons learned from the summer workshop to more quickly introduce materials. We also developed a software environment during the fall workshop to allow girls to program created robots.

The CSbots project had one major pilot, at a community college in Pittsburgh. The pilot involved four sections of our partner educator's CIT-111 (Introduction to Computer Science in Java) course. Students were introduced to the robot platform and software API for their third assignment, and completed assignments using the robot through the rest of the course. As the robots could not be taken home, students completed their assignments during lab times staffed by a member of the design team.

Requirements Evaluation

In each case, the evaluation of our pilots was focused on the effectiveness of the curricular approach and of the configurable embodied interface. As the primary goal of our analysis was to link the student outcomes to specific aspects of the design so as to produce recommendations, lessons learned, and additional design constraints for the next design cycle, an analysis of student learning and outcomes was required. In addition, analysis of student outcomes provided us with both important validation of the overall idea as well as pointing to specific areas of improvement. Given the small number of participants and high number of interacting variables, an analysis of student outcomes that attempts to link these outcomes to features of the design (either hardware or curricular) is open to intuition and subjective judgment. I accept that none of the evaluations provided us with the ability to make strong claims of causation but submit that this does not invalidate the notion of using formal evaluation techniques in small, experimental studies. To the contrary, I strongly believe that careful, planned evaluations of the pilots allowed us to discover constraints and determine efficacy of the programs in ways that ad-hoc observations of the pilots would not have.

In the Braille Tutor program, our evaluation instruments included a log of observations kept by the member of the design team while she was participating in the pilot, and surveys and assessments given to the students in the pilot. The sur-

veys and assessments sought to determine if the tutor improved learning of Braille Writing for a small cohort of students, as well as assessing their interest in using the tutor and their estimation of how difficult it was to use the tutor. These assessments suggested that the tutor's immediate feedback was providing a net benefit to the students. The observations focused mostly on how students and teachers interacted with the Braille Tutor: the difficulties they encountered starting and using the hardware and software, how frequently the tutor was used outside of regular class hours, and which individuals benefited most from tutor use and why.

Collected data for Robot Diaries included interviews with participants and their parents, and electronic activity logs. Participants were interviewed individually at the beginning and end of the workshop. Interviews included questions about relevant declarative knowledge (e.g., identify and provide a definition for relevant parts, such as sensors and motors) and designed systems (e.g., examine an electronic toy and describe its components/how it works). Participants were also asked to imagine how they might build a new system (an alarm) using a fixed set of components (a battery pack, alligator clips, switch, LED, servo, and sensor). Parents were interviewed in their homes at the beginning of the workshop and again after the workshop was completed. In the pre-interview, parents were asked about their child's previous experience with robotics and related technologies and about the family's activities related to science and technology. Post-interviews mainly focused on parents' impressions of the workshop and what their child gained from participation. Electronic activity logs were derived from girls' contributions to the Doodlechat and Roboticon Messenger online communities. From these logs we could see how frequently girls were logging in, contributing through chat, and contributing through doodling or posting Roboticons; as such, these logs provided us with a measure of interest in the software.

As more students were involved in the first evaluation of the CSbots program, the evaluation instruments were less geared towards discovering potential causalities at the student level. Instead, we relied extensively on surveys, comparisons to previous classes, and the observations of partner educators and members of the design team involved with the pilot. Survey data sought to characterize the typical interests and past experience of entering and passing students, as well as to track confidence and interest in assignments throughout the semester. These surveys allowed us to discover that assignments that were more interactive were highly-rated by students, leading directly to the decision to make the Finch richly interactive. Comparisons to past classes sought to examine the effectiveness of the pilot at the course level - comparing average grades, retention rates, and assignment completion rates to previous instantiations of traditional courses taught by the same educator. Finally, observations primarily focused on student points of frustration or confusion with respect to the software API.

7.1.4 Systems Alignment Cycles

System alignment cycles of design, pilot, and evaluation have a different primary goal from the constraint-finding process. These cycles are about using the constraints found by the initial evaluation and participatory design studies to create a fundamentally new design, and to improve the alignment of the various components of the design (curriculum, learning goals, assessments, and tool) through iteration. This difference in emphasis leads to the employment of qualitatively different methods and procedures.

Engineering and Curriculum Design

Systems alignment cycles tend to follow a more standard engineering design process than is the case with the constraint-finding process. Creating an incomplete or overly instrumented design and working with teachers and students in participatory design sessions is no longer necessary, as previous sessions have yielded design constraints based in the educational context. This does not mean that consultation with teachers or students is no longer important - as the design comprises hardware, software, and curriculum, there are likely to be many areas in which teachers or students can take either primary roles as designers, or offer feedback.

Even within the systems alignment cycles there are differences between the *first* design step and *subsequent* steps. The aim of the first engineering and curriculum design phase is to create a design that meets the constraints found during the initial evaluation and constraints-finding process. The aim of subsequent design steps is to use lessons learned from earlier systems evaluations to incrementally improve the design through the addressing of misalignments between design components, bugs in the hardware or software, or points of confusion or frustration in the curriculum.

In the Braille Tutor project, the first systems alignment design step resulted in major changes to the hardware and software of the tutor. The hardware was modified to make the tutor easier to set up and easier to use. The software was modified so as to create a more adaptive tutor (capable of reacting to individual student needs), and to make it easier to create additional programs for the Braille Tutor. Subsequent cycles have not resulted in changes to the hardware, but in additional software that expands the tutor's capabilities and thus its applicability to different ages and educational activities.

The first and only (to date) systems alignment cycle of the Robot Diaries project began with the identification of an overarching theme (the design process) which we felt would support our primary learning goals. From this theme and these goals, we concurrently designed a curriculum for an after-school multi-week

workshop, a kit of technical and craft materials, a controller (the Hummingbird) to control the LEDs, servos, and motors in the kit, and a software environment based on the environment created in the constraint-finding process to allow programming of the created robots. It was our intention to create a design that was sufficiently detailed and complete that it could be taught to educators who typically run after-school activities.

The major effort of the first systems alignment cycle of the CSbots project was the creation of a new robot, the Finch, with features aligned to the needs of the introductory Computer Science curriculum. Our principal design criteria for the new robot were that it should be highly interactive, robust, low-cost enough for individual student ownership, and have hardware that did not distract students from learning CS concepts. We updated our software API to reflect the different capabilities of the Finch, but kept much of the way in which students use the API similar to our prior design. Assignments for the Finch were designed in cooperation with our partner educators during the pilots.

Systems alignment cycles are marked by an effort to improve the alignment between the features of the configurable embodied interface, the software, and the learning goals, assessments, and instruction of the educational activity. Projects can vary greatly in the amount of flexibility they provide to the designers - in *Robot Diaries*, we were entirely free to choose and modify all of the elements of the overall design. With *Braille Tutor*, the learning goals were fixed, but other elements could be modified (including the assessments). In CSbots, we were able to modify only the hardware features, software API, and the assignments (a portion of the instruction of the course). In essence, differing amounts of flexibility yield differing design constraints - in CSbots, the learning goals and instruction of the course created design constraints for the hardware. In *Robot Diaries*, technological constraints in the hardware created constraints in the instruction and in some of the lower-level learning goals.

Systems alignment cycles may also strike out in new directions for the configurable embodied interface and may occur unguided by the original design team. Just as it is possible to create an aligned design without being able to modify learning goals or assessments (as in CSbots), so it is possible to create an aligned design without modifying the hardware feature set. This is precisely what is happening in the *Braille Tutor* project, with new work modifying the software to create practice games. We are planning a similar extension in CSbots, by creating support for programming languages that are appropriate for different age groups, so as to allow the Finch robot to be used by younger students. Finally, in many cases teachers and curriculum designers with no ability to change the hardware or software of the tool may still create their own systems alignment cycles by creating and revising curricula that are aligned to configurable embodied interface features.

Teacher-run Pilots

There are two major differences between the participatory pilots of the constraints-finding process and the teacher-run pilots of the systems alignment cycles: Firstly, teacher-run pilots were executed with significantly less involvement from the design team; in these pilots, designers served solely to offer technical assistance and when necessary, train teachers in executing the curriculum. Secondly, pilots were scaled up - they generally involved more students, occurred at multiple sites, or both. These two transitions, having educators lead the pilots and involving more students, are important steps to creating a design that can be disseminated to teachers and that fits into a broad educational context.

Pilots of the second Braille Tutor design continued at the site of our original partner, and also expanded to three locations in other developing countries. Although each pilot involved roughly the same number of students as our original pilot, the expansion to different sites provided strong evidence for the ability of the Braille Tutor to be integrated into the educational context of a school for the blind.

The redesigned Robot Diaries curriculum and controller were piloted with two groups - one was a group of home schoolers, and the other was a group at an after-school community center. Groups consisted of roughly ten girls, and so each was similar in size to the original pilot workshops. Teachers at both locations were trained by the design team on the curriculum, and executed this curriculum on their own. A member of the design team observed each workshop session to see if the curriculum was implemented as we expected.

The main CSbots pilot was done at the same location as our main 2007 pilot. We had students using the Finch robots for two semesters in both the introductory and intermediate Computer Science courses of our partner. In all, roughly twice as many students were involved with this pilot as in the first design cycle pilot. Designers were more hands-off with this pilot - we helped design some of the assignments with our partner, but our partner was fully responsible for grading students' Finch programs and explaining the Finch software and hardware to students.

Systems Evaluation

The evaluation philosophy behind the new pilots remained much the same: we were still focused primarily on evaluating the effectiveness of our design. However, given that the design was more explicit in terms of which learning goals should be met, student outcomes became a more important part of this overall evaluation. Other measures of effectiveness were the impressions of partner educators carrying out the program, as well as robustness of the tool and how closely the tool came to

the desired technical capabilities.

With the Braille Tutor, we used similar methods as in the participatory pilot - we ran small studies of student performance, collected observations of on-site members of the design team and teachers, and surveyed students. The strength of the second round of pilots was the general finding that the Braille Tutor was useful in additional locations.

The Robot Diaries second round of evaluations was heavily based on interviews of teachers, parents, and participants. We also observed all workshop sessions. Finally, we developed two evaluation activities - the debugging task (Hamner et al., 2010) and the creative design exercise. These activities were designed as an attempt to measure progress towards some of the higher level learning goals - goals that were difficult to measure using standard assessments.

The CSbots evaluation proceeded in much the same way as our initial pilot - we relied heavily on surveys, both pre/post and short post-assignment surveys. We also compared retention rates, assignment completion rates, and grades to prior years in the class.

It is important to note that in none of the three cases did we do the kind of evaluation that could provide strong evidence of improved learning; none of the studies involved a randomized control trial (RCT) with a control group. Instead, these evaluations were still oriented more towards determining which elements of the design to improve, and to determine in a preliminary way if our approach was succeeding. The evaluation methods do attempt to yield preliminary evidence of improved student outcomes, but I accept that none of the evaluation results could be used to make strong claims about the effectiveness of these programs on students. Instead, I suggest an evaluation trajectory that constantly involves more participants in a broader set of locations. I believe that a place for RCTs exists in evaluating the effectiveness of tools created with this process, but that such a study would not be particularly enlightening until a tool has been broadly disseminated and is in use at a number of locations by teachers who are totally independent of the research team.

7.1.5 Measuring Alignment

A systems evaluation should also attempt to determine whether the design was well-aligned. Though other measures (such as student outcomes) may hint at the alignment of the design, alignment can be measured explicitly both formatively and summatively. In all three cases, we were explicitly measuring how well-aligned the design elements were while engaged in the engineering and curriculum design cycle. The best example of how this was done is from the Robot Diaries project. Appendix A presents the full curriculum created for Robot Diaries, and this cur-

7. Summary, Analysis, and Conclusions

riculum shows how the activities were matched to learning goals, in part as a check to ensure that all learning goals were being addressed. Though not clear from the written documents, we also made sure that the activities could be performed with the tools we created. Measuring alignment in a summative fashion can be as simple as asking students questions related to alignment. In CSbots, we asked students the following open-ended question:

Please tell us how much you agree or disagree with this statement and why: “The assignments, lectures, in-class activities, and exams all focused on the same material and were all useful to learning the subject of the class.”

The responses demonstrated that some students did notice misalignments in the class, though fortunately these were minor in our case.

7.1.6 Dissemination

There are two types of dissemination for projects such as these: dissemination of ideas and dissemination of tool. The dissemination of ideas occurs through well-worn paths: publication of academic papers and presentations at conferences. This is relatively easy and occurs throughout the design process, but to be impactful of educational practice, it must be accompanied by dissemination efforts outside the academic realm. Dissemination of tool has the potential to impact educational practice, but is significantly more difficult. As such, the rest of this section is devoted to discussing dissemination of tool.

Unlike the other steps in this design process, there are few examples in our case studies to highlight effective dissemination strategies. The closest experience in the case studies was the dissemination of the alpha and Finch robot platforms in the CSbots study (see section 6.7.4). As such, the ideas presented here are grounded more in the future plans of these programs than in past experience and should be considered as future work.

When to Disseminate

A difficult problem for a design team is choosing when to end the iterative process - in other words, when is the design sufficiently well-aligned and ready for dissemination? There is always a tension between the potential for more improvement and the desire to provide the educator community access to a potentially useful new tool. As such, the decision is typically an exercise in ‘satisficing’ (Simon, 1957) - that is, choosing to disseminate when the design is good enough and acknowledging that perfection or potential further improvements do not justify the effort.

I offer a number of guidelines that have been used in our decisions about when to disseminate a tool:

- Does the tool's feature set support the curriculum's high-level learning goals, and not just tested assignments/activities?
- Do the evaluations suggest positive effects for students in the program? Are there any sub-groups that show especially large positive or negative effects?
- Does the projected commercial cost reach required cost constraints for acceptance?
- Are the curriculum, documentation, and interface sufficiently complete to allow educators to integrate and use the tool independently? Are there instances of educators borrowing the tool and using it with minimal hands-on guidance?
- Are involved educators requesting commercially available copies of the tool?

Of these, the first is likely to be the most difficult to evaluate. Unlike the other guidelines, which can be evaluated concretely, the first guideline draws from the design team's evaluation of the degree of alignment between all aspects of the design. As such, this guideline needs to be evaluated through discussion within the design team, with evidence drawn from thought experiments about potential other assignments or instructional techniques that are not in the current curriculum.

What to Disseminate

Throughout this section I have emphasized that the design consists of a hardware tool, software for the tool, and a curriculum with a set of learning goals that has been aligned with the features of the tool. A dissemination effort must include all three of these elements, such that:

- There is a commercial source for the hardware used in the design.
- The software is either freely or commercially available.
- Documentation for running the hardware and software is sufficient to allow a teacher or student in the target audience to be capable of setting up and running the embodied interface independently.
- If a new curriculum was created as part of the design, this curriculum is readily available, well-documented, and has explicit learning goals.

7. Summary, Analysis, and Conclusions

- If the interface is to be used in an existing curriculum, there are extensive examples showing how the tool can be integrated into the curriculum, where it is most useful, and case studies of how partner educators have integrated the interface in the past. Ideally, these case studies would demonstrate not just the assignments and activities modified by the tool, but also describe the specific learning goals of the class and the assessment strategies used, including those assessments which were not modified by the tool.

In addition to these, efforts should be made to foster a method of allowing educators using the tool to share experiences with one another and help one another. Online forums and/or shared wiki sites may be appropriate ways to achieve this.

The end goal of the dissemination package should be to provide sufficient information to allow *teachers to design with the tool*. Although there are broad similarities, every classroom and set of students is different. Teaching is about making design decisions to optimize student outcomes - not broadly, but in the context of a single class with a single set of students; in fact, teachers may change their strategy from one year to the next simply because the set of students changed. The dissemination package must allow teachers the flexibility to adapt and align the provided materials to their individual context, while being specific enough to provide examples of how to do such alignment.

How to Disseminate

Once such a dissemination package is available, there are a number of routes for spreading adoption of the design: one can ask the educators who have participated on the project to act as evangelists, spreading the program by contacting their networks of colleagues, one can arrange workshops to train educators in the design, or one could publish in journals typically read by educators or school administrators in a specific field.

At the point where a number of teachers have adopted the tool, it may be worthwhile to do a randomized controlled trial for the purposes of dissemination as well as to examine the effectiveness of the tool. Positive results from an RCT is in many cases necessary to allow the design to jump from being adopted by a few interested educators to being considered by school administrations and policymakers.

7.2 Similar Design Processes

I am aware of two external projects within the domain of Configurable Embodied Interfaces that have used an alignment-centered design process to create Configurable Embodied Interfaces. An important lesson from this related work is that

designers can do alignment-centered design *without making explicit use of the alignment framework or terminology presented in this chapter*. Rather, the original contribution of this thesis is making the model explicit so as to allow others to pre-emptively organize the planning of a design process.

The first of these projects is the Handy Board, briefly mentioned in chapter two, which was originally created for the MIT robot design course 6.270. The design of the Handy Board, the process of which is described in detail in Martin (1994), was iterated upon concurrently with iterations to the curriculum of 6.270, with important design features of the tool (like the use of an on-board microcontroller, or the development of an easy-to-use programming language) stemming directly from observations by the designer/teacher of student needs in the course.

The second project is the creation of the Fluke controller and Myro programming environment by the Institute for Personal Robots in Education (Blank et al., 2007). This project, much like CSbots, aims to develop an appropriate robot to use in Computer Science education. The Fluke is a small robot controller that interfaces with and adds capability to the Scribbler¹, a low cost robot platform made by Parallax, Inc. The capabilities of the Fluke, specifically a wireless tether and a camera, and the Myro environment and software API were created with the specific learning goals and context-derived constraints (like cost) of Introductory Computer Science education.

7.2.1 Learner-centered Design

In the field of human computer interaction, a similar approach to alignment-centered design has been espoused by Soloway et al. (1994). This approach, labeled learner-centered design by the authors, builds on top of user-centered design (Norman and Draper, 1986) and identifies three key requirements of educationally-focused software that are not relevant to software for experts or professionals: allowing growth in a student's understanding, reflecting the diversity of learners, and ensuring that learners remain motivated throughout the learning process. It also discusses how each of these three learner needs are addressed in the four components of a learning environment: The educational context, the tasks required of the students, the tools available to students to complete those tasks, and the interface between the student and the tool. The resulting design process, as exemplified by several case studies (Soloway et al., 1996), is not so different from the design process described in this work.

Thus, there exist both projects that made implicit use of alignment-centered design, and approaches that use similar techniques in other domains. That being

¹<http://www.parallax.com/tabid/455/Default.aspx>

said, there does appear to be a universal theme in all of this work: *Design trajectories succeed in creating educational relevant tools when they incorporate feedback from students and teachers frequently and at all stages of the design process.*

7.3 The Domain of Alignment-Centered Design

Until this point in the thesis, we have discussed alignment-centered design within a specific design domain, that of Configurable Embodied Interfaces. The observation at the conclusion of the last section is very general, and calls into question whether alignment-centered design as a process couldn't be applied more broadly, perhaps to software-only projects, or to the design of devices that are not educational but that still require contextual immersion by designers. The answer to this question is that there are no bright lines delineating the types of situations for which alignment-centered design is appropriate. Instead, I imagine that as the situation becomes further removed from educational contexts, and from tangible devices, the specific methods and case studies used in the thesis will become less applicable. For example, a designer applying alignment-centered design to the creation of a device for water purification in developing countries would not be able to use the method of instructional alignment, though participatory design and design-based research methods may still apply. Similarly, the creator of a software-based tutor for writing could follow alignment-centered design but modifications should be made to the process to reflect the design advantages of software-only systems (specifically the speed with which changes can be made to the design). Reflecting on the bounds of alignment-centered design, one realizes that as the specific methods and context are stripped away, the underlying structure begins to approach a more generic model of design.

7.4 Alignment-Centered Design and Engineering Design

In addition to the focus on feedback from students and teachers, similar work in configurable embodied interfaces shares a second commonality with alignment-centered design; all of the design processes derive from and add to a simple iterative model of engineering design pictured in Figure 7.2. It is worth describing the engineering design process in more detail, and explaining how alignment-centered design modifies this process.

The steps and description of the engineering design process are derived from two sources: personal conversations between the author and inventors of popular Configurable Embodied Interfaces, and a reflection on the author's own process in projects prior to the work described in this thesis. It should be noted that for

7.4. Alignment-Centered Design and Engineering Design

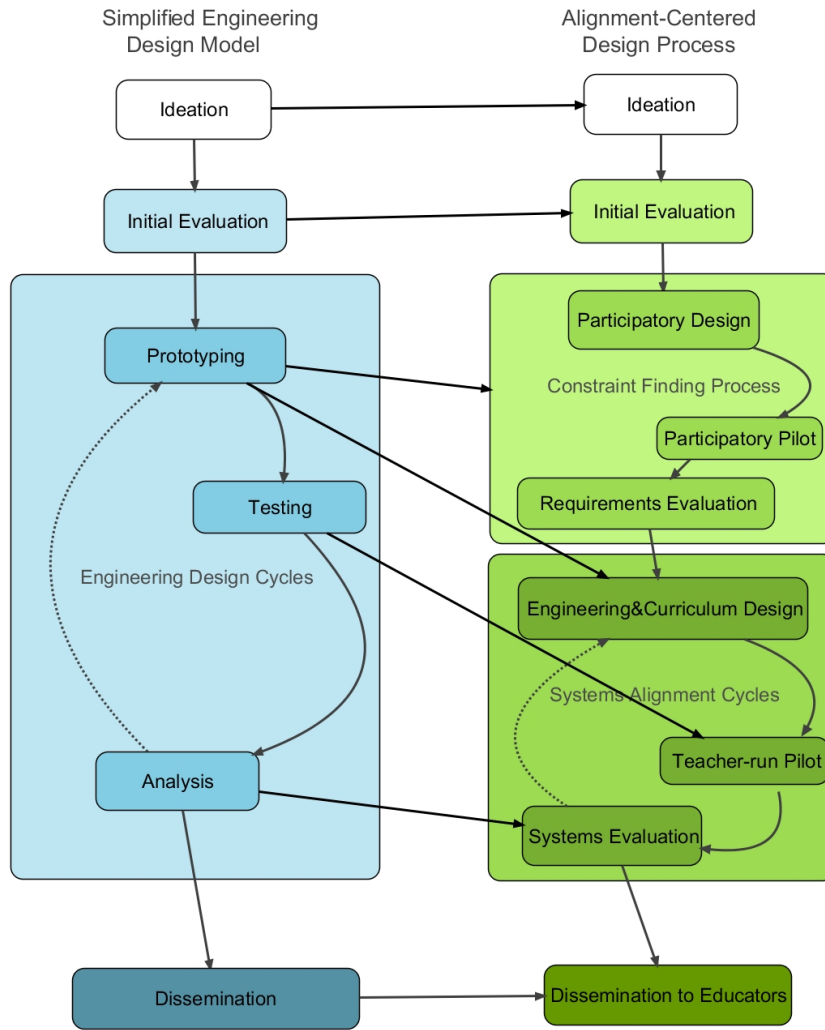


Figure 7.2: A simplified model of engineering design mapped to alignment-centered design

the purposes of this discussion, the diagram of the process and number of steps have been greatly simplified; many explanations of engineering design are available (and at times contradictory) (Simon, 1996; Schon, 1984); the model presented here represents a core process that is generally agreed upon as taking place.

7.4.1 Ideation

Universally, engineering design ideas spring in response to a perceived need or desire coupled with a unique knowledge or muse on the part of the ideator. Ideas are formed through the interactions of the ideator(s) with objects, people, or situations that cause perception of problems to solve. In alignment-centered design, it is likely that ideas spring from interactions between the ideator and teachers, educational environments, or students. It is fascinating that ideas may come years or decades after the sparking interaction; witness the case of Robot Diaries, the sparking interaction of which came from the ideator's experience as a seventh grade girl.

7.4.2 Initial Evaluation

Initial evaluations seek to retire some risks that the idea, if implemented, would fail. Typically in engineering design, there are two important risks to investigate: technological and resource-use. The first investigation seeks to answer the question of technical feasibility - essentially, is the idea possible with today's technology and does the design team have the necessary expertise to execute on the idea? The second risk is related to the use of resources that the problem's solution would impose on society or individuals - can the proposed solution be made so that benefits outweigh the costs of not solving the problem? This second question weighs heavily on user-acceptance, as users are unlikely to accept a solution with minimal perceived benefits.

The initial evaluations of the three case studies all devoted far more time to answering the second of these two questions. While technical feasibility was cursorily evaluated in all three cases, none presented a difficult engineering problem. However, in the case of Robot Diaries and CSbots, there was extensive risk to the potential solution from the educational context; simply put, we did not know if our proposed solutions were of interest to the students and teachers who would use the solution.

Provided that the initial evaluation suggests that the idea may succeed, the process next moves into an iterative cycle of prototyping, testing, and analyzing tests to determine if the solution meets all the desired constraints.

7.4.3 Prototyping

Hard and broad constraints are brought to the start of the prototyping phase by the problem that the idea is meant to address, and by the initial evaluation; it is these constraints that delimit the design space. The prototyping process is an exploration of this space through experimentation and reflection. Schon (1984) describes this

process as a conversation with available materials, and it is through this conversation that additional constraints and solutions are found, and a testable design eventually created. It is important to note that there is testing in the prototyping process, but it is qualitatively different from the type found later in the engineering design cycle. Early tests may be made of incomplete portions of the solution, as experiments to inform the prototyping process and make additional changes or additions to the design.

Prototyping shows up in two ways in the alignment-centered design process. The constraints-finding process is an extended prototyping phase seeking to discover educationally relevant constraints. This is why the process is participatory in nature, and why the design in this early stage is highly fluid; it allows the design team to conduct mini-experiments, both technical and social, to determine the low-level constraints and capabilities required in the design to meet the high-level design goals.

Prototyping also shows up in a more traditional engineering fashion during the engineering and curriculum design portion of the systems alignment cycles. There we are taking the constraints found in the previous cycle and using these to create prototype designs consisting of hardware, software, and curricula.

7.4.4 Testing

The conclusion of the prototyping process is marked by the creation of a design that is testable with respect to the goals set forth by the initial idea. The test itself is a reflection of the attributes of the design that cause the design to meet these goals. In the alignment-centered design process, our principal tests were carried in the format of teacher-run pilots. It is within such pilots that we were able to test several important attributes of our designs:

- The ability of the design to cause movement towards specified learner goals.
- The usability of the design by students.
- The effectiveness of the training portion of the design on teachers.

Our tests were fairly skewed towards the relevance and effectiveness of the design within the educational context; other projects could easily skew in different directions. For example, projects that focus heavily on creating new capabilities might focus on robustness and cost; an example of this might be the development of an electric car. Other projects (for example, the iPod) may focus heavily on usability.

7.4.5 Analysis

The analysis methods required simply reflect the tested attributes. Typically, the analysis has two broad goals: to determine how the prototype design performed on the test, and to determine whether this performance is sufficient to consider the design for dissemination, and, if not, to describe ways in which the design should be improved during the next prototyping phase. The systems evaluation within the alignment-centered design process maps to the analysis phase. In the case studies, we used a variety of evaluation methods to perform this analysis. Reflecting the nature of our testing process, these methods were focused on student outcomes as well as observations to determine if the tool could be used by trained teachers.

7.4.6 Dissemination

The engineering design process eventually leads to a disseminable design; this is the end-goal, as a solution to a problem that is beneficial can only be so if it is used by society. In the case studies, dissemination will likely occur through networks of teachers developed over the course of the research, and through continued efforts on the part of the design teams to publicize the work by organizing training workshops, visiting teacher conferences, and seeking funding for outreach activities. More generally, commercial dissemination pathways often involve marketing and advertising, while other projects may be able to immediately disseminate by government fiat (as is the case with military engineering research).

7.5 Challenges and Limits of Alignment-Centered Design

In my experiences with the three cases described in this thesis, I have found two significant challenges to and two major limitations of the alignment-centered design process as presently formulated. The first challenge relates to the evaluation of the tool in educational settings. The second challenge is a matter of the human resources required to navigate the process. The limitations relate to the incorporation of student interests and background, and to the constraints that the educational context places upon the design. I offer advice for meeting the challenges and dealing with the limitation of alignment-centered design below.

7.5.1 Evaluation

A major challenge of alignment-centered design is the proper and careful evaluation of the educational pilots, both during constraint-finding and systems alignment. Evaluation is typically made more difficult by a number of factors, including:

7.5. Challenges and Limits of Alignment-Centered Design

- A small number of participants are typically desired as untried interventions require a heavier teaching presence to catch problems as they appear.
- If the intervention targets learning goals that aren't typically addressed, a control group can not be established that spends an equal amount of time on a 'similar' educational activity.
- Some learning goals are much more difficult to measure than others. If the new intervention targets difficult to measure goals, this adds further complexity to the evaluation problem.

Essentially, these characteristics prevent the use of a number of standard evaluation techniques. The small number of participants and lack of a control ensure that there will not be enough statistical power to show significance. The learning goals themselves may be difficult to measure using traditional methods like exams. We are left, then, with a number of methods that have worked in our cases. Robot Diaries is by far the best example of the three cases in maximizing information sources from a small number of participants. Participants were interviewed, surveyed, their parents were interviewed, researchers either led or observed all workshop sessions, logs were kept of online activity, workbooks were scanned and analyzed, and novel evaluation techniques were created to assess post-workshop capability. The Robot Diaries case is an excellent example of a workable evaluation strategy when faced with the factors described above and when creating a new program; Bernstein (2010) is a doctoral thesis devoted entirely to the Robot Diaries evaluation methods and results.

CSbots provides an interesting case study in how to compare a course modified by a new tool but where the learning goals and assessments are essentially the same as for previous years. In this case, we had data from our partner about his students' retention rates and grades going back six years. Based on this data, we were able to compare our pilot years with non-pilot years and were able to build a case that our intervention was improving retention and doing no harm to learning. This was a valuable result, but it is important to note that it is not considered strong evidence - comparisons to previous years have problems with validity (it is possible that external factors, like a recession, changed the makeup of incoming students from year to year). Even so, such a comparison was relatively easy to do because the data was readily available. Another lesson of the CSbots case study is the difficulty of using survey data; though our surveys provided us with valuable data, they were required to be voluntary, and as such, we frequently had very low completion rates. If our completion rates had been near 100%, the survey data would have provided much stronger evidence.

7. Summary, Analysis, and Conclusions

Developing an evaluation strategy is itself a design process. The case studies of this thesis described a large number of possible methods to use: likely not all are required for any given project, instead, methods should be chosen based on their ability to provide the team with useful qualitative or quantitative data.

7.5.2 Human Resources

Closely linked to the evaluation challenge of alignment-centered design processes are the human resources required to navigate through the process. Each of the three case studies was made up of a core design team, and for good reason: Alignment-centered design processes in the space of configurable embodied interfaces require a number of different skills that are rarely found in a single individual: software interface design, electrical and mechanical engineering, teaching, curriculum design, and educational evaluation are all necessary. I believe that it is this reason, more than any other, that would prevent designers from pursuing an alignment-centered process. That being said, I believe that it is possible in a number of settings to assemble the required team of people: a partner teacher, someone who is knowledgeable of educational evaluation, and engineers who can do both hardware and software design. Of these, I assume that the designer is capable of or can easily recruit people who can do the requisite engineering design; after all, such people are required for any engineering design process. In my personal experience, partner teachers who are enthusiastic about projects or ideas are fairly easy to find; it is a tribute to the teaching profession that many teachers are generally selfless and constantly searching for ways to improve the experience for their students. Thus, finding a teacher to work with is simply an exercise in contacting a sufficient number of teachers in the relevant subject area. I refer the reader to the recruitment of high school teachers in CSbots described in section 6.7.4 as an example of a way to do this. I suspect that finding a person capable of performing educational evaluation is regarded by many as the true human resource hurdle. Before advising how this challenge can be met, I share some personal circumstances to illustrate my advice.

Two personal circumstances greatly improved my ability to think about and carry out alignment-centered designs. The first of these is my involvement in the Program for Interdisciplinary Education Research (PIER)², a pre-doctoral program designed to provide students in a wide number of home departments with training in the learning sciences. This training included courses in curriculum design (the ideas from which led to the central theme of this thesis), research methods, and a literature review of education research and policy issues. It also provided me

²<http://www.cmu.edu/pier/>

access to a community of people personally knowledgeable with carrying out educational evaluations in real-world settings. The second of these circumstances was a close collaboration with a Ph.D student at the Learning Research and Development Center. This student directed the evaluation of the Robot Diaries program, and through my experience helping her to carry out the evaluation, I learned a fair amount about evaluation as well.

Based on my personal experience, my advice to any designers interested in pursuing educational evaluations of their projects is to:

- Identify a physically nearby academic community that is interested educational evaluation. As designers are searching for someone to aid in evaluation, so many evaluators are searching for designs to evaluate.
- Develop a close collaboration with someone familiar with the research methods required for educational evaluation.
- If possible, take a course in research methods to develop familiarity with the topic.
- Work closely with the evaluator to help develop the evaluation instruments. This provides valuable experience to the designer, and ensures that context- or design-specific elements are included in the evaluation. If the evaluator is not aided by the rest of the design team, it is possible that the evaluation instruments will miss important details of the design.

7.5.3 Student Interests and Background

In each of the three case studies, the interests and capabilities of students are primarily discovered during the constraint-finding process. I believe that it was an omission of our specific designs to wait until this step to engage with students. If the initial evaluation is about retiring the risk of a design that is unacceptable to end-users, students should be part of that evaluation. To some extent this was done in the Robot Diaries program through the use of a focus group, and in this case the lessons learned from the focus group caused significant changes to the program design trajectory. As the other two programs sought to create tools for existing curricula, the initial evaluations of these studies focused on gathering information from teachers. If I were to re-run these studies, I would have surveyed students in all three case studies before creating any hardware. By doing so, we may have discovered certain constraints in time to affect the designs used in constraint-finding. For example, we may have learned of the students' fear of electrical wires in the Braille Tutor case, or discovered what kinds of assignments excite students in computer science education. Fortunately, in our case studies the initial tools designed

7. Summary, Analysis, and Conclusions

for the constraint-finding process were workable and acceptable to students. That being said, I believe it is possible that an initial evaluation that solely focuses on teachers could lead to a design that can not be tested in a constraint-finding process because it is unacceptable to students.

7.5.4 Constraints

The alignment-centered design process is strongly focused on creating tools that are well aligned to specific educational environments. This can lead to tools that are highly specific. Such tools may have deep impact on a small community, but are unlikely to be usable in other contexts. Of the three case studies, the Braille Tutor is the best example of such a tool; the specific constraints that created successes in the educational environment and that ensured that learning would transfer to a regular slate and stylus also ensured that the tool would only ever be applicable in schools for the blind in developing countries. The CSbots program has made similar design choices that limit applicability; many robots are used in both engineering and Computer Science programs (for example, LEGO Mindstorms and NXT kits are used in both). By focusing on Computer Science education, we made design choices that limit the applicability of the Finch to an engineering classroom; the robot is self-contained and has neither mechanical mount points nor accessible electrical signals.

Thus, different design processes may be called for if the idea for a project stems more from technological innovation than from a clear application of existing technologies to an educational context. In the space of configurable embodied interfaces, new sensing, processing, and actuation technologies are constantly appearing that may allow for the creation of a configurable embodied interface that has fundamentally new capabilities. The designer may recognize broad educational applications for the interface, but may not have a specific curriculum or set of learning goals in mind. In such a case, the alignment-centered design process may lead to an overly specific design. I present an illustrative example of a project that was characterized by creating a fundamentally new capability.

7.5.5 Conventional Engineering Design: The CMUCam

The CMUCam (Rowe et al., 2002) is a configurable embodied interface consisting of a camera, microcontroller, and I/O pins to communicate with other microcontrollers and with computers. The microcontroller runs a program that uses computer vision algorithms to determine things such as the center of a blob of the same color, or the edge of white or black line. The microcontroller allows input from other microcontrollers to set what color blob or type of line to track. Designed in

2001, the CMUCam was revolutionary; it allowed hobbyists and students to create robots that could track brightly colored balls using very simple robot controller boards. Prior to the CMUCam, a robot had to have an on-board full-scale computer to display similar behavior. The CMUCam has found its way into a number of robotics, computer science, and engineering courses at the university level; despite this, the design process followed by the inventors was focused much more on engineering than on education. The following is an account of this process, described to me by one of the inventors of the camera.

The idea for the CMUCam came from the recognition of a new low-cost 8-bit microcontroller, the Scenix SX28. It was posited by one of the inventors that the chip might be fast enough to be able to simple vision processing. The utility of such a sensor was instantly recognized by the inventors, all of whom had personal experience as robot hobbyists. The CMUCam was then prototyped - a camera module was located, several revisions of the CMUCam circuit board were created, and the firmware was written. This happened over the course of several months, and very much followed the style of Schon's reflective process. The communication protocol between the CMUCam and other microcontrollers was written into the firmware. Although the protocol was the way for outside users to access the CMUCam's functionality, the protocol was not tested by outside users. The inventors did test the protocol with other microcontrollers to ensure there were no errors. Finally, the CMUCam had several additional features - a status LED, and two ports that allowed servo control. The servo control ports were added specifically because there was just enough space in program memory left to support them, and it would allow the CMUCam module to be turned into a ball tracking robot as a demonstration. The CMUCam was released in late 2001, with a manual documenting the hardware and communication protocol. Although it was somewhat difficult to use, the usefulness of the sensor generated a strong technically-savvy initial community, who then provided support to one another. The CMUCam hardware iterated on twice in the ensuing years, with releases of the CMUCam2 and CMUCam3 generally trending towards both expanded functionality (due to improved underlying technologies) and improved programmability/user-friendliness.

It would have been possible to use an alignment-centered design process on the initial idea for the CMUCam; for example, the idea could have led to a product well-suited to either a computer vision or embedded systems design course. However, such a process may well have led to the inclusion or exclusion of features that would have prevented broad applicability.

7.6 Conclusions: A Choice of Process

Ultimately, the choice of a design process to use is up to the design team. In many cases, this choice may be unconscious - it is quite possible for designers to simply navigate an engineering design process in the same way that they are used to. A major contribution of this thesis has been the formulation and description of the alignment-centered design process. It is my fervent hope that the case studies and analysis described herein allow designers to consider alignment-centered design, in whole or in part, if the circumstances of their idea merit it.

A convergence of trends has decreased the engineering effort of creating a new configurable embodied interface over the last twenty years:

- The number of available sensors and actuators has grown considerably; furthermore, interfacing these sensors and actuators to microcontrollers has become easier.
- Software and turn-key prototyping facilities have both simplified the process of manufacturing complex circuit boards, mechanisms, and housings, and has decreased the costs associated with prototyping.
- The component costs of sensors, microcontrollers, and actuators has fallen significantly while the capabilities of these technologies have increased, allowing more feature-rich tools to be created at lower prices.

The end result of these trends is that it is now possible to easily customize configurable embodied interfaces to support the curricula of a wide variety of courses and educational activities. Configurable Embodied Interfaces of the future may be more diverse and more specialized than the general purpose tools built to date. Tailored configurable embodied interfaces can be created to support any number of topics, from geometry to world history, hand writing skills to microbiology, and everything in between. These new tools will succeed in becoming educationally relevant if they are designed not just for, but aligned with the educational context.

Appendix A

Robot Diaries Curriculum

This section contains the curriculum created for the second iteration of the Robot Diaries program. The curriculum was created by Robot Diaries team members Emily Hamner, Debra Bernstein, Kristen Stubbs, and the author. The curriculum consists of five parts: an introduction to teachers explaining the philosophy behind the program, three parts prescribing a chronologically ordered curriculum, and an appendix containing additional activity ideas, materials and handouts, and some documentation. Though assessment was not a major part of this curriculum, the explicitness of the curriculum and learning goals should allow the creation of well-aligned assessments. Each section starts with a table linking knowledge and skills learning goals to specific instructional activities. Dispositional goals were not included in these tables as they were seen as program-general (it was inappropriate to link specific dispositional goals to activities). The specific dispositional goals are specified at the end of this appendix.

Arts & Bots

An Arts and Technology Workshop for Girls



Community Robotics, Education and
Technology Empowerment Lab
(CREATE Lab)
The Robotics Institute
Carnegie Mellon University

University of Pittsburgh Center for Learning
in Out-of-School Environments
(UPCLOSE)
Learning Research and Development Center
University of Pittsburgh

Table of Contents

Overview – *Introduction for instructors including learning goals and materials.*

Part 1: Introduction - *Introduction to robots as expressive devices, hardware components, and programming software.*

- Robots and Expression
- Motors, LEDs, and Basic Circuits
- Servos, Microcontrollers, and Program Building Blocks
- Sequential Programming and Communicating with Programs

Part 2: Building Basics – *Design, build, and program a simple robot.*

- The Design Process
- Build Your Own Robot
- Write and Share Expressive Robot Programs

Part 3: Advancing Your Design – *Iterative cycles of design, building, and structured use activities.*

- Expand Your Design
- Expand Your Robot
- Write and Share Longer Programs

Appendix – *Expansion activities and supporting materials.*

- Group Activities
- Individual Activities
- Pictures of Arts & Bots Materials

Introduction to Teachers

In the Arts & Bots workshop girls create a personal, customized robot designed to serve as a means of expression. Using light, sound, and movement, students can choreograph their robot to express emotions. They can then share these expressions with their friends in the Arts & Bots community. Ultimately, the robot provides a unique means of exploring, expressing, and sharing emotions, ideas and thoughts while promoting technological literacy and informal learning.

The Arts & Bots curriculum is driven by strong social narrative along a thread that has extant value and meaning to diverse students. Technology is not the prime motivator but rather an enabler for emotional and social communication. This sets Arts & Bots apart from other technology programs, such as the US First, BEST, and Botball competitions. These programs may engage certain types of students, but they also may limit participant diversity because they are short-term, high-intensity, competition-driven and technology-focused.

By appealing to students' experiences with storytelling, arts and crafts, and personal communication, Arts & Bots aims to encourage technology literacy in a way which appeals to a broad range of students; we hope this may eventually lead to greater diversity within fields such as computer science and engineering.

Learning Goals

The Arts & Bots curriculum draws upon students' interests in arts, crafts, communication, and storytelling while providing them with the technical knowledge and knowledge of the design process which enables them to build and utilize their own personal robots. At the beginning of each of the three main parts of the curriculum, you will find a chart detailing more specific learning goals for that section. In general, the curriculum focuses on:

- Technical knowledge (i.e., how a circuit functions, how different motors move)
- Application of technical skills (i.e., attaching technical components to a controller board)
- Design process knowledge (i.e., awareness of the steps of the design process)
- Application of the design process (i.e., use of the design process to create and modify technological designs)

The overall structure of the curriculum is built around the design process, which consists of:

- Planning
- Prototyping
- Testing, evaluating, and critiquing designs
- Troubleshooting
- Documenting designs

After creating an initial robot, each student will be asked to evaluate and refine her robot several times over the length of the course. This helps students to feel a stronger sense of

ownership of their robots and encourages the development of problem-solving skills. In addition, the Arts & Bots curriculum is designed to help students develop confidence in themselves and their ability to work with technology, such that by the end of the program, the student:

- Is aware that there are many ways to use technology, including in situations which do not appear to be 'technological'.
- Has interest in further workshops/activities.
- Has interest in continuing to use the robot after the program has ended.
- Voluntarily uses chat software, creates complex programs, or designs custom robot parts.
- Persists at tasks despite mistakes or difficulties.
- Is willing to use new technologies.
- Is willing to try to solve a problem before asking for help.
- Is willing to try a variety of solutions to a problem.
- Asks for help when it is needed.

Materials List

Technical Supplies

- Alligator clips
- Battery packs
- Batteries
- Switches
- Computers with internet access (1 / 2-3 students)
- Hummingbird microcontroller, power supply, mini USB cable (1 / student)
- Servos, servo extenders (at least 4 / student)
- Full color LEDs (2 / student)
- LEDs (at least 4 / student)
- Vibration motors (at least 2 / student)
- Motors (at least 1 / student)
- Wheels for motors (1 / student)
- Wire
- Speaker and speaker cable (1 / student)
- IR sensors
- Light sensors, LED flashlights, flashlights

Documentation Supplies

- Design notebooks (3-ring binders) (1 / student)
- Pencils, Pens
- Paper
- Digital camera or Polaroid camera
- Poster boards (documentation option 2)
- Scanner (optional; documentation option 1)

Craft Supplies

- Foam board, cardboard
- Scissors, glue
- Hot glue guns, hot glue sticks
- Foam
- Bells, chimes, sandpaper, pie pans
- Corn, rice, small boxes or containers with caps
- Styrofoam balls (1" to 1.5" diameter)
- Feathers, pompoms, pipe cleaners, ribbon
- Popsicle sticks, dowel rods, wooden skewers
- Jewels, googley eyes, felt
- Cardboard cake circles
- Construction paper
- Markers
- Kraft paper
- Rubber bands
- Zip ties, electrical tape
- Animal print fabrics
- Pushpins
- Pegboard

Tools

- #4 flathead screwdriver (1 / student)
- #1 Phillips screwdriver
- Wire cutters
- Wire strippers
- Exacto knives, replacement blades (1 / student)

Miscellaneous

- Fairy tales

Additional Information on Materials

The craft supplies and materials you provide to the students can have a big impact on student creativity and expressiveness. Below is more information on some of the supplies we chose for our workshop and the reasoning behind these choices. You can also find pictures of the technical supplies in the appendix.

Technical Supplies

- LEDs – Provide students with a variety of LED colors (red, orange, yellow, green, blue) to aid in emotional expression.
- Wheels for Motors – Although the goal is not to make mobile robots, the wheels attach easily to the motor shaft and make it easier to attach other things to the motors.
- Servos, Motors – Servos and motors are both good examples of components that roboticists use regularly.

- Vibration motors – Students enjoy using the vibration motors and they can relate them to something in their everyday lives (vibrating cell phones).

Craft supplies

Bring craft supplies to each session. They can be integrated in to many of the activities, even those that may not seem immediately crafty such as the sound making activities.

- Foam board – Foam board comes in white, colored (color one side, white other side), and black. Avoid harder types of board such as Gator board because it is too difficult to cut. Cardboard can be used as a less expensive alternative and as a cutting surface.
- Hot glue sticks – In our experience, students loved using the hot glue guns. Get an abundant supply of hot glue sticks.
- Styrofoam balls (1” to 1.5” diameter) – These are great for diffusing light from LEDs. LEDs can be pushed directly in to the Styrofoam.
- Bells, chimes, sandpaper, pie pans, corn, rice, small boxes or containers with caps – These can be used for the noise making activities.
- Jewels – Jewels can add a fun decorative accent and add to the inviting, girl-friendly feel of the workshop.
- Cardboard cake circles – Cake circles are a quick and easy way to get circular cutouts.
- Kraft paper – Robot building can get messy. Use kraft paper to cover work surfaces.
- Zip ties or cable ties – As well as organizing cables, zip ties can be useful for attaching things together. However, students are often not familiar with them and don’t make use of them without a demonstration.
- Pushpins – Pushpins are useful for making or starting holes.
- Pegboard – Pegboard can be used as a base surface. For example during the make a noise activity, servos can be attached to the pegboard with zip ties or wire to help hold them in place.

Documenting Designs

Throughout the curriculum, we have provided activities which encourage the girls to document their designs. These documentation activities are an important part of the curriculum, as giving girls the chance to share their experiences in their own words and show off their creations fosters a sense of pride and accomplishment. There are two documentation options:

Option 1: Girls create personal web pages using a blog format. This option requires digital pictures and possibly video clips.

Option 2: Girls create posters. They may make formal posters or use collage techniques to explain the development of their robots. This option requires Polaroid pictures or printouts of digital pictures, and poster boards.

In our experience girls loved creating their own web pages. They also really enjoyed personalizing their pages through color and pictures. Using web pages to communicate their experiences is in line with the Arts & Bots theme of using technology as a means of creative self-expression and communication. If you choose not to use web pages in your curriculum, you can still help girls reflect on their experiences and share their accomplishments by displaying the girls' posters for their families and peers.

Software Installation

As part of the workshop setup, you will need to install software on all the computers that the students will be using. All the necessary software is included on the CD accompanying this curriculum. Detailed software installation instructions are included in the appendix.

Part 1: Introduction - *Introduction to robots as expressive devices, hardware components, and programming software.*

- Robots and Expression
 - Motors, LEDs, and Basic Circuits
 - Servos, Microcontrollers, and Program Building Blocks
 - Sequential Programming and Communicating with Programs
-

Timeline

Activity	Estimated Time	Elapsed time
Introduction to Arts & Bots	10 minutes	0:10
Survey	15 minutes	0:25
Group Activity: Expressive Charades	20 minutes	0:45
Introduction to Motors, LEDs, and Circuits	10 minutes	0:55
Group Activity: Hypnotic Eyes	10 minutes	1:05
Introduction to Hummingbird and Software	25 minutes	1:30
Group Activity: Make a noise	15 minutes	1:45
Home Activity 1: Scavenger Hunt	5 minutes	1:50
Home Activity Follow-Up: Scavenger Hunt	10 minutes	0:10
Introduction to Express-O-Matic	20 minutes	0:30
Group Activity: Foley Artist	40 minutes	1:10
Choose a User Name	15 minutes	1:30

Materials:

Design notebooks, pencils, markers (1 / student)

Alligator clips, battery packs, batteries, switches

Computers with internet access (1 / 2-3 students)

Hummingbird, power supply, USB cable (1 / student)

Servos

LEDs

Vibration motors

Motors

Craft supplies, Pipe cleaners

Tools

Digital camera

LED flashlights

Sound materials (bells, chimes, paper, sandpaper, other frictional noises, pie pans, etc.)

Learning Goals

		Part 1: Introduction - Activities						
		Introduction to Motors, LEDs, and Circuits	Group Activity: Hypnotic Eyes	Introduction to Hummingbird & Software	Group Activity: Make a Noise	Individual Activity 1: Scavenger Hunt	Introduction to Express-O-Matic	Group Activity: Foley Artist
Learning Goals	Technical knowledge							
	Describe how a simple circuit functions.	X	X					
	Describe how different motors move (DC, vibration, servo).	X		X	X			X
	Describe how a vibration motor works.			X	X			X
	Technical knowledge application							
	Find the technical components presented in the workshop which are a part of an everyday object (i.e. robotic toy, Furby, appliance).					X		
	Demonstrate the use of hand tools such as glue guns, screwdrivers, and x-acto knives.				X			X
	Construct a simple circuit.	X	X		X			X
	Attach servos, vibration motors, and LEDs to a controller.			X	X			X
	Create short programs using a graphical programming language.						X	X
	Design process application							
	Planning							
	Formulate design goals.							X
	Identify design constraints.				X			X
	Identify available resources.				X			X
	Identify the limitations of an available resource.				X			X
	Prototyping							
	Assemble the resources needed in order to implement a design.				X			X

Introduction to Arts & Bots

(10 minutes)

An introduction to Arts & Bots that covers:

- Tell girls what kinds of activities they will be engaged in during the workshop (designing, building, personalizing, and programming robots; creating their own web page to document their experiences; playing games with their robots and putting on a play). Even if students have not worked with robots before, if they can do arts and crafts, they can do this workshop.
- Open with the idea that robots can be used to express emotions. Discuss the robots and robotic art pieces pictures on the handout. Talk about how they express emotions and moods, communicate, make music, and use movement for expression. Also point out that robots can be beautiful and creative. You can also show one or both of the videos of Keepon included on the CD (one is very short, just 14 seconds). See the Expressive Robots and Robotic Art instructor information and student handout at the end of this section.
- Hand out binders so students have somewhere to store handouts, design ideas, etc.
- Inform girls that Arts & Bots is part of a research study. The researchers want to see what kind of technology activities girls like. They might ask you to fill out a survey or answer some interview questions. They will also be here to observe the activities.



Video: Keepon Expressing and Keepon Dancing



Handout: Expressive Robots and Robotic Art

Survey

(15 minutes)

Allow time to complete the introductory survey if not already completed. Survey will be provided by the researchers.

Group Activity: Expressive Charades

(20 minutes)

During this ice-breaker girls will play charades with some modified rules.

1. Girls will pick from a list of the following emotions:
Angry, Anxious, Ashamed, Bored, Cautious, Confident, Confused, Depressed, Disgusted, Ecstatic, Embarrassed, Exhausted, Frightened, Frustrated, Guilty, Happy, Hopeful, Hysterical, Insecure, Jealous, Lonely, Love-struck, Mischievous, Overwhelmed, Sad, Shocked, Shy, Smug, Surprised, Suspicious.
2. The girls presenting is not allowed to vocalize in any way (speak or make sounds).
3. Girls can access a number of props to make sounds and use a variety of colored LED flashlights.

Each girl will have one minute to allow their team to guess the emotion being expressed. After 15 minutes, we suggest the teachers guide a short discussion on the kinds of motions and expressions that were successful in communicating an emotion. For large groups, we suggest that this game is played in groups of 6, with the teacher moderating each game.

Introduction to Motors, LEDs, and circuits (10 minutes)

Show students what an LED, a motor, a switch, an alligator clip, and a battery pack are and do. Explain how a circuit needs to be a closed loop in order to work. Explain how energy flows from high to low potential, and then explain polarity to show how LEDs only turn on in one direction while motors just change direction based on polarity.

Each group of girls should quickly put together a circuit that turns on either a motor or an LED (green, blue or white LEDs only) with a switch.

Two AA batteries work with green, blue, and white LEDs. **Red and yellow LEDs can't be used for this activity because the 3-volt, AA battery packs will burn out the LEDs.**



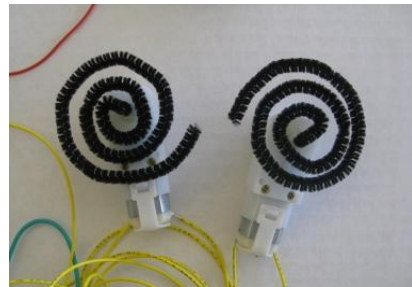
Handout: Circuit Basics

Group Activity: Hypnotic Eyes

(10 minutes)

Materials per student: 2 pipe cleaners, 1-2 motors, 1-2 switches, 3-6 alligator clips, 1 battery pack for 2 AA batteries (3.2V), 1 battery pack for 4 AA batteries (6.4V). (If there are not enough materials for each student to have 2 motors and circuit sets, students can work in pairs, each making 1 circuit.)

Students should use the pipe cleaners to create hypnotic spiral eyes and attach these to the motors. They should then experiment with (a) making the eyes spin in the opposite direction by flipping the positive and negative sides of the battery, (b) making the eyes spin faster or slower by changing between the 3.2V battery pack and the 6.4V pack.



Introduction to Hummingbird and Software (25 minutes)

Begin this introduction by explaining what the purpose of the Hummingbird is (to act as a way for girls to control and program their robots without constantly flipping switches/pushing buttons). In groups, lead girls through hooking up their motors from the previous activity (keep the eyes attached), an LED, a vibration motor, and a servo. Since the girls haven't yet seen servos and vibration motors, their function should be briefly demonstrated. See the How to Connect to the Hummingbird handout for connection details.



Handout: How to Connect to the Hummingbird

Once the girls have hooked up the motors, LED, servo, and vibration motor, lead them through launching the software and show them how each component can be controlled. They can then control the hypnotic eyes using the software instead of by making changes to the physical circuit as in the previous activity. Note how it is much easier to change the direction or speed or rotation and that the software gives finer control over speed.



Handout: Software Instructions (located in the Appendix)

Group Activity: Make a noise

(15 minutes)

Task girls with building something using motors or servos and noise-making materials (bells, chimes, paper, sandpaper, other frictional noises, pie pans, paper cups, and other craft materials) to create a simple noise. Their creations should be controlled using the software.

Home Activity 1: Scavenger Hunt

(5 minutes)

Girls go home with the task to find objects in their home that have one or more of the following: buttons & switches, motors, LEDs, something programmable, and something robotic. ‘Computer’ is not an acceptable answer for any category. There’s a Bonus for finding things that can fit in two or more of the categories!



Handout: Individual Activity 1: Scavenger Hunt

Home Activity Follow-Up

(10 minutes)

Review the scavenger hunt findings. See if any students found an item that non one else found. Compare answers and see who found the most unique items.

Introduction to Express-O-Matic

(20 minutes)

Launch the software and show the Express-O-Matic tab. Explain how it allows you to script your creations to move and act automatically. Explain the process of using Express-O-Matic, specifically:

- Using the Hummingbird tab to create and save expressions.
- Chaining together expressions.
- Creating a loop to continuously repeat expressions.
- Changing the delay between expressions.

We suggest you have a robot ready to demonstrate with, and quickly create a sequence of expressions by saving two or three expressions and chaining them to make a sequence.



Handout: Refer to the Software Instructions (located in the Appendix).

Group Activity: Foley Artist

(40 minutes)

Several Harry Potter clips are included on the CD accompanying the curriculum. Copy these clips the students' computers as part of the setup for this activity. Make sure VLC Media Player is installed on the students computers prior to this activity. (See the appendix for VLC Media Player installation instructions.)

Assign each group of 3-4 girls a ten second clip from a Harry Potter film. (Longer clips can be divided so that each group has a few good sounds in their section of the clip. For large workshops, multiple groups may work on the same clip.) Tell students about how artists create sound effects in movies, and how sound effects are very important for setting the mood and tone of a scene. Then introduce them to the activity – they'll need to make robots that create sound effects for short film clips. The sound effects they decide on don't need to sound the same as the ones in the clip. It's up to their artistic imagination what kinds of sounds fit. The last five minutes of this activity should be dedicated to the groups of girls demonstrating their effects.



Handout: Harry Potter Clip Contents

Choose a User Name

(15 minutes)

Students will be using software that allows them to share messages with each other. In order to send and receive messages, each girl will need to create her own unique user name and password. Girls can also add an avatar image to their profile (which will show up when they use the messaging software). See the Create a User Name information sheet for instructors and handout for students.

You will need to give a list of the girls' user names to one of the researchers so that each student can be given access to the messaging software. (The user names will also be used

to create personal web pages for the girls.) Be sure to collect a list of all the girls user names.



Handout: Choose a User Name

Handouts and Additional Materials

In this appendix you will find the following handouts and additional materials that you may need during Part 1 of the Arts & Bots workshop:

- **Expressive Robots and Robotic Art** – This is an informational guide to the robots pictured in the handout below. It contains background information on the robots pictured in the student handouts to aid instructors in guiding discussion.
- **Expressive Robots and Robotic Art Images** – This handout is used to illustrate how robots can be used for emotional expressiveness and communication in the introduction.
- **Circuit Basics** – Covers how circuits work and how to make them.
- **How to Connect to the Hummingbird** – This handout covers how to hook up servos, speakers, LEDs, motors, and vibrating motors to the Hummingbird. It also gives brief descriptions of what the components do or how they work.
- **Individual Activity 1: Scavenger Hunt** – This is the handout that girls take home to use in the Scavenger Hunt activity.
- **Harry Potter Clip Contents**
- **Choose a User Name (for Instructors)** – Step by step instructions for creating a messaging user name and password. Includes additional notes for instructors.
- **Choose a User Name** – Step by step instructions to walk students through creating their messaging user name and password.

Expressive Robots and Robotic Art

Background Information for Instructors

Below are descriptions of various robots and robotic art pieces that are expressive, communicative, musical, kinetic, and beautiful. Background information taken from the various robots' websites is included for your reference.

Keepon – by Marek Michalowski and Hideki Kozima

Keepon can turn, nod, rock side-to-side, and bob up and down allowing him to direct attention and express emotion. Keepon can also detect beats using its microphone and is a great dancer.

Two Keepon videos are included on the CD accompanying this curriculum (they are also available on beatbots.org). One shows Keepon being attentive to a toy and expressive when he sees a face. The other shows Keepon dancing.

From <http://beatbots.org>:

“Keepon is a small creature-like robot designed to interact with children by directing attention and expressing emotion. Keepon's minimal design makes its behaviors easy to understand, resulting in interactions that are enjoyable and comfortable—particularly important in our research on human social development. Keepon has soft rubber skin, cameras in its eyes, and a microphone in its nose.

“Keepon has been used since 2003 in research on social development and communication. We have studied behaviors such as eye-contact, joint attention, touching, emotion, and imitation between Keepon and children of different ages and levels of social development. In the case of children with autism and other developmental disorders, we have had encouraging results with the use of Keepon as a tool for therapists, pediatricians, and parents to observe, study, and facilitate social interactions.”



On Beyond Duckling – by Ian Ingram

This bird-like robot swims on a pond and performs a bird mating ritual. It has a forlorn, sad appearance when the mating ritual is not answered.



From <http://www.ingramclockworks.com>:

“A new species of urban, mechanical waterfowl native to southwest Pennsylvania, found exclusively on ponds in Schenley Park. First described by Ian Ingram, the Ingram Clockworks resident naturalist, the machine has been observed engaging in what appears to be an unrequited mating ritual while rowing around its natural habitat.”

Augmented Fish Reality – by Ken Rinaldo

In this art piece the fish's movements control the movement of the pedestal holding their fish bowl. Projections from cameras inside the fish bowls create the feeling of being inside the bowl.

From <http://accad.osu.edu/~rinaldo>:

“Augmented Fish Reality is an interactive installation of 5 rolling robotic fish-bowl sculptures designed to explore interspecies and transpecies communication. These sculptures allow Siamese Fighting fish to use intelligent hardware and software to move their robotic bowls - under their control. Siamese fighting fish have excellent eyes which allow them to see for great distances outside the water. They have color vision and seem to like the color yellow.



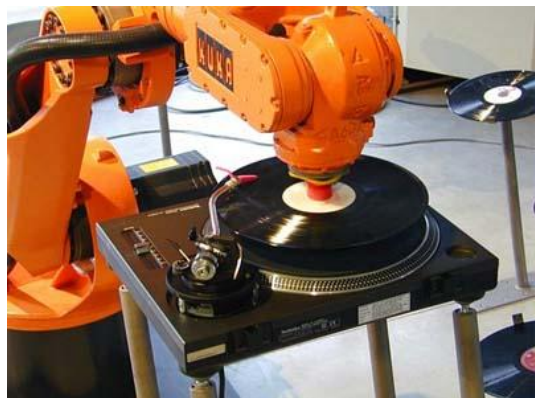
“This design uses 4 active infrared (IR) sensors around each bowl which allow the fish to move forward & back and turn the bowls. By swimming to the edge of the bowl the fish activate motorized wheels that move the robots in that direction. Humans will interact with the work simply by entering the environment. In past artworks I have found that the Siamese fighting fish move toward humans, presumably because they associate humans with food. Still, these are robots under fish control and the fish may choose to approach and/or move away from the human participants and each other. These bowls consist of a living environment of peace lillies, which help to absorb the waist stream from the fish. The bowls and robots are designed to allow the fish to get to within 1/4 inch of each other for visual communication between the fish, both male and female. Small lipstick video cameras mounted on forty-five degree angles under two of the bowls image the interior of the fish bowls as well as humans in this environment and these images are intercepted with transceivers and projected back to the walls of the installation and give human participants a sense both looking to the interior of the tanks and feeling as if they are immersed in the tanks.”

Juke_bots – by Matthias Gommel, Martina Haitz, and Jan Zappe

Juke_bots is two robotic arms that create musical performances by manipulating records on a turntable, like a DJ.

From <http://www.robotlab.de>:

“juke_bots is an installation, in which two robot arms act like a DJ. Each machine is circularly surrounded by records which it can select and pick up. In front of each arm a record player is positioned so that the robot can thread the records under its needle.



“Without putting down the record on the turntable the robot can play and scratch it, both forward and reverse, speed up or slow down the music, and thus generate new sounds with own characteristics.

“The 'juke_bots' can be displayed either as an autonomous performance or as an 'interactive' installation. In this version the audience can start the robots either by motion sensor or by throwing coins in the slot of the interface.”

Blue Butterfly Machine – by Rebecca Horn

This art piece combines delicate butterfly wings with small gears and motors creating a machine that is striking and beautiful. The intricate mechanism makes the butterfly wings flutter delicately.

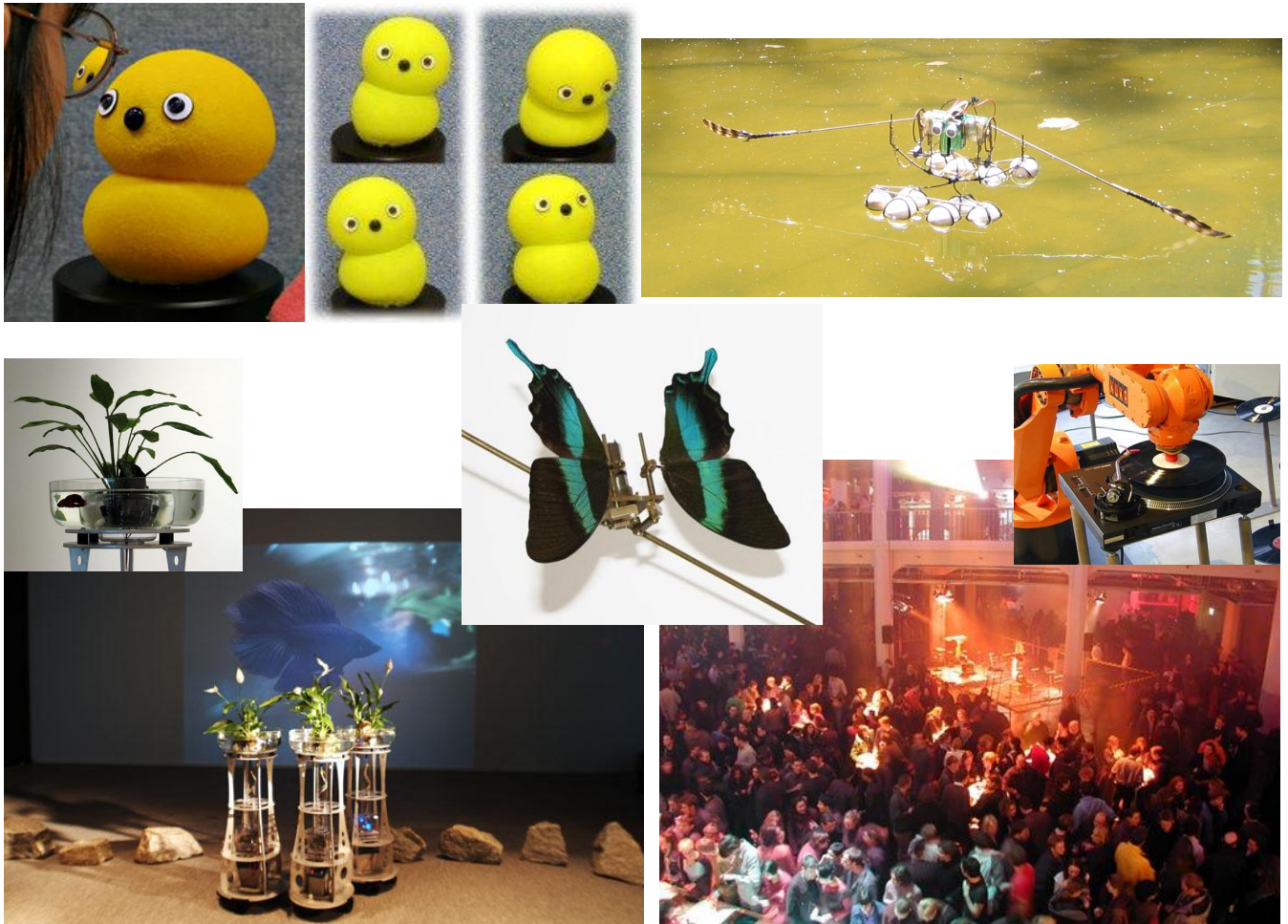
From

http://philonous.typepad.com/musings_from_the_lehigh_v/2006/11/an_obsession_wi.html:

“Illness became a metaphor for [Horn] as an artist and incapacity led to her insignia: use of the lightest of materials, feathers most of all; an obsession with the fragility of the body, with what can be appended to it and how it can be modified into an artwork ...



“What's striking about the [work] is also the ironic way in which Horn juxtaposes the fragile natural beauty of the butterfly wings with the artificial and mechanical ... calling to mind the tension between such delicate natural beauty and the artistic impulse to freeze nature, to ‘fix [it] in a formulated phrase,/ And when [it is] formulated, sprawling on a pin,/ When [it is] pinned and wriggling on the wall,’ to wonder at its fragility and impermanence.”

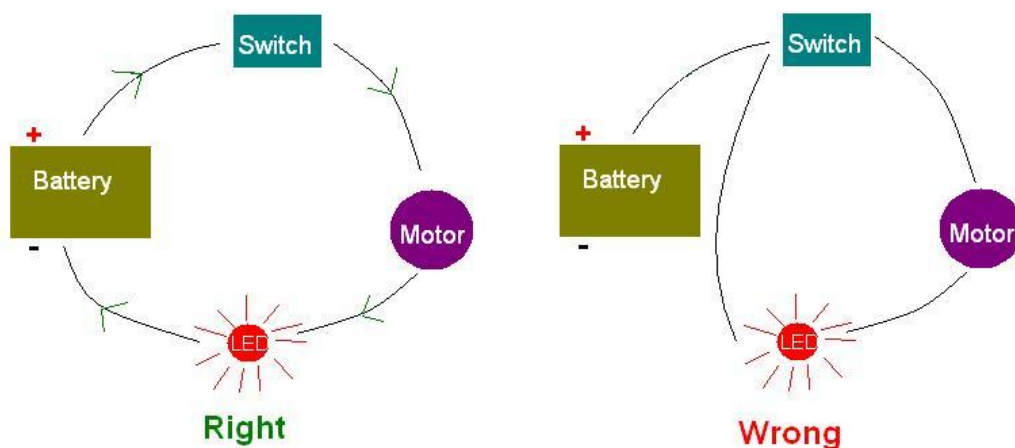


Arts & Bots | Carnegie Mellon University & University of Pittsburgh

How Circuits Work

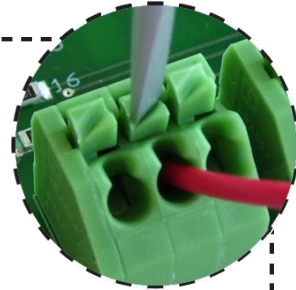
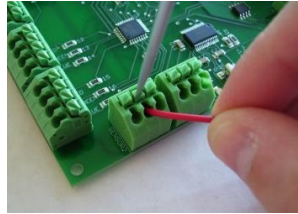
To get your circuit to work, you need a battery and some alligator clips. You then have a choice of other things to put in the circuit – LEDs, motors, and/or switches.

The most important thing you need to know to get your circuit to work is that everything you put in should make a loop or chain. This is because electricity flows along the wire, and everything it flows through turns on. Switches work by breaking the loop when they're turned to off. Look at the following drawings for right and wrong ways to hook up circuits. You can from the diagram that when the circuit is set up right, electricity flows from the + side of the battery to the minus side. You don't need to have all the circuit parts show in the example – you could have just a battery and a motor, for example.



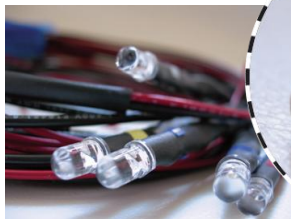
How to Connect to the Hummingbird

Before you do any connecting, make sure the Hummingbird is turned off!



Inserting wires in to locking terminals:

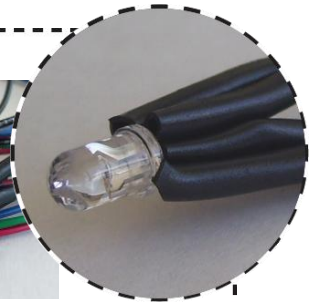
Use a screwdriver to depress the locking tab. Insert the wire in to the hole. Release the tab and gently pull on the wire to see if it stays in place.



LED

LED stands for 'Light Emitting Diode'. LEDs are small lights that can come in many different colors depending on the type of material used to make them.

Connect to one of the 'LED' ports. Connect the red wire to one of the '+' ports. Connect the black wire to the adjacent '-' port.



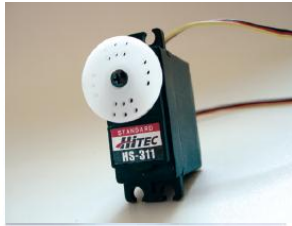
Full-color LED

Full-color LEDs are like three single-color LEDs in one. They can produce red, green, and blue light. By varying the intensity of the three colors, the combined light output can be any color.

- + Plug in to one of the 'ORB' ports.
- + Connect the red wire to one of the 'RED' ports.
- + Connect the green wire to the adjacent 'GREEN' port.
- + Connect the blue wire to the adjacent 'BLUE' port.
- + Connect the black wire to the adjacent '-' port.

How to Connect to the Hummingbird

Before you do any connecting, make sure the Hummingbird is turned off!



Servo

Can move back and forth about 140°. The HS-311 servo is a medium power servo.

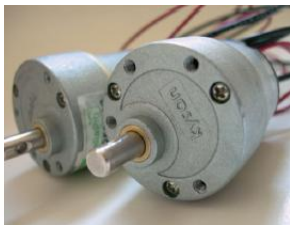
Plug in to one of the ports labeled 'SV1' through 'SV4'. Make sure the black wire is towards the edge of the board (plugged in to the pin labeled 'B').



Speaker

The Hummingbird has a built-in audio amplifier that supports MP3 and WAV files.

Connect the two speaker wires to the 'SPEAKER' port. Connect the speaker cable to the 'MONO' jack on the Hummingbird and the headphone jack on your computer.



Motor

Spins forwards or backwards at varying speeds.

Connect to the 'MOTORS' ports. Connect the red wire to one of the '+' motor ports. Connect the black wire to the adjacent '-' motor port.



Vibration Motor

Has an off-center weight that makes the motor vibrate when it spins.

Connect to the 'VBR' ports. Connect the red wire to one of the '+' ports. Connect the black wire to the adjacent '-' port.

Scavenger Hunt!

Find objects in your house that have one or more of the following: Buttons & switches, motors, LEDs, something you can program, and something you think is robotic. You can't use 'Computer' as an answer in any category. Bonus for finding things that can fit in two or more of the categories!

Something that has
buttons or a switch

Something that has
a motor in it

Something robotic

Something that you
can program

Something that has an
LED or indicator light

Harry Potter Clip Contents

Dementor Battle

0:00-0:11 Dementors arrive
0:11-0:22 Dementors attack
0:22-0:27 Harry tries to get his wand
0:27-0:32 Harry casts his patronus
0:32-0:35 Dementor flees

Neville Finds Room

0:00-0:27 Door to the Room of Requirement appears and opens

Neville vs. Practice Dummy

0:00-0:08 Dummy advances
0:08-0:14 Neville tries to disarm it

Stupefy practice

0:00-0:12 Nigel practices stupefy on Harry

Harry and Cho practice

0:00-0:04 Nigel is floating
0:04-0:09 Cho loses concentration

Azkaban Escape

0:00-0:05 Explosion
0:05-0:19 Thunder and lightning

Umbridge Breaks Through

0:00-0:10 Rumbling
0:10-0:16 Mirror shatters
0:16-0:35 Final blow

Dumbledore Escapes

0:00-0:06 Phoenix arrives
0:06-0:10 Phoenix leaves

Voldemort Battle

0:00-0:02 Voldemort casts spell
0:02-0:12 Destruction!

Choose a User Name (for Instructors)

Students will be using software that allows them to share messages with each other. The messaging software depends on a special server to send and store messages. CMU will run the message server for the workshop. Access to the messaging server, and thereby the ability to send messages to members of the group, requires a user name and password.

Each girl will need to create her own unique user name and password. Preferably she should also add an avatar image to her profile. Follow the instructions below to create a user name, password, and avatar image (special notes to instructors are included in italics).

1. Go to the TeRK website: <http://www.terk.ri.cmu.edu>. *(TeRK is a project run by CMU. Arts & Bots is a part of the TeRK project and uses TeRK software.)*
2. Click the 'Become a member' link in the top right corner of the homepage.
3. Enter an email address. A confirmation email will be sent to the email address so it should be one that you can access from the computer you are currently using. *(If a student does not have an email address, you may use an instructor's email address. However because the TeRK database will only allow one user to sign up for each email address we recommend changing the email from 'instructor@site.org' to 'instructor+studentUserName@site.org'. Most mail programs ignore anything after the plus sign unless you have special filters set up. However, if no email arrives, be sure to check your spam box.)*
4. Enter a screen name. (Hint: your screen name does not have to be very complicated because TeRK has a relatively small set of users.) Before going any further write down your screen name in your design notebook.
5. You can select an avatar image now or later.
6. Hit 'Submit' at the bottom of the page.
7. Open the confirmation email sent from the TeRK site and click the confirmation link.
8. Enter your screen name and choose a password. Hit 'Submit' to complete your registration.
9. If you did not previously select an avatar image, you can do so by clicking 'Edit Profile'. If you want to change your avatar later, you can log in to the TeRK site using your user name and password and then edit your profile.
10. An instructor is making a list of everyone's user names so that you can use the messaging software. Make sure the instructor writes down your user name on the list. *(Collect all of the girls' user names and give them to one of the researchers so that they can be given access to the messaging server being run at CMU.)*



Choose a User Name

Some of the software you will be using during the workshop allows you to send messages to other workshop participants. In order to send and receive messages, you need your own user name (also called a screen name). You can also upload an avatar image that will show up by your user name when you send messages. Follow these steps to create a user name and password and upload an avatar:

1. Go to the TeRK website:
<http://www.terk.ri.cmu.edu>.

2. Click the 'Become a member' link in the top right corner of the homepage.

3. Enter an email address. A confirmation email will be sent to the email address so it should be one that you can access from the computer you are currently using.

4. Enter a screen name. (Hint: your screen name does not have to be very complicated because TeRK has a relatively small set of users.) Before going any further write down your screen name in your design notebook.

5. You can select an avatar image now or later.

6. Hit 'Submit' at the bottom of the page.

7. Open the confirmation email sent from the TeRK site and click the confirmation link.

8. Enter your screen name and choose a password. Hit 'Submit' to complete your registration.

9. If you did not previously select an avatar image, you can do so by clicking 'Edit Profile'. If you want to change your avatar later, you can log in to the TeRK site using your user name and password and then edit your profile.

10. An instructor is making a list of everyone's user names so that you can use the messaging software. Make sure the instructor writes down your user name on the list.

The screenshot shows the 'Become a Member' registration form. The form has a title 'Become a Member' and a paragraph explaining the registration process: 'Fill in the membership information below. We will send a confirmation email to the address you provide with a hyperlink to complete your membership registration. Only your email address and screen name are required to register. Your email address will be kept private, but other information will be viewable by others on your profile page. See our Privacy Policy.' The form contains several input fields: 'First Name:', 'Last Name:', 'Email *:' (with the example 'jdoe@yahoo.com'), 'Screen Name *:' (with the example 'Jane'), 'Avatar Image:' (with a file path 'C:\Pictures\kiccoro.gif' and a 'Browse...' button), 'City:', 'Country:', 'Favorite Robots (Real or fictional):' (a large text area), and 'Tell us about your experience and interests:' (another large text area). At the bottom, there is a checkbox for 'Email me TeRK events and news:' and two buttons: 'Reset' and 'Submit'.

Part 2: Building Basics – *Design, build, and program a simple robot.*

- The Design Process
 - Build Your Own Robot
 - Write and Share Expressive Robot Programs
-

Timeline

Activity	Estimated Time	Elapsed time
The Design Process	5 minutes	0:05
Design / Plan	15 minutes	0:20
Build / Implement Designs	1 hour	1:20
Group Activity: Robot Dance Party	30 minutes	1:50
Documenting Designs	10 minutes	2:00
Documenting Designs: Sharing Your Design Journey	30 minutes	0:30
Group Activity: Show and Tell	20 minutes	0:50
Group Activity: Improv	10 minutes	1:00
Group Activity: Charades	30 minutes	1:30
Introduction to Messenger	15 minutes	1:45
Individual Activity 2: Share a Story	5 minutes	1:50

Materials:

Design notebooks	(1 / student)
Pencils, markers	
Computers with internet access	(1 / 2-3 students)
Hummingbird, power supply, USB cable	(1 / student)
Servos	(2 / student)
LEDs (full color or tri-color if available)	(2 / student)
Cardboard and foam board	
Craft supplies	
Tools	
Camera	
Scanner (optional)	

Learning Goals

		Part 2: Building Basics - Activities					
		The Design Process	Planning	Prototyping	Documenting Designs	Group Activity: Charades	Individual Activity 2: Share a Story
Learning Goals	Technical knowledge application						
	Demonstrate the use of hand tools such as glue guns, screwdrivers, and x-acto knives.			X			
	Construct a simple circuit.			X			
	Attach servos, vibration motors, and LEDs to a controller.			X			
	Create short programs using a graphical programming language.					X	X
	Design process knowledge						
	Describe all of the steps in the design process.	X					
	Explain why design is an iterative/experimental process.	X					
	Design process application						
	Planning						
	Create a technological design or modify a feasible technological design to meet an individual or community need.		X				
	Formulate design goals.		X				
	Identify design constraints.		X				
	Identify available resources.		X				
	Identify the limitations of an available resource.		X				
	Draw a sketch or diagram of a technological design.		X				
	Prototyping						
	Assemble the resources needed in order to implement a design.			X			
	Construct a physical prototype of a technological design.			X			
	Documenting designs						
	Describe a technological design using writing or photography.				X		

The Design Process

(5 minutes)

When people want to create or improve something—anything from buildings to clothing to cakes—there are a set of steps that they follow in order to turn their ideas into reality. These steps are known as **the design process**.

Suppose that you want to create a new recipe. You first need to spend some time **planning**: you have to figure out what kind of dish to make and what ingredients to use. You might decide you want to make a new type of chocolate chip cookie with some special ingredient or flavor added, so you would have to plan out how much of your new ingredient you want to add to an existing recipe.

You would then bake a batch of cookies following your plan: this is called **prototyping**, producing a physical representation of your idea to see how feasible it is. Once you've baked the cookies, naturally you would want to eat one! You would **test** your cookie to see if it tastes the way that you wanted. If the cookie doesn't quite turn out the way you hoped, you would start **troubleshooting**—figuring out what might have gone wrong and how you might fix it. For example, if your cookies are too crunchy, you might need to bake them at a lower temperature or for a shorter amount of time. Or if you can't taste your secret ingredient, you might need to add more of it, or use something slightly different. If you're making almond-flavored cookies, for example, and you tried adding almond extract to the dough, they might not taste nutty enough: you might want to add some chopped almonds, too.

Once you think you've figured out what the problem is, you can **iterate**—that is, you'll go back and repeat the design process, remembering what you've learned. So you'll make a new plan for how to change your recipe, you'll bake another batch of prototype cookies, and you'll evaluate how they turned out. You can keep repeating the process until you've got a cookie recipe that you really love.

Another way to make your cookie recipe better is to get **feedback** from other people. You can share some of the cookies with your friends and then listen carefully to their reactions.

As you're creating your cookies, you'll want to **document** what you've been doing, such as what changes you're making to your recipe and how the results turn out. Once you're happy with your cookies, you'll want to be sure you write down the recipe you used so you can make them again or share the recipe with your friends.

In summary, the steps in the design process are:

- Planning
- Prototyping
- Testing and troubleshooting
- Evaluating and critiquing designs
- Documenting designs



Handout: The Design Process

Planning

(15 minutes)

In our experience students often worked with the materials right in front of them. If students aren't already familiar with the craft materials available, you can aid their creativity by making them aware of the variety of materials available to them. It may help students to look briefly at the available materials before or during their planning.

**Handout: Instructions for Design Planning and Building**

Today you are going to build an expressive robot that has a body, 2 servos, and 2 LEDs. The servos can control arms, legs, ears, antennae, wings, fins, heads, tails, and so on. The LEDs can be used in the eyes, ears, nose, antennae, or some other part of your robot.

How will your robot look?

Before you begin building you need a plan. Draw one or more pictures of your robot. Label the moving parts and the LEDs.

Prototyping

(1 hour)

Have the students refer to the handout and their design sketches as they start building. Be sure students know what materials are available to them. Remind students that this is the first iteration through the design cycle. The robots do not need to be perfect or polished at this point. In future sessions, after testing and evaluating their robot designs, students will be revising their designs, adding additional components (more servos, more LEDs, motors, and a speaker), as well as continuing to decorate and personalize their robots.

In our experience, kids can spend a long time building and personalizing their robots. We have specifically tried to keep building time short at this point in the curriculum so that students can get through all stages of the design cycle and iterate on their designs before getting too attached to the first iteration of the robot.

Group Activity: Robot Dance Party

(30 minutes)

Have the girls bring in music from home (or provide a selection of music). Each girl should choose a 1-minute music clip, and programs her robot to dance to the music. If they want to, pairs or groups of girls can work together to coordinate their dances.

Some students will take longer than others to assemble their robots. If there are students who do not yet have working robot, assist them in completing their robots during this time.

After 25 minutes, go around the room and ask girls to demonstrate their dancing robots.

Documenting Designs

(5-10 minutes)

Tell the students about the web pages/posters they will be making in a future session. The photos and documentation activities they do today will help them with their web

pages/posters. Each girl's web page/poster is a place for her to share the story of how she created her own unique robot.

Take pictures of each of the robots. If you have access to a scanner, you can scan the girls' design sketches or you can take photos of the sketches for the web page.

Have the girls write in their notebooks for 5 minutes. They may answer one or more of the reflection questions or write about something else related to their work today.



Handout: Reflection 1

Reflection questions:

- What was the biggest challenge you faced in designing and building your robot today? How did you solve it?
- Why did you choose to make your robot look the way it looks?
- Is there something you want to add to your robot?

Documenting Designs: Sharing Your Design Journey (30 minutes)

At this point in the workshop, the girls should be ready to begin creating web pages that document their ‘design journey.’ (If a web page is not available or is difficult to access, they can do this on a poster).

Introduce students to the documentation website, robotdiaries.org. Show them how to log in and edit their personal page. Students will use these web pages throughout the workshop to document the process of designing and building their own robot. They should refer to their answers to the reflection questions as a source of material for their sites.



Handout: Website Instructions

In preparation for this activity you should upload the pictures you have taken of the students’ robots over the course of the workshop. Follow the Uploading Photos instructions on the Website Instructions for Instructors page included in the additional materials section.



Additional Materials: Website Instructions for Instructors

Group Activity: Show and Tell (20 minutes)

Give each girl a chance to show her robot to the rest of the group. If you have a large group, this can be done in smaller groups of 4 to 6 girls.

Girls should show their robot and talk briefly about its expressive features. Because feedback is an important part of the design process, encourage other girls to ask questions, complement good ideas, give suggestions, etc. Before beginning the show and tell you may want to review the feedback guidelines handout to encourage kind and constructive feedback and help the girls make the most of the feedback they receive.



Handout: Feedback Guidelines

Group Activity: Improv (10 minutes)

This game is designed to stimulate thinking about expression as well as get girls moving around.

Players should stand in pairs facing each other. Ask players to look at each other with fear. Sounds and physical movement are encouraged. Then ask everyone to change partners. Ask players to look at each other with envy. Change partners again. Ask players to look at each other joyfully.

Then yell any of the three emotions. Players need to find the partner with which they did that emotion and do it again. Repeat this several times. Then ask players to walk around

the room. When they meet one of their 3 partners, they return to the look that went with that partner. In between partners they stay neutral.

You can try more or different emotions, of course. A list of emotions is provided in the charades game description below.

Group Activity: Charades

(30 minutes)

Have the girls brainstorm a list of emotions as a group. Display the list in a visible place so that girls can reference it throughout the activity. If you are short on time, you can supply the list of emotions. Sample emotions include: happy, sad, angry, afraid, surprised, disgusted, ambivalent, anxious, bored, confused, depressed, doubtful, envious, embarrassed, frustrated, guilty, proud, regretful, shameful, euphoric, grateful, hopeful, hysterical, lonely, in love, paranoid, pity, pleased, enraged, compassionate, calm, suspicious, ecstatic, melancholy, joyful, excited, nervous, meditative, shocked, lovesick, cheerful. You'll want a list of at least as many emotions as there are girls (3 emotions for each team of 2-3 girls).

Split into teams. Ideally you should make 3 or more teams of 2-3 girls each. Have each team draw 3 of the emotions from a hat.

Give the teams time to program roboticons for their selected emotions. Then have them present their roboticons to the other teams by playing them on their own team's robot.

Scoring Option 1 (shorter): The other teams try to guess which emotion is being expressed. Award 1 point to the first team to make the right guess.

Scoring Option 2 (longer): The guessing teams consult amongst themselves and come up with a single guess as to which emotion is being expressed. After each team has reached a consensus, share the guesses. Award 1 point to each team that guesses correctly. Award 1 point to the presenting team for each correct guess they receive.

If the girls will be creating web pages, this may be a good opportunity to take some short video clips of the robots in action.

Introduction to Messenger

(15 minutes)

Briefly show students how to use the messenger portion of the software. Have each girl or each group post their favorite sequence created during charades to the public message board, and then play a sequence posted by another group.



Handout: Refer to the Software Instructions (located in the Appendix).

Individual Activity 2: Share a Story

(5 minutes)

Individual activities are design for students to complete on their own, either at home or during free time at the workshop location. Students should expect to spend roughly 15-30 minutes on this activity.

- Either write a diary entry for your day or week or write a story.
- Make at least 3 roboticon programs for your robot that help express your diary entry or tell your story.
- Use the RoboticonMessenger to share what you've created with the group. If you wrote a personal diary entry, you do not need to share the text of your diary entry; you can choose to share just the roboticons and the emotions they represent.
- Read the stories shared by your friends and try playing their roboticons on your own robot. Can you feel the emotions they are expressing? Do you think the roboticons look different on your robot than on your friends' robots?



Handout: Individual Activity 2: Share a Story

Individual Activity Follow-Up

(the following session)

Address any problems students faced with the individual activity. If you have time, invite 1 or 2 students to share their story and play the roboticons on their own robot.

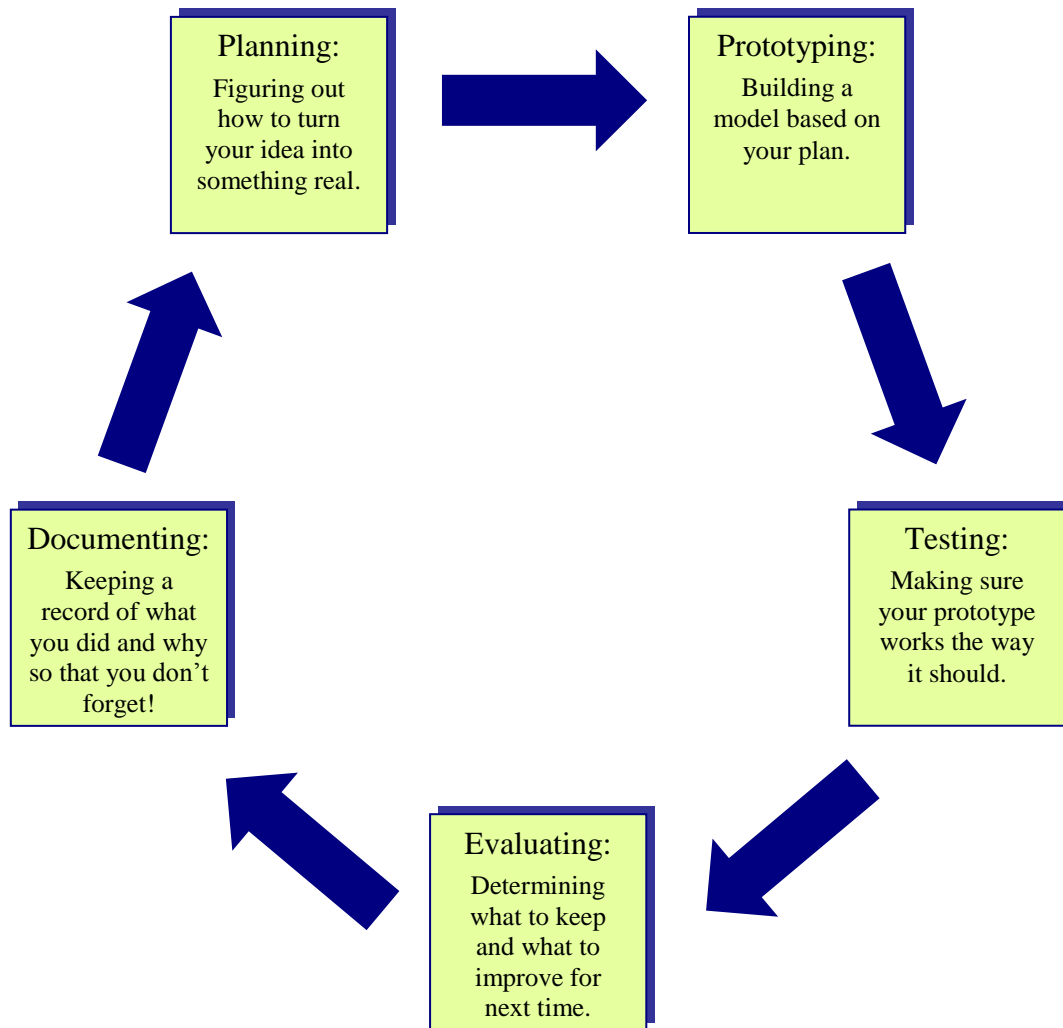
Discussion questions:

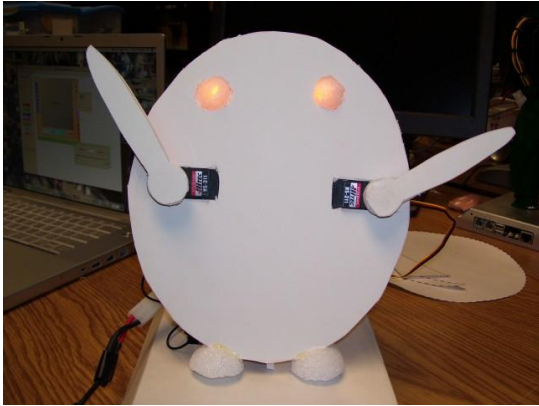
- Can you feel the emotions the robots are expressing?
- Do the roboticons look different on your robot than on your friends' robots?
- Are there any group favorites? What makes those particular roboticons stand out?

Handouts and Additional Materials

- **The Design Process** – A diagram illustrating the design process.
- **Instructions for Design Planning and Building** – Contains instructions for planning and building the first version of the robot. Also contains pictures of sample robots to serve as inspiration and provide visual clues for building.
- **Reflection 1**
- **Website Instructions for Instructors** – Directions for setting up the students' personal web pages and uploading photos to Flickr for the students to include on their web pages.
- **Website Instructions** – Covers logging in the robotdiaries.org site, adding content, and changing the theme for girls' personal pages.
- **Feedback Guidelines** – Guide for giving and receiving effective feedback.
- **Individual Activity 2: Share a Story**

The Design Process





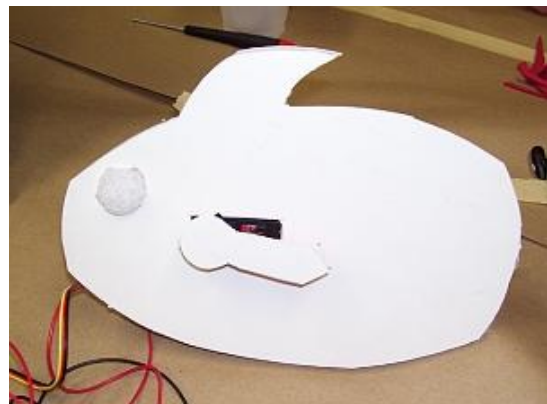
Today you are going to build an expressive robot that has a body, 2 servos, and 2 LEDs. The servos can control arms, legs, ears, antennae, wings, fins, heads, tails, and so on. The LEDs can be used in the eyes, ears, nose, antennae, or some other part of your robot.

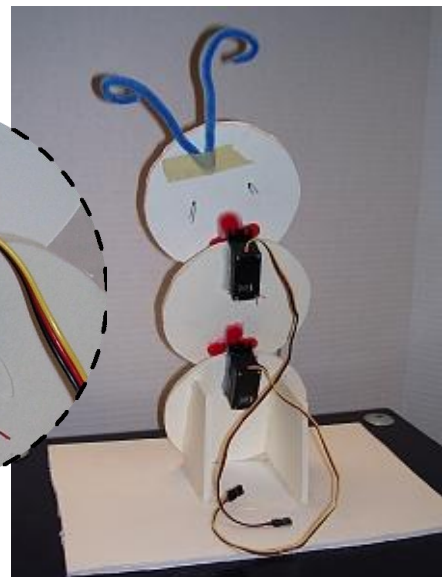
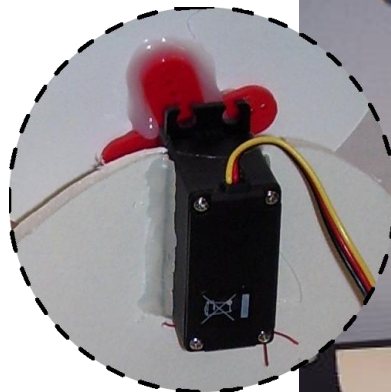
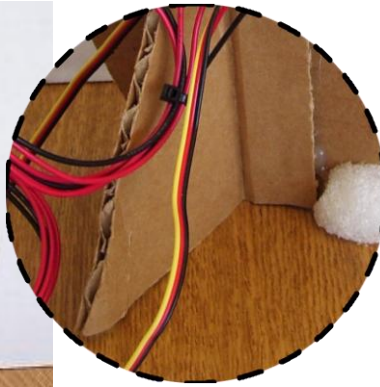
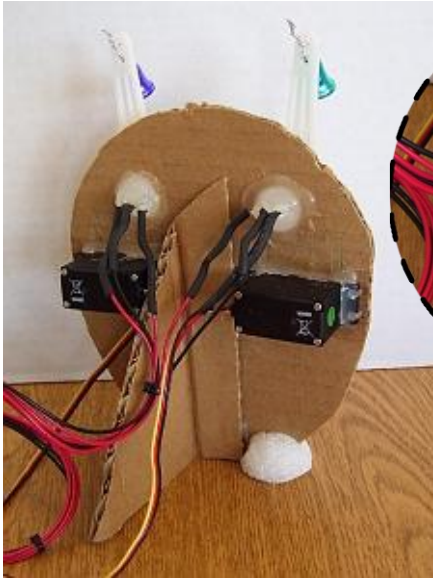


How will your robot look?



Before you begin building you need a plan. Draw one or more pictures of your robot. Label the moving parts and the LEDs.





Reflection 1

Reflecting on and documenting our experiences as we design something helps us to improve our design. Your reflections today will also be a source of material for your web page or poster.

Reflection Questions:

- What was the biggest challenge you faced in designing and building your robot today? How did you solve it?
- Why did you choose to make your robot look the way it looks?
- Is there something you want to add to your robot?

Webpage Instructions for Instructors

Girls will be creating their own web pages in order to document their design journey.

Creating User Accounts

You will need to create a user account for each student. We have started this process but you will need to make the login name and password for each student. Use the same login name that girls created for the messaging software.

There is a webpage ready for each girl. The pages use different color names, for example robotdiaries.org/red, robotdiaries.org/blue, etc. The list of colors available for your particular workshop is included with your curriculum materials.

1. Go to <http://www.robotdiaries.org/FirstColorName> (see the list of colors for your workshop site).
2. Log in to the site with Username:
 Password:
3. Click 'Log In'
4. Log in with Username:
 Password:
5. Click the 'Admin' link at the top of the page.
6. Click the 'Manage' link on the top right.
7. Click the 'Users' tab.
8. Click the '+ New User' button.
9. Enter the girl's first name, an email address (you can use info@robotdiaries.org), and login name. The Group should be set to Member. Enter the password and confirm the password (use '...' as the default password). Then click the 'Add User' button.
10. Click the 'Settings' link on the top right.
11. Change the Site Name from 'My Awesome Site' to the girl's login name or first name. Then click the 'Update' button.
12. Repeat the steps with a new color for each girl.

Uploading Photos

Students will be able to include images, videos, and sounds on their web pages. We have set up a Flickr account where you can upload photos and videos so that you they will be available for the girls to add to their websites. We have already uploaded some images of the technical components the girls will be learning about and screenshots of the robot programming software in case girls would like to use these images on their websites.

Because photos uploaded to the Flickr account will be public, limit uploads to images of robots and other items created by the students. **Do not upload images of the students.**

To upload new images or videos:

1. Go to <http://www.flickr.com/photos/robotdiaries>
2. Click 'Sign In' in the top right corner.
3. Sign in using ID:
Password:
4. Click 'Upload Photos'
5. Click 'Choose Photos'
6. Select the photos you want to open on your computer and click 'Open'.
7. Make sure that 'Public' is selected under Set privacy.
8. Click 'Upload Photos'.

Webpage Instructions

Logging In

1. Go to <http://www.robotdiaries.org>
2. Log in to the site with Username:
Password:
3. Click on your user username (or page name) in the column on the right.
4. Click the 'Log In' link (on the right under Welcome Guest!).
5. Enter your Username and Password. Your default password is '...' until you change it.

Changing Your Password

1. Log In to your page. Your default password is '...'.
2. Click the 'User Controls' link in the column on the right.
3. Enter your new password, confirm the password by entering it again, and click the 'Update' button.

Posting

1. To add a new post to your website, click the 'Write' link at the top of the page (you must be logged in).
2. You can now add text, audio, links, photos, quotes, or videos to your page by clicking on the appropriate tab. You will probably use 'Photo' and 'Text' most often.
3. Let's say you want to add a photo. Click the photo tab. Then either 'Browse' to find the photo on your computer or enter the URL (web address) where the photo is located online. In the 'Caption' box, enter the text that you want to accompany your post.
4. Click 'Publish' when you are ready to add your post to your website.

Flickr Photos

We have created a Flickr account so that you can easily add pictures from the workshop to your web page. Follow the instructions above for posting photos. When it is time to select the photo, you will enter a URL in the box that says 'From URL?' (you will not use the 'Browse' button).

To get the URL of a photo in the Flickr account, follow these steps:

1. Go to <http://www.flickr.com/photos/robotdiaries>
2. Find the picture that you want to include in your web page and click on it.
3. Click and drag the photo to the address bar.
4. When you release the photo in the address bar you should see a webpage showing just the photo and nothing else (no text or other images).
5. Copy the URL (the web address).

To copy the URL you must highlight the full address in the address bar. Click on the address with the right mouse button.

Select 'Copy' from the menu that appears.

6. You can now paste the URL in to the box that says 'From URL?' to post it to your web page.

To paste, click in the box with the right mouse button.

Select 'Paste' from the menu that appears.

Themes

You can change the color theme for you page.

1. First click 'Write' at the top of your page.
2. Then click 'Extend' on the top right.
3. Click the 'Select' button to choose the theme you want.
4. Click 'View Site' on the top right to return to your site.

Feedback Guidelines

Giving and receiving feedback is an important part of the design process. Here are some guidelines for giving good feedback and for receiving feedback from others.

The person giving feedback should:

ALWAYS balance a negative comment with a positive one.

ALWAYS be specific. Suggest a specific improvement instead of just saying something needs improvement, or give examples of familiar products that have desirable features that could apply this concept.

ALWAYS ask precise clarification questions.

NEVER make a vague statement (e.g., “I don’t like the way it looks”).

NEVER attack the character of the designer.

NEVER make comments about the designer instead of the idea.

The person receiving feedback should:

ALWAYS listen actively and make notes.

ALWAYS pay attention to both positive and negative aspects of the feedback.

ALWAYS thank the person providing feedback for particularly good insights.

NEVER defend or argue about the negatives brought up.

NEVER turn clarification questions or answers into justifications for every aspect of the system. **ONLY** ask questions to help you understand the comments.

NEVER take the comments personally.

Individual Activity 2: Share a Story

- Either write a diary entry for your day or week or write a story.
- Make at least 3 roboticon programs for your robot that help express your diary entry or tell your story.
- Use the RoboticonMessenger to share what you've created with the group. If you wrote a personal diary entry, you do not need to share the text of your diary entry; you can choose to share just the roboticons and the emotions they represent.
- Read the stories shared by your friends and try playing their roboticons on your own robot. Can you feel the emotions they are expressing? Do you think the roboticons look different on your robot than on your friends' robots?

Part 3: Advancing Your Design – *Iterative cycles of design, building, and structured use activities.*

- Expand Your Design
 - Expand Your Robot
 - Write and Share Longer Programs
-

Timeline

Activity	Estimated Time	Elapsed time
Individual activity follow-up	5-10 minutes	0:10
Iteration Example: Robotic Desk Lamp	10 minutes	0:20
Design Reflection	5 minutes	0:25
Making Changes 1: Brainstorming and Building	60 minutes	1:25
Introduction to speakers	5 minutes	1:25
Group activity: Robot Mad Libs	25 minutes	1:50
Document designs + reflection	10 minutes	2:00
Individual activity 3: Robot meets world		
Individual activity follow-up	5-10 minutes	0:10
Making Changes 2: Design reflection and brainstorming	15 minutes	0:25
Building (adding new components to robot)	45-55 minutes	1:20
Prepare for show and tell	15 minutes	1:35
Group show and tell and feedback	20 minutes	1:55
Document designs + reflection	10 minutes	2:05
Individual activity 4: Pull the Chain		
Individual activity follow-up	5-10 minutes	0:10
Introduction to sensors	15-20 minutes	0:30
Introduce idea of Play	5 minutes	0:35
Planning for Play	30 minutes	1:05
Design/re-design and building for play	50 minutes	1:55

Documenting designs + reflection	10 minutes	2:05
Individual activity 5: Silly messages		
Individual activity follow-up	(optional)	
Continue working on play	75 minutes	1:15
Group Activity: Robot makeover	30 minutes	1:45
Survey	15 minutes	2:00
Group Activity: Play Presentation	30-60 minutes	1:00

Materials:

Design notebooks	(1 / student)
Pencils, markers	
Computers with internet access	(1 / 2-3 students)
Hummingbird, power supply, USB cable	(1 / student)
Servos	(2 / student)
LEDs	(4 / student)
Vibration motors	(2 / student)
Motors	(1 / student)
IR distance sensor	(approximately 1 / student)
Light sensor	(approximately 1 / student)
Speaker and speaker cable	
Cardboard and foam board	
Craft supplies	
Tools	
Digital camera	
Scanner (optional)	
Copies from book of fairytales	

Learning Goals

		Part 3: Advancing Your Design - Activities								
		Design / Plan	Build / Implement Designs	Documenting Designs	Group Activity: Mad Libs	Individual Activity: Robot Meets World	Group Activity: Show and Tell	Individual Activity: Pull the Chain	Group Activity: Play	Individual Activity: Silly Messages
Learning Goals	Technical knowledge application									
	Demonstrate the use of hand tools such as glue guns, screwdrivers, and x-acto		X						X	
	Attach servos, vibration motors, and		X						X	
	Create short programs using a graphical				X			X	X	X
	Design process knowledge									
	Describe all of the steps in the design	X	X	X						
	Explain why design is an important part									
	Explain why design is an	X	X	X						
	Design process application									
	Planning									
	Create a technological design or modify a feasible technological design to meet	X							X	
	Formulate design goals.	X	X						X	
	Identify design constraints.	X	X						X	
	Identify available resources.	X	X						X	
	Identify the limitations of an available	X	X						X	
	Draw a sketch or diagram of a	X							X	
	Create a design which utilizes		X						X	
	Modify a design or implementation	X	X	X					X	
	Prototyping									
	Assemble the resources needed in order	X	X						X	
	Construct a physical prototype of a		X							
	Evaluating designs									
	Assess how well a technological design meets an individual or community need.			X	X	X	X	X	X	X
	Judge the benefits and drawbacks of a design modification with respect to	X					X			
	Troubleshooting									
	Identify problems with a technological	X	X						X	
	Devise possible solutions to the problems with a technological design.	X	X						X	
	Documenting designs									
	Describe a technological design using			X		X	X			

Individual Activity Follow-Up

(5-10 minutes)

Address any problems students faced with the individual activity. If you have time, invite 1 or 2 students to share their story and play the roboticons on their own robot.

Discussion questions:

- Can you feel the emotions the robots are expressing?
- Do the roboticons look different on your robot than on your friends' robots?
- Are there any group favorites? What makes those particular roboticons stand out?

Iteration Example: Robotic Desk Lamp

(10 minutes)

Show the Robotic Desk Lamp as a sample robotic technology that went through several design iterations. Show the Robotic Desk Lamp video from the CD. See teacher discussion guide and student handouts.



Video: Robotic Desk Lamp



Handout: Design Iterations of a Robotic Desk Lamp

Design Reflection

(5 minutes)

Based upon your discussion of the homework, ask students to reflect on this question:
How can I make my robot more expressive?



Handout: Reflection 2

Making Changes 1: Brainstorming and Building

(60 minutes)

Remind students of the goal of being able to communicate and express feelings with the robot.

Today, you can make your robot more expressive by adding additional servos, LEDs, motors and sound capabilities. You will have at least two other sessions to work on your additions and robot customization so you can test your robot between sessions and make improvements at the next session. The additional components you can use are: 2 additional servos, 4 additional LEDs, 2 vibration motors, 1 motor, and 1 speaker.

By the end of the final design session you will need to integrate at least three of the five different types of components listed above, but try to integrate as many as you can. For today, try to integrate at least one new component into your robot.

Adding these additional components will require you to re-design parts of your robot. This is all part of the design process. You did a great job building your initial robots, and now you have a chance to improve your robot and make it exactly the way you want it!

Before students begin building, they should spend 20 minutes brainstorming about how they want to add these new components to their robots.



Handout: Redesign Instructions

Introduction to Speakers

(5 minutes)

When students have finished brainstorming, show students the speaker, and demonstrate how to attach the speaker to the Hummingbird. The How to Connect to the Hummingbird handout from Part 1 describes how to connect the speaker.

We have noticed that students sometimes have trouble managing their time during design sessions. To help keep the students on track, we recommend regular reminders about how much design time they have remaining.

Group Activity: Robot Mad Libs

(25 minutes)

This activity is a take-off on the game Mad Libs. Mad Libs is a game where one person asks another for a list of words, and then inserts the words into a story in pre-determined places. The result is a very funny story. In this version of the game, we will insert roboticons into a story along with words.

There are some sample stories (along with slots for roboticons) in the Appendix. However, the girls in the group can also write their own stories for this activity.

There are several ways to run this activity:

- Have the girls brainstorm a list of emotions (or use the list generated during the charades game), and have the girls each create roboticons that express one or two of those emotions. Make sure the girls post their emotions on the messenger software, so that anyone in the group can access them. Then, split the group up into pairs and distribute the stories (or have the girls write their own). Each girl will provide her partner with the necessary words/emotions/actions to fill in the Mad Lib, and then the partner will play the story for her, including her words and the roboticons from the messenger. (Remember, an important part of this game is that the person providing the words/emotions/actions has not yet seen the story.)
- Mad Libs can also be played as an individual activity. Girls can create roboticons for different emotions outside of class time, and write their own stories if they wish. Follow up during the next workshop session by inviting students to ‘play Mad Libs’ (play the roboticons during the stories) at the session.



Additional Materials: Sample Mad Libs

Documenting Designs and Reflection

(10 minutes)

Take pictures of each of the robots. If you have access to a scanner, you can scan the girls’ design sketches or you can take photos of the sketches for the web page.

Have the girls write in their blogs for a few minutes. Ask them to answer the following questions. In addition, they can also write about other aspects of their work today.

Reflection questions:

- What changes did you make to your robot today?
- Why did you make those changes? Were any of these changes the result of something you learned while using your robot? If so, what?
- What was the biggest challenge you faced in redesigning your robot today and how did you solve it?
- Based on your experience playing Mad Libs, do you think your robot is more expressive now than it was before? How?



Handout: Reflection 3

Individual Activity 3: Robot Meets World

Individual activities are designed for students to complete on their own, either at home or during free time at the workshop location. Students should expect to spend roughly 15-30 minutes on this activity.

The purpose of this individual activity is to give the students a chance to ‘show off’ their robot to someone outside of the workshop.

- First, identify someone (such as a family member, friend, or teacher) who you think would enjoy meeting your robot.
- Talk to them about how you can use a robot to express yourself, and demonstrate how your robot can express at least 3 different emotions.
- Identify at least 3 technical parts on your robot (e.g., servo, motor, LED) and explain what each one does.



Handout: Individual Activity: Robot Meets World

Individual Activity Follow-Up

(5-10 minutes)

In-class discussion questions for the ‘Robot Meets World’ activity:

- Who did you teach about your robot?
- What did you teach them? What emotions did you demonstrate?
- How did they respond?
- Had they ever seen a robot like yours before?

Making Changes 2: Design Reflection and Brainstorming

(15 minutes)

Remind students of the goal of being able to communicate and express feelings with the robot.

- Direct students to read the reflections they did during the last session, particularly their answer to the question, “*Based on your experience playing Mad Libs, do you think your robot is more expressive now than it was before?*” Encourage them to think about their answer to this question, and also to reflect on the reactions they received during the Robot Meets World activity. Then ask them to think about this: *What changes do you want to make to your robot as a result of those activities? For example, are there other things you would like the robot to be able to do or express?* Encourage students to draw some of their ideas about how the robot can be more expressive.
- Remind students that they have additional materials to work with while expanding their robots (2 additional servos, 4 additional LEDs, 2 vibration motors, 1 DC motor, and 1 speaker available to expand your robot).

Remind students of the building handout from the previous session.

Building

(45-60 minutes)

After reflecting and brainstorming, students will be ready to make changes to their robots.

Prepare for Show and Tell

(15 minutes)

Fifteen minutes before the group show and tell is to begin, let students know they should begin preparing. This may mean putting the finishing touches on their robot, getting the part of the robot they are currently building to work, or writing a program to show off their cool new expressive features.

Group Activity: Show and Tell and Feedback

(20 minutes)

The goal of Show and Tell is to give each girl a chance to show off her robot to the rest of the group, and to receive feedback on her robot from the rest of the group.

If you have a small group of girls (6 or fewer), go around the room and have each girl present her robot. If you have a large group, you may want to break the group up in to small groups of 4 students each, and have students do the show and tell within their small groups.

Before beginning the show and tell and feedback session, you may want to review the Feedback Guidelines handout from Part 2.

Questions for the group to think about when seeing each robot:

- What feeling or impression do you get from observing or interacting with this robot?
- What is the most interesting or creative thing about this robot?
- Do you have any suggestions for how she can change her robot to make it even more expressive?

Allow 15 minutes for show and tell followed by a group discussion.

Group discussion questions:

- Have your robots become more similar to each other or more different from each other since the first day? In what ways?
- Are there any trends or similarities in how people used particular components for expression? Did anyone come up with radically different expressive uses for any components?

Documenting Designs and Reflection

(10 minutes)

Take pictures of each of the robots. If you have access to a scanner, you can scan the girls' design sketches or you can take photos of the sketches.

Have the girls write in their blogs for a few minutes. Ask them to answer the following questions. In addition, they can also write about other aspects of their work today.

Reflection questions:

- What changes did you make to your robot today?
- Why did you make those changes? Were any of these changes the result of something you learned while using your robot? If so, what?
- Based on the feedback you got at the Show and Tell, do you want to make additional changes to your robot? If so, what?
- Have your ideas about what you want your robot to do or to be changed since you first built it?



Handout: Reflection 4

Individual Activity 4: Pull the Chain

Pull the chain is a storytelling game that can be run as either an individual or a group activity.

- To play as an individual game, decide who will begin the story. That girl should write the opening line of a story and make an expression or sequence to accompany it. Pass your story and roboticon to the next person in the group using private messaging (in your message, remember to include a list of each girl who has already contributed, so everyone has a turn to contribute). Each person should add another line to the story and another roboticon until each person in the group has contributed to the story. Share the final story on the public messaging board for everyone to read and play on their robots.

- Pull the chain can also be played as a group game within the workshop. To play as a group game, have *each* student write the opening line of her own story and make an expression or program to accompany it. Students then pass their stories and programs on to the next student (using private messaging), and that student adds another line and program to the story until each person in the group has contributed to each story. Share the final stories on the public messaging board for everyone to read and play on their robots.



Handout: Individual Activity 4: Pull the Chain

Pull the Chain Follow-Up

(5-10 minutes)

If this activity is done as an individual activity, share the whole story in class and let the students watch the story as it is expressed on a few different robots. If the activity is done as a group activity, choose one or two stories to share with the class.

Discussion questions:

- Think about how the story looked on each of the robots. Did the story look different on these different robots?
- If so, how was it different?
- Why do you think it looked so different?
- What does this tell us about the importance of robot form for expressing emotions?

Introduction to Sensors

(15-20 minutes)

Show students the sensors that are available to use with the Hummingbird (light and distance sensors). Demonstrate how to attach sensors to the Hummingbird and how to read their values from the Hummingbird tab in the software and use the Conditions in the Express-o-Matic tab.

It may be helpful to bring materials that the girls can use to explore the sensors. For example, flashlights can be helpful when learning about light sensors.



Handout: How to Connect Sensors to the Hummingbird

Introduction to Play

(5 minutes)

The group will create a play based on a fairy tale. The play will involve robotic actors (humans can be actors too). The girls will also create robotic props or set pieces using sensors.

We recommend splitting into groups of 3-6 girls each (3-4 is ideal per group), and having each group work on their own play. Within each group, girls can perform different roles (playwrights, robot directors, human actors, sets and costumes, etc.).

Remind students that as well as making custom programs for the play, they can reuse emotions programmed for previous activities. They will tell the play to parents and families at the end of the workshop.

Begin this activity by sharing the fairytales with students. Ask them to select one of the fairytales and start thinking about the play (they can adapt the fairytale for the play if they want to). The next step is to figure out what (if any) changes they will need to make to their robots for the play (i.e., building new components, developing new programs). Each student should incorporate a sensor into the technology. They can do this by adding a sensor to their robot, or creating a prop or set piece that incorporates a sensor.



Handout: Fairytales for play. (Copy from book of fairytales.)

Planning for Play (30 minutes)

After students are introduced to the idea of the play (above), ask them to begin planning the new robots and/or props they will need for the play. Ideally, they will do this planning on paper. Students can make lists of the elements they will need and begin sketching out sets and props. By the end of this planning session, students should know how they are going to modify their existing robots, and what they still need to build to complete the play.

Design/Re-design and Building for Play (50 minutes)

During this time, students will actively design, re-design, and build elements they will need for the play.

Documenting Designs and Reflection (10 minutes)

Ask the girls to take pictures of their new or modified designs and reflect on the following questions on their websites:

- Have you made any changes to your robot today? If so, what changes did you make and why?
- Are you building new elements for the play? If so, what?
- How are you using sensors?



Handout: Reflection 5

Individual activity 5: Silly Messages

This is the last individual activity of the workshop. Invite the students to send a silly roboticon to the friend of their choice.



Handout: Individual Activity 5: Silly Messages

Silly Messages Follow-up

If the girls want to, they can share their silly messages with the group.

Continue Working on Play

(75 minutes)

Students will continue working on their play, including narration and robotic elements. By the end of this session, students should be finished building and have their final play props in place. Students should practice the play at least once before the end of the session.

Group Activity: Robot Makeover

(30 minutes)

This activity doubles as an imbedded evaluation metric, allowing instructors and researchers to assess if individual students can (a) construct a simple circuit, (b) attach components to a microcontroller, (c) demonstrate the use of hand tools, and (d) write robot programs.

The goal here is to have the girls perform a ‘robot makeover’, where the girls are given a robot that is only capable of expressing one type of emotion. Their goal is to broaden the range of expressions possible for the robot.

To complete this task, students will need to:

- Construct a circuit that lights an LED when a switch is flipped.
- Attach servos, vibration motors, motors, LEDs, and sensors to the Hummingbird.
- Create the missing blocks in a programming puzzle.

Each student should solve the puzzle individually, but her time can be added to her team time.

NOTE: This activity is subject to change as we observe the progress of the workshop and the students. Researchers will help run this activity.

Survey

(15 minutes)

Allow time to complete the post workshop survey if not already completed. The survey will be provided by the researchers.

Group Activity: Play Presentation

(30-60 minutes)

Present the completed play(s) to parents, family, and friends.

You can also invite family members and friends to view demonstrations of the robots and view the webpages/posters.

Handouts and Additional Materials

In this appendix you will find the following handouts and additional materials that you may need during Part 3 of the Arts & Bots workshop:

- **Design Iterations of a Robotic Desk Lamp** – Discussion guide for teachers to accompany the handout below. Includes discussion points for each desk lamp image in the student handout as well as additional background information.
- **Design Iterations of a Robotic Desk Lamp** – Handout for students showing the multiple iterations in the design of a professional expressive robot.
- **Reflection 2**
- **Redesign Instructions** – Instructions for brainstorming and redesigning the robots.
- **Sample Mad Libs** – Sample Mad Libs to be read by the instructor as an introduction to the Mad Libs game.
- **Reflection 3**
- **Individual Activity 3: Robot Meets World**
- **Reflection 4**
- **Individual Activity 4: Pull the Chain**
- **How to Connect Sensors to the Hummingbird**
- **Reflection 5**
- **Individual Activity 5: Silly Messages**

Design Iterations of a Robotic Desk Lamp

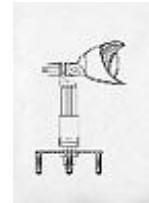
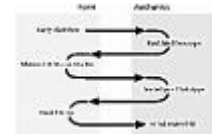
Discussion Guide for Teachers

Robot design and images by Guy Hoffman, MIT Media Lab

The Robotic Desk Lamp was designed to study how a robot could evoke a personal relationship with people through abstract gestures and nonverbal behavior. The robot's movements can be programmed and the lamp can shine in a variety of colors such as purple, red, blue, and white. Because of the Desk Lamp's expressiveness, it was part of a theater performance with human actors.

The designer of the Robotic Desk Lamp went through multiple design iterations. With each round of iteration he refined the form of the robot and its mechanical structure.

1. Early Sketches – The designer experiments with different forms in his early sketches.



2. First Iris Prototype – The first prototype is only a portion of the whole robot.



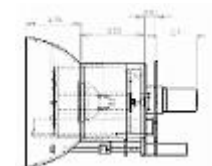
3. Material and Shape Studies – The sketches become more concrete and specific. The designer experiments with different materials and different shapes for the head of the lamp. Note the four different head shapes in the computer sketch.



4. Iris and Lens Prototype – The next prototype is more elaborate.



5. Final Design – The final design sketches are very detailed and exact.



6. Final Assembly – The final robot is the result of many iterations of design.





Additional Information

From <http://web.media.mit.edu/~guy/aur/>:

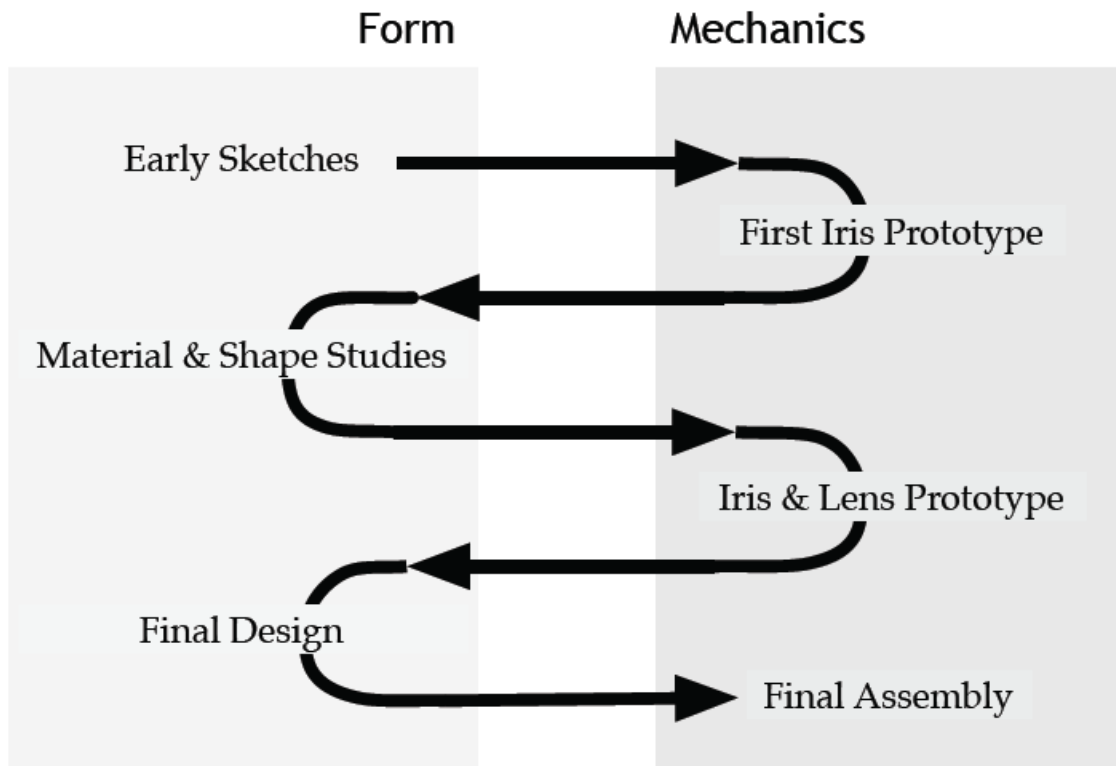
“AUR is a robotic desk lamp, a collaborative lighting assistant. It serves as a non-anthropomorphic robotic platform as part of Guy Hoffman’s Ph.D thesis on human-robot fluency, embodiment, and nonverbal behavior.

“The lamp's design was conceived around an existing 5-DoF robotic arm, and is aimed to evoke a personal relationship with the human partner without resorting to creature-like features such as eyes, limbs, or a mouth. By retaining the lamp's "objectness", Hoffman hopes to explore the relationship that can be maintained between a human and an object through abstract gestures and nonverbal behavior alone.

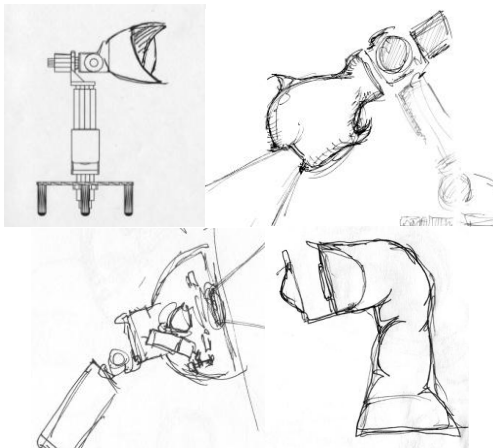
“The lamp is animated using a custom pipeline enabling the dynamic control of behaviors authored - in part - in a 3d animation system, and has perform in a unique human-robot joint theater performance in May 2007.”

Design Iterations of a Robotic Desk Lamp

Robot design and images by Guy Hoffman, MIT Media Lab



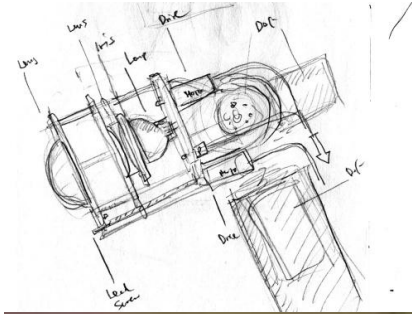
1 Early Sketches



2 First Iris Prototype



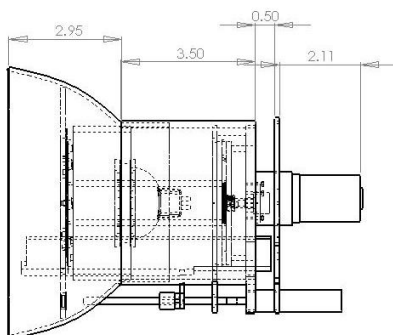
3 Material and Shape Studies



4 Iris and Lens Prototype



5 Final Design



6 Final Assembly



Reflection 2

Reflection Question:

- How can you make your robot more expressive?

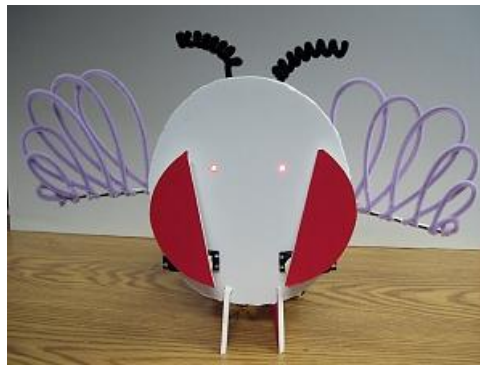


Today, you can make your robot more expressive by adding additional servos, LEDs, motors and sound capabilities. You will have at least 3 sessions to work on your additions and robot customization so you can test your robot between sessions and make improvements at the next session.

The additional components you can use are:

- 2 additional servos
- 4 additional LEDs
- 2 vibration motors
- 1 motor
- 1 speaker

Try to integrate as many components as you can. By the end of the final design session, you should have used at least 3 of the 5 different types of components listed.

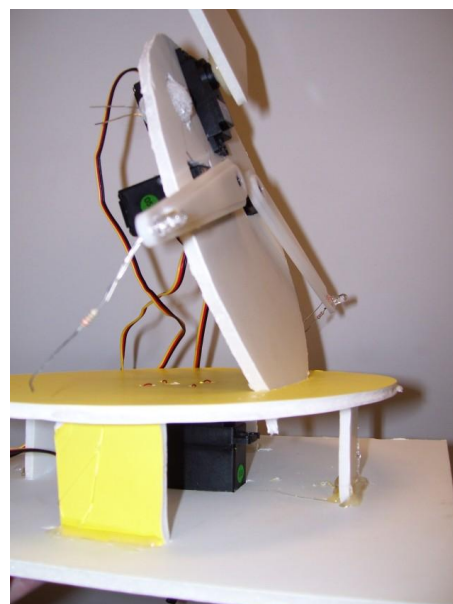
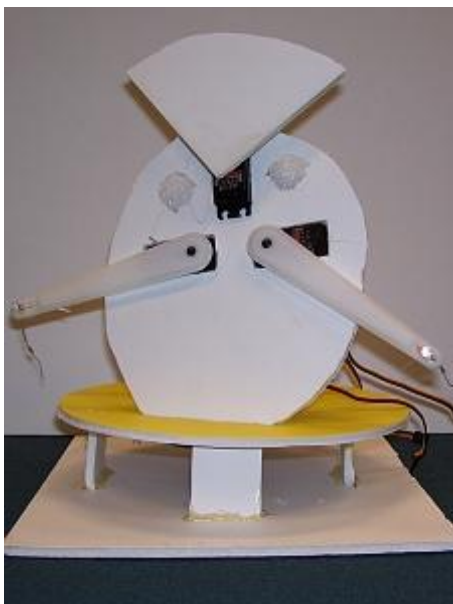


For today, try to integrate at least one new component into your robot.

Adding these additional components will require you to re-design parts of your robot. This is all part of the design process. Now you have a chance to improve your robot and make it exactly the way you want it!

Before you begin building spend 20 minutes brainstorming about how you will add new components to your robot to make it more expressive. Draw some sketches in your notebook and label the new components.





Sample Mad Libs

Siblings

My little brother Jamie makes me so (emotion roboticon) ! He was in my (place) earlier this morning and he messed everything up - I was so (emotion roboticon) at him. He pulled all my (adjective) (plural noun) off the shelf and threw them around the room, and then he broke one of my (plural noun) by knocking it off my dresser. I was so (emotion roboticon) that I (past tense verb) at him, but he wouldn't stop. Then my dad came in - boy was he (emotion roboticon). He picked Jamie up and carried him out of the room. This made Jamie (emotion roboticon), and I felt really (emotion roboticon). But then I brought him a (noun), and that made him (emotion roboticon).

Spiders

There is a (noun) in my bedroom. I really hate (repeat first noun)s. This one is (adjective) – almost the size of my thumb. Ever since I could remember, I've been (emotion roboticon) of (repeat first noun)s. I once found a (adjective) one in my closet, sitting on top of my favorite (noun) – yuk! Now this (adjective) (repeat first noun) is in my bedroom, and I don't know what to do about it.

Maybe I'll try (verb)ing to scare it off. Okay, here goes... amazing, it didn't even move. What if I (verb) at it? Again, nothing. Now I'm (emotion roboticon) – what's going on here? Let me take a closer look...

No wonder it didn't move. It's not a (repeat first noun) at all – it just a piece of (noun). I'm so (emotion roboticon) !

Reflection 3

Reflection Questions:

- What changes did you make to your robot today?
- Why did you make those changes? Were any of these changes the result of something you learned while using your robot? If so, what?
- What was the biggest challenge you faced in redesigning your robot today and how did you solve it?
- Based on your experience playing Mad Libs, do you think your robot is more expressive now than it was before? How?

Individual Activity 3: Robot Meets World

Introduce your robot to someone who is not part of the Arts & Bots workshop.

- First, identify someone (such as a family member, friend, or teacher) who you think would enjoy meeting your robot.
- Talk to them about how you can use a robot to express yourself, and demonstrate how your robot can express at least 3 different emotions.
- Identify at least 3 technical parts on your robot (e.g., servo, motor, LED) and explain what each one does.

Reflection 4

Reflection Questions:

- What changes did you make to your robot today?
- Why did you make those changes? Were any of these changes the result of something you learned while using your robot? If so, what?
- Based on the feedback you got at the Show and Tell, do you want to make additional changes to your robot? If so, what?
- Have your ideas about what you want your robot to do or to be changed since you first built it?

Individual Activity 4: Pull the Chain

Pull the chain is a storytelling game. Someone from the group should be selected to start the story.

Write the opening line of a story and make an expression or sequence to accompany it. Pass your story and sequence to the next person in the group using private messaging (in your message, remember to include a list of each girl who has already contributed, so everyone has a turn to contribute).

Each person should add another line to the story and another sequence until each person in the group has contributed to the story. Share the final story on the public messaging board for everyone to read and play on their robots.

How to Connect Sensors to the Hummingbird

Connecting Sensors

- + Make sure the Hummingbird is turned off before attaching the sensors!
- + Locate the ports on the Hummingbird labeled “SENSOR1” and “SENSOR2”.
- + Each sensor has three wires, a red power wire, a black ground wire, and a yellow signal wire.
- + Insert the red wire into the power port (labeled ‘+’).
- + Insert the black wire into the ground port (labeled ‘-’).
- + Insert the yellow wire into the signal port (labeled ‘S’).
- + Gently tug the three wires to make sure they stay in place.



IR range-finder (distance sensor)

Detects objects between 4” and 30” away

The range sensor is composed of an emitter and detector module and some sophisticated circuitry. The emitter operates by emitting a narrow beam of infrared light. If this beam hits an object, the light will reflect and some of it will be picked up by the detector. The circuitry at the core of the sensor determines how long it took for the light to travel to the object and back to the detector – a pretty impressive trick, given that light travels 1 billion feet in one second! The circuitry converts this measured distance to a voltage, which is read in by the Hummingbird and can be used by our programs.



Photoresistor (light sensor)

Measures light levels

The light sensor is based on a photoresistor, which is a device that changes electrical resistance based on the amount of light hitting the surface – as more light hits the surface, the resistance decreases. We have connected the photoresistor in series with a fixed resistance resistor, and are applying a fixed voltage across the circuit – as the resistance of the photoresistor changes, the voltage at the junction between the photoresistor and resistor also varies. The Hummingbird sensor ports can measure this voltage, and convert it into a value that we can use in our programs.

Reflection 5

Reflecting on and documenting our experiences as we design something helps us to improve our design. Your reflections today will also be a source of material for your web page or poster.

Reflection Questions:

- Have you made any changes to your robot today? If so, what changes did you make and why?
- Are you building new elements for the play? If so, what?
- How are you using sensors?

Individual Activity 5: Silly Messages

This is the last individual activity of the workshop.
Send a silly roboticon to the friend of your choice.

Appendix – Expansion activities and supporting materials.

- Group Activities
 - Individual Activities
 - Pictures of Arts & Bots Materials
 - Software Installation Instructions
-

Group Activities**Furby Take Apart**

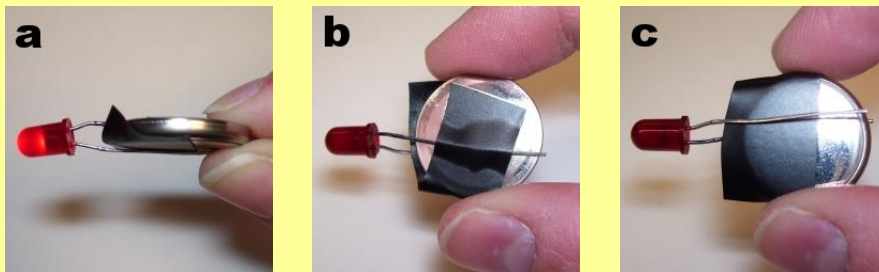
Take apart a Furby or other robotic or electronic toy (ex: singing, dancing hamster). This activity can be done as a demo or in small groups.

LED Clothing and Jewelry

Girls create clothing and jewelry with integrated LEDs. Some ideas for materials that can include LEDs (purchase or have students bring in):

- Gloves
- Stretchy wrist bands
- Necklace strands
- Foam or paper masks

This activity can start with a quick (3-5 minute) presentation of all the available materials. During this presentation, you should demonstrate how an LED (without any wires attached) and a coin cell battery can be put together with electrical tape to make a glowing orb. Girls should be urged to think for a couple of minutes before starting to figure out what they want to make.



Students can tape both pins of the LED to the battery to create a light that is always on. To create an LED flashlight that lights up when pinched (a), tape one side of the LED to the battery (b) and use electrical tape to partially insulate the other side of the battery (c).

LEDs without wires attached are fairly inexpensive, as are coin cell batteries. The materials used for this activity can be considered one-time use, if you want to allow the girls to take their creations home.

LED Light Show

Create an LED light show set to a piece of music.

Name that Tune

This activity fits after the “Make a Noise” activity and can be used when introducing the programming software. After creating something that can make noise, program the object to sound out the rhythm of a familiar song or nursery rhyme. See if others can guess what the song is.

Provide a list of songs for students to choose from in order to make the guessing easier. It can be quite difficult to pick out the tune based on rhythm alone.

Robot Dance Party Expansion

Lead in to Robot Dance Party activity with a discussion about how movies use music to help convey emotions. Shrek has some good examples. Have students name other example movie scenes. Talk about how you feel when you hear certain pieces of music. Do you feel proud or patriotic when you hear the national anthem or God Bless America? How do you feel when you hear Amazing Grace or Kumbaya?

Hokey Pokey

Program your robot to do the hokey pokey. If your robot doesn't have movable body parts for all parts of the song, be creative!

Sensing Box

Make a box that responds when someone picks it up. Use this activity as an introduction to sensors.

Individual Activities

Emotions Warm Up

What emotions do you feel and communicate? Draw pictures to illustrate or represent the emotions. What do your pictures have in common with pictures drawn by the others in the group (colors, shapes, etc.)?

Invitation to the Robot

(If girls will be taking their robots home.) Write an invitation for the robot to come live at your home.

Prepare a Home

(If girls will be taking their robots home.) Find a place for your robot to live. Prepare a special home where your robot will live.

Robot Biography

Write a story or draw a cartoon about your robot's history or background. The story might explain the robot's appearance, name, or other interesting things about your robot.

My Roboticon Profile

Create a robot program that expresses your personality and identity. Share your program on the public messaging board. Take a video clip of the program running on your robot to include on your web page as sort of personal profile.

Cool Robot Video Clip

Create a program that shows off the expressiveness and uniqueness of your robot. The program may be accompanied by a back story. Create a video clip of your program to include on your website.

Arts & Bots Materials



Alligator clips



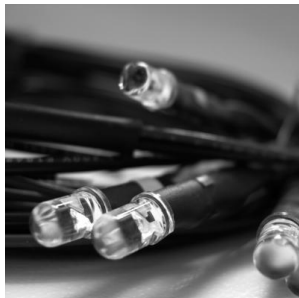
Battery pack



Switch



Coincell battery



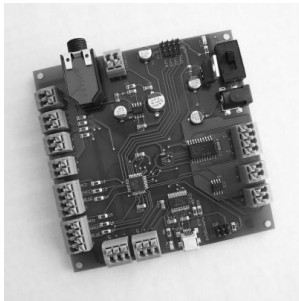
LED
Light emitting diode



Motor and wheel
Spin forward and back



Vibration motor
Makes vibrations



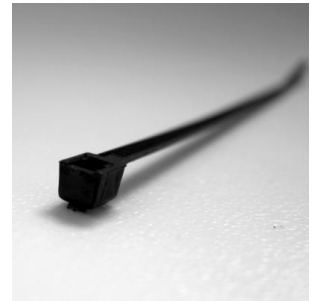
Hummingbird



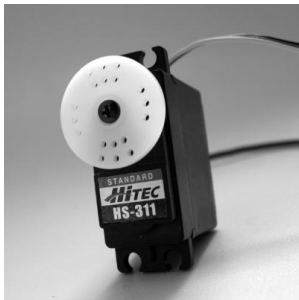
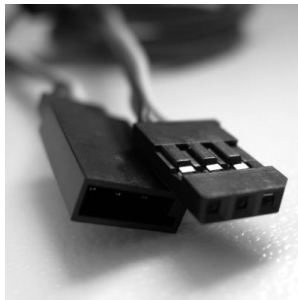
AC adapter



USB cable



Cable ties

Servo
Moves back and forth

Servo extender



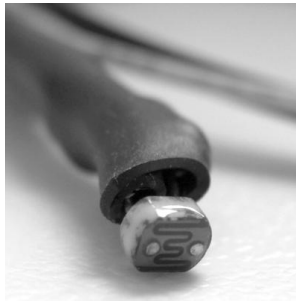
Speaker



Speaker cable



IR range-finder
Distance sensor



Photoresistor
Light sensor



Wire cutter



Wire stripper



Screwdriver

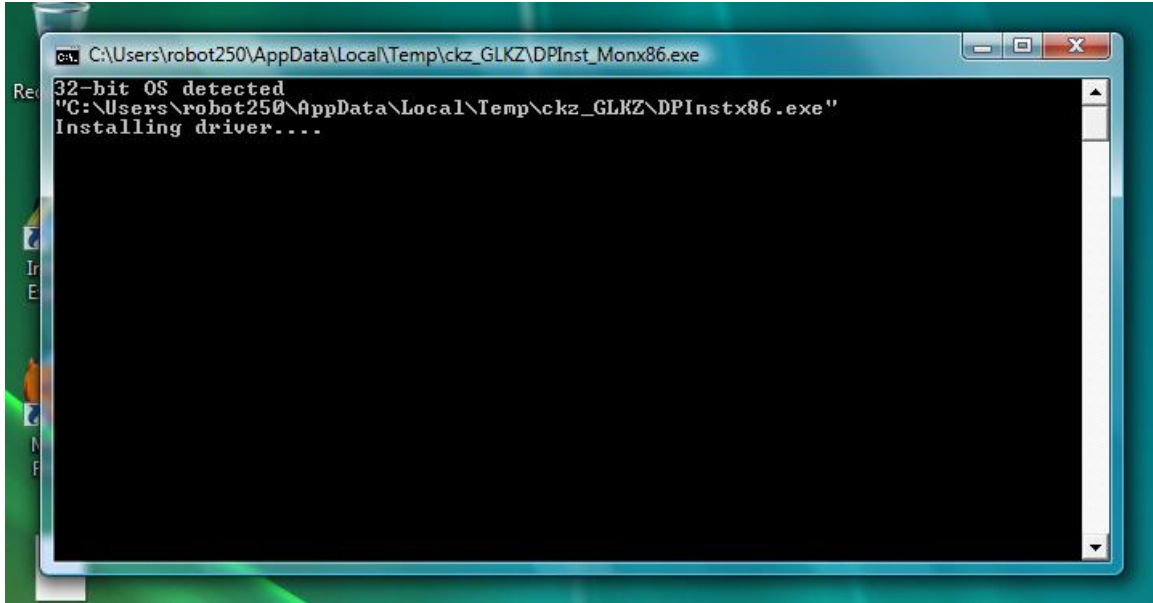
Software Installation Instructions

This document provides instructions for installing the various software packages necessary to run the Arts and Bots workshop. All of the software installation files have been included on the accompanying CD. You will need to install the following software **in order**:

1. Hummingbird Drivers (install using a computer administrator account)
2. VLC Media Player (install using a computer administrator account)
3. Java Runtime Environment (JRE) (install using a computer administrator account)
4. Arts and Bots (install under the students' account)

Installing the Hummingbird Drivers

1. Double click on the file labeled CDM 2.04.06.exe in the CD's Hummingbird Drivers folder. This will open an installation dialog that looks like the following:



Note that installation may take several minutes. Upon completion, the screen will disappear without warning - do not be alarmed if this happens, it means the software was installed successfully.

Installing the VLC Media player

1. To install the VLC media player, used for playing movie clips used during the workshop, double click on the file vlc-0.8.6h-win32.exe. This will bring up an installation program.
2. Click 'Next' through all of the menus, and then hit 'Finish' when you have completed installation.

Installing Java

1. To install Java, double click on the file in the Java JRE folder on your CD called “jre-6u6-windows-i586-p-s.exe
2. Click ‘Accept’ to the first question posed by the installation program. The program will then start installing Java, bringing up the following status window:

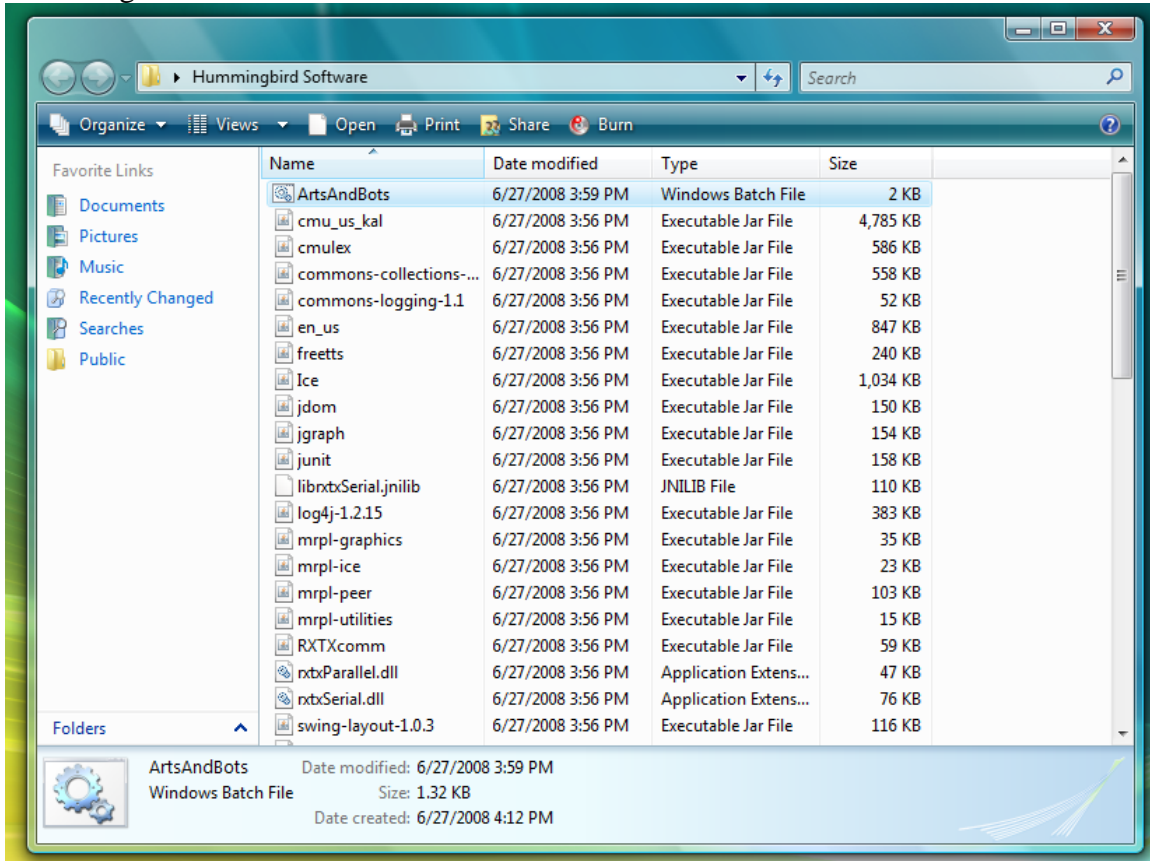


Installation may take 5-15 minutes. Once installation is complete, you will see the following window:



Arts and Bots Software Installation

1. Copy the entire 'Arts and Bots' folder from the CD to the computer desktop.
2. **Plug in and power on a Hummingbird.**
3. To run the Hummingbird Software, double click on the file 'ArtsAndBots.bat' in the Hummingbird Software:



Arts and Bots Software Instructions

This document is meant as a training and reference guide for using the Arts and Bots software. With the software you can make your robot make a single expression (like happy), make programs that cause the robot to move between expressions and react to sensors, and to share these programs and expressions with other people.

The Arts and Bots Software is divided into three sections:

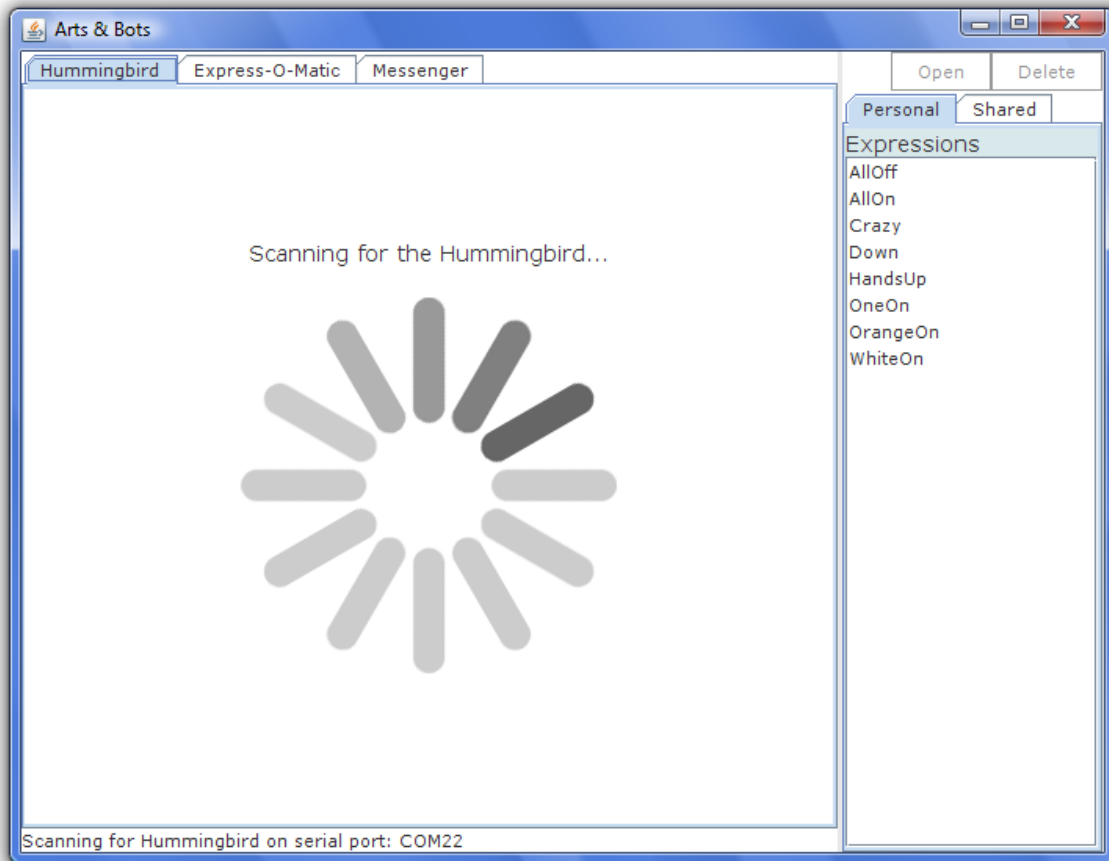
- The ‘Hummingbird’ allows you to directly control parts of your robot that are attached to the Hummingbird robot controller. You can use it move all the parts of your robot into a single **expression** (like arms up and LEDs green for happy), and then save that **expression**.
- The Express-O-Matic allows you to chain together any number of **expressions** into a **sequence** to make your robot move from one expression to another. You can also use sensors to have your robot react to things like motion and light.
- The Messenger allows you to share the **expressions** and **sequences** you make as **roboticons**. You can also use it to look at, play, and modify the **expressions** and **sequences** that your friends make.

So that’s the overview – now we’ll look at each section in more detail to give you an idea of how to use them. But before we do that, we have to start the software!

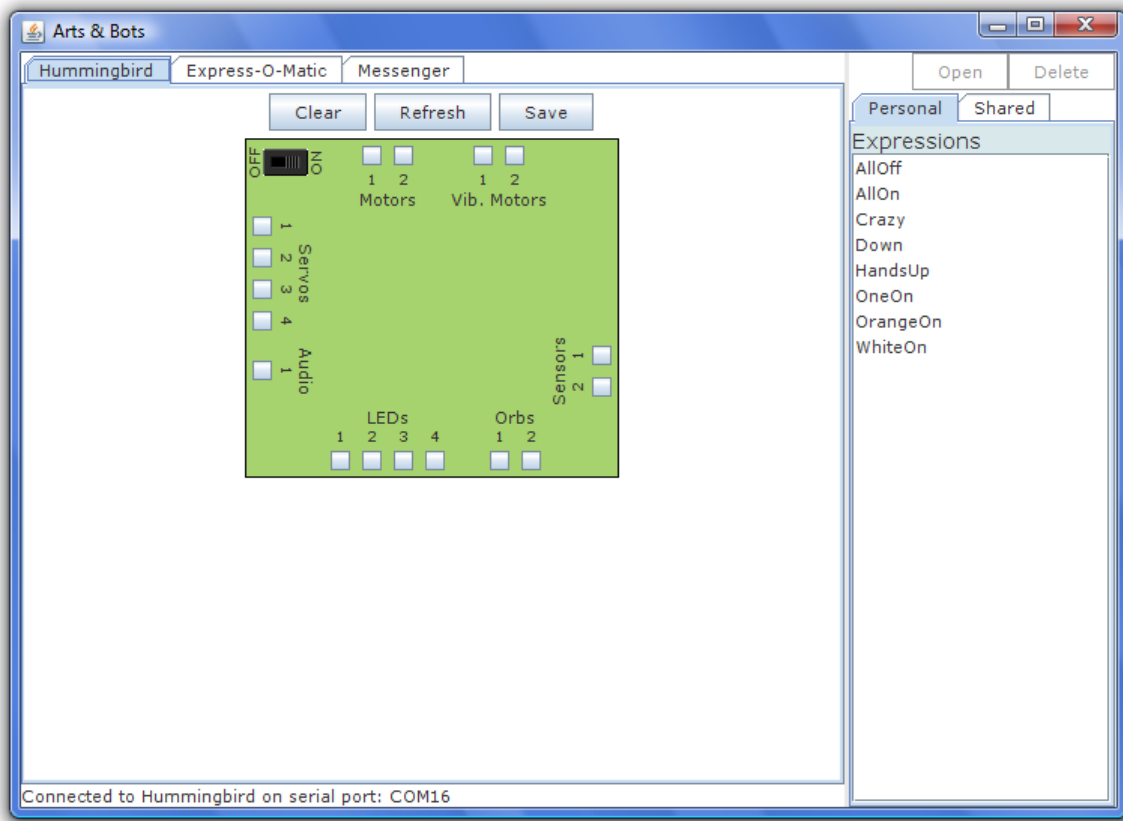
Starting the Arts and Bots software

Go to Arts and Bots software directory on your computer, and double click on 'ArtsAndBots' to launch the software:

The Arts and Bots software will launch and start looking for Hummingbird controllers attached to the computer:

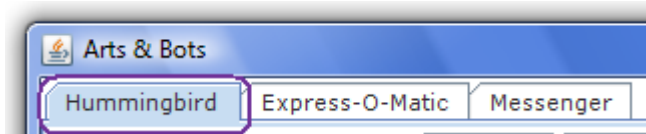


If you have not already done so, plug a Hummingbird into your computer using the USB cable, and power on the Hummingbird by connecting its power supply and turning it on. The software may continue to look for the Hummingbird for up to one minute; once it is found, it will look like this:

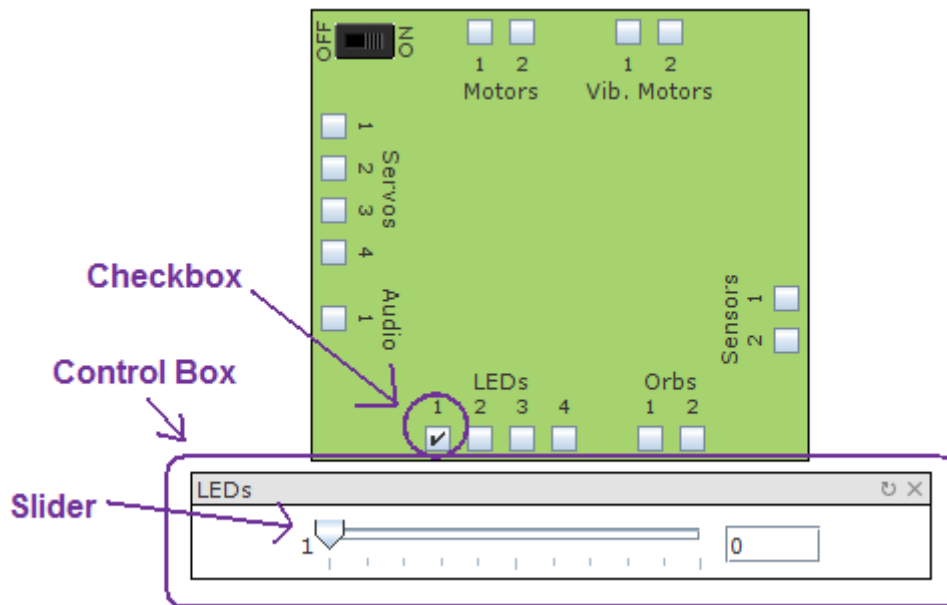


Congratulations, you are now set up to use the software!

Hummingbird



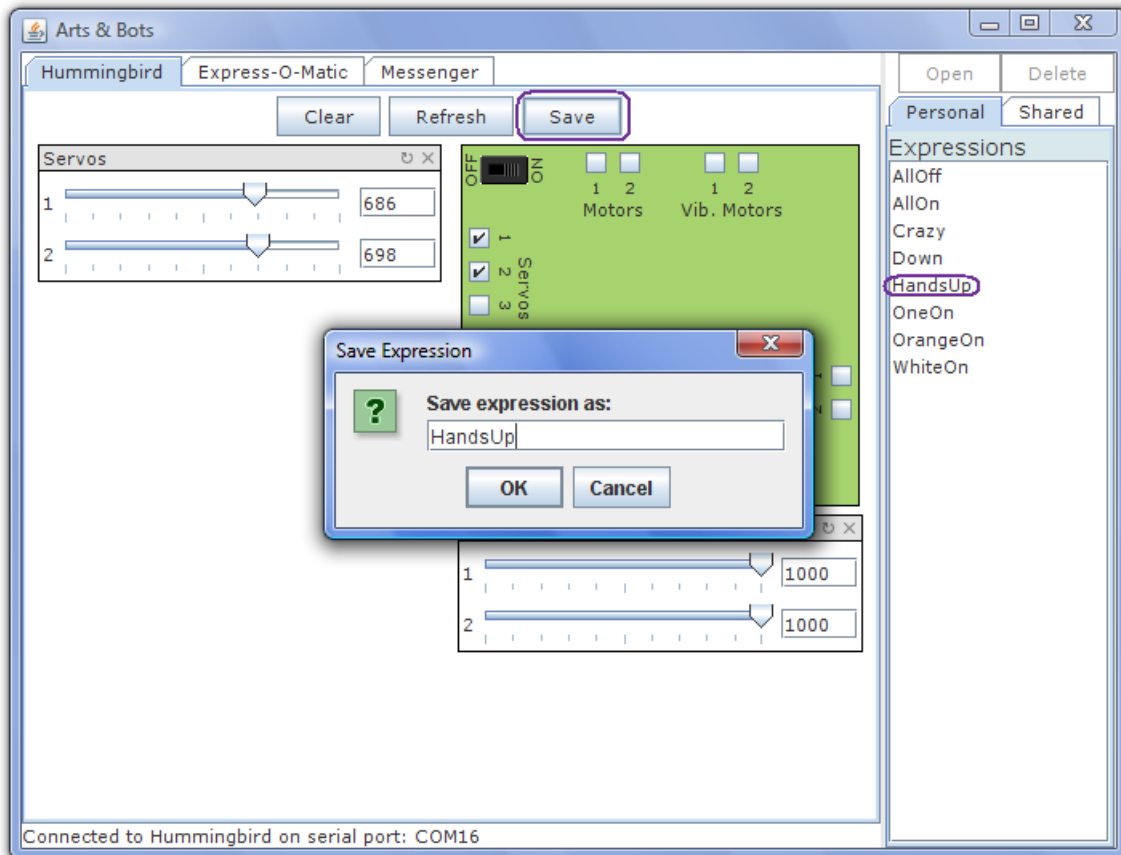
The Hummingbird Section of the software allows you to control components attached to the Hummingbird. To control a specific component, simply click on the checkbox for that component – for example, if I wanted to control LED 1, I would click on its checkbox:



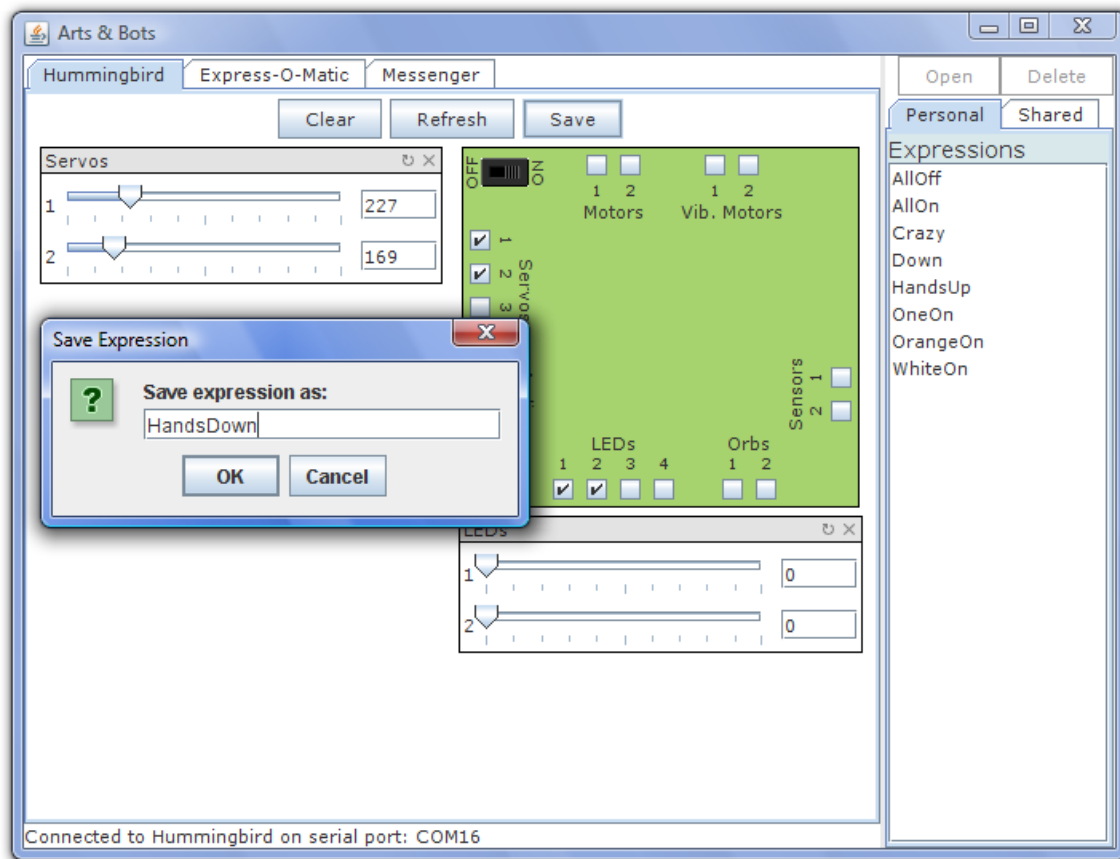
Once a checkbox is checked, a control box that allows you to control it will appear. Most control boxes are sliders – for the LED the slider controls how bright the LED is. Try the other checkboxes to see how they work.

A special note about the sensor boxes; they show you the current value of the sensor (a number from 0-1000). You can't change this because the sensor is changing it, and if no sensor is attached, the number it displays doesn't mean anything.

Once you're used to using the checkboxes, make your robot express by changing the right components to something that you like. For example, if I had a robot with two servo arms and two eyes, I could set the arms to point up and turn on the eyes. I would save that expression by hitting the 'Save' button, and then the software would ask me to name the expression. I called it 'HandsUp' and once it is saved, it shows up in the 'Personal' list of Expressions that I've made.

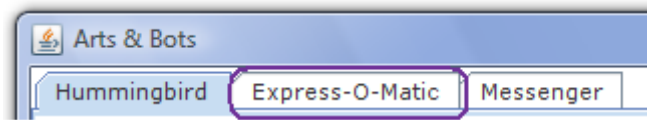


Just for fun, I'll make a second expression called HandsDown where the arms are down and the eyes are off:



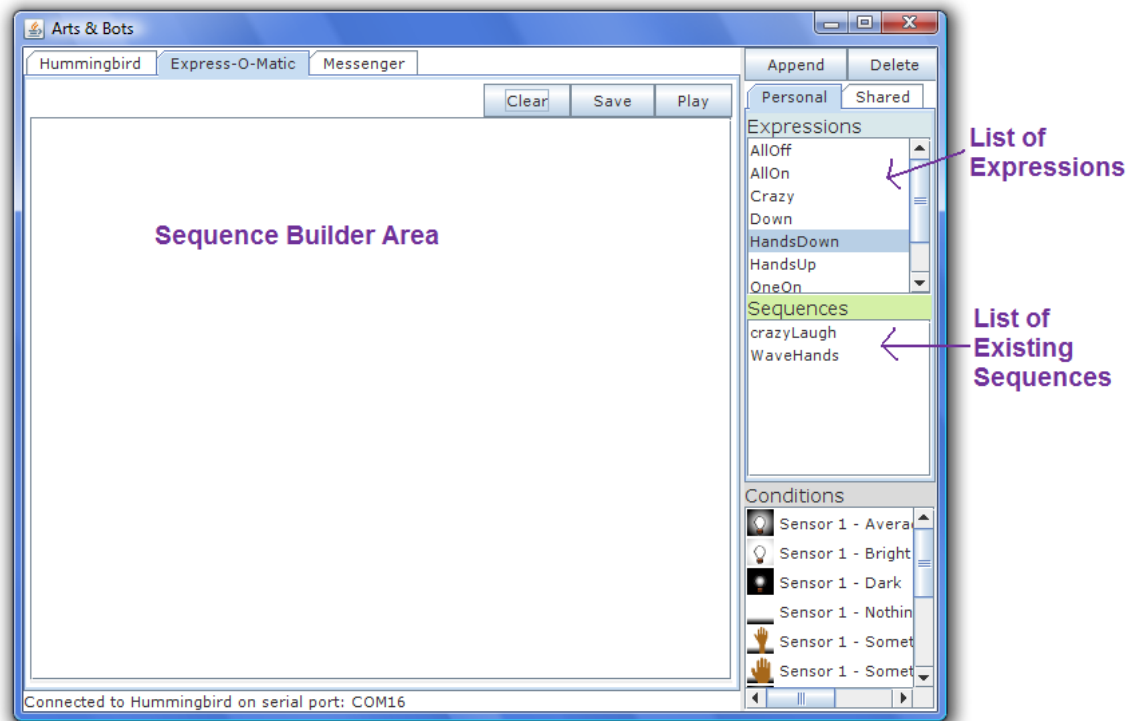
Now that I have both of those expressions, I have the necessary building blocks to write a program that moves the robot back and forth between the two – in my case, that would mean I could have the robot move its arms up and down and eyes on and off. To do that, I'll need to switch tabs to Express-O-Matic.

Express-O-Matic

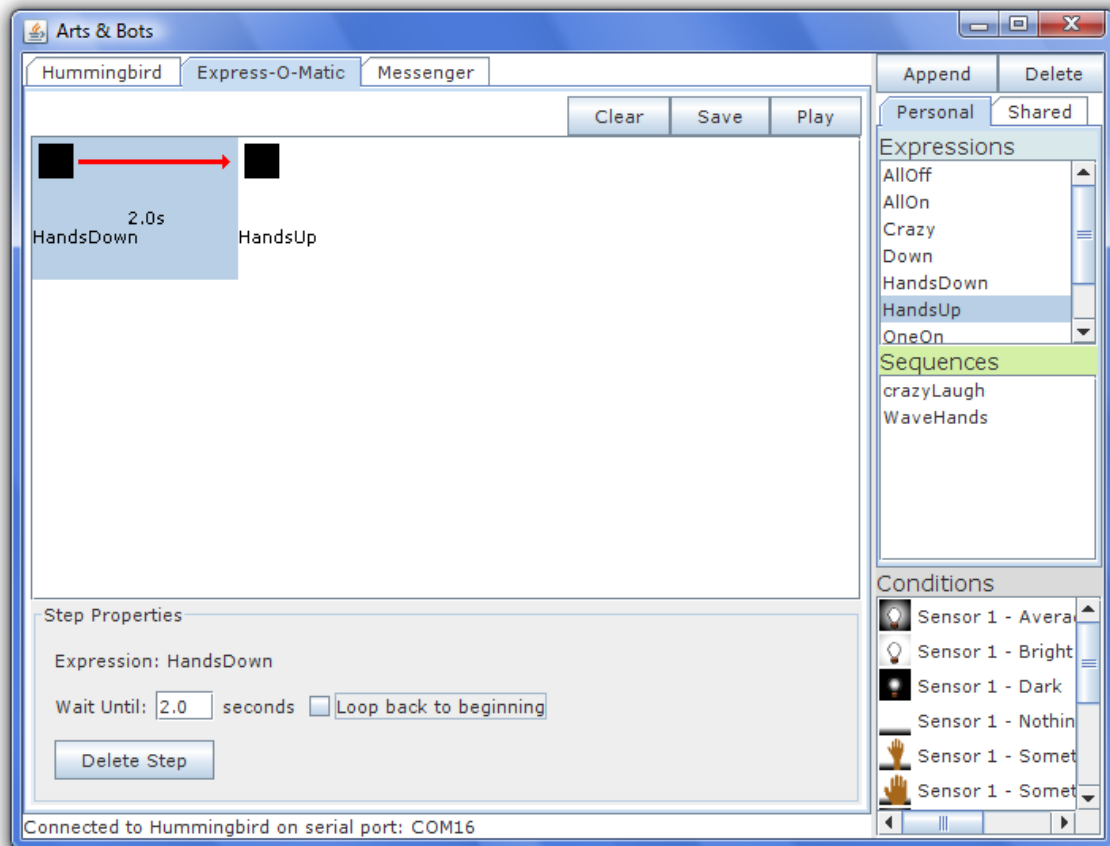


Click here to move to
Express-O-Matic

The Express-O-Matic allows you to chain Expressions together to build Sequences, allowing your robot to automatically change from one expression to another. To build a sequence, you need to click on an Expression in the Expression List, and drag it into the Sequence Builder:



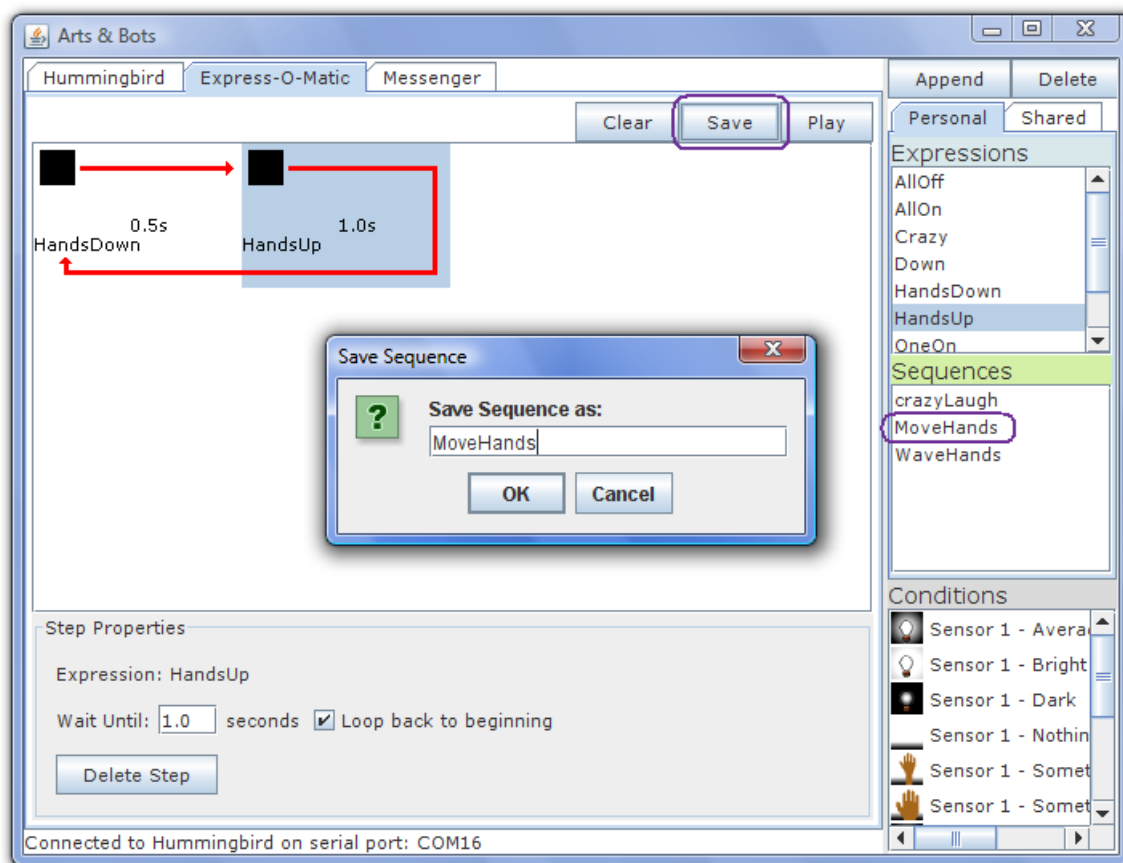
I'll do so with the HandsUp and HandsDown expressions I made previously:



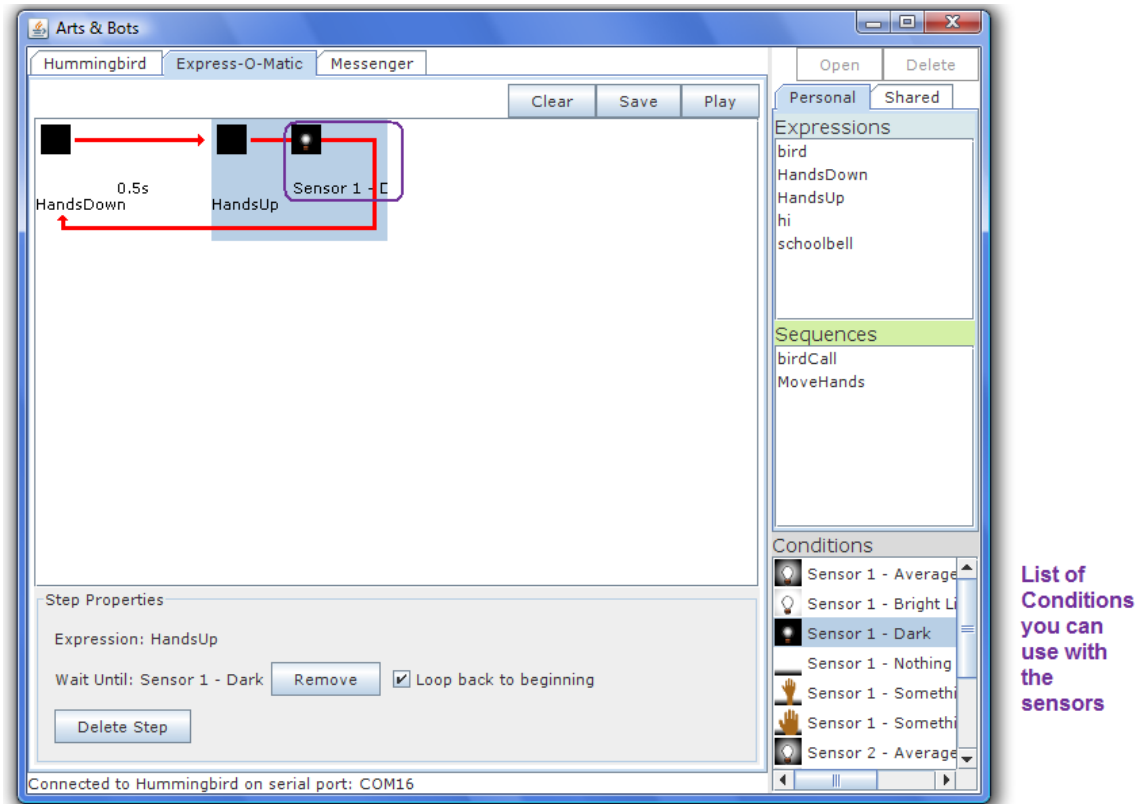
Now I can play these expressions by hitting the play button. It'll move the robot to the settings from the first expression, wait 2.0 seconds, and then move to the second expression. To make it more interesting, we can click on the checkbox marked 'Loop back to beginning'. Now if we hit play again, it the robot will move between the two expressions forever, waiting 2 seconds at each one. You can adjust how long it takes to move from one expression to the next by changing the 'Wait Until' at each step – you can make it real small (like 0.1 seconds), or real long (like 10 seconds).

Try modifying these things and playing the program to see how it looks. Note that your program needs to be stopped for you to change any part of it.

Once you have something you like, you can save it with the save button – it'll show up in the 'Sequences' list, and you can always reopen it by double clicking on it:

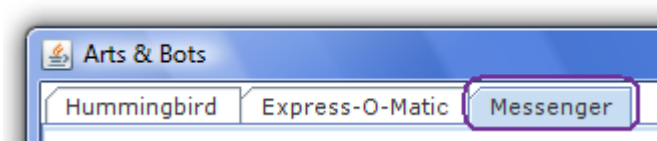


You can also use the sensors to make your robot wait until a sensor has been tripped. There are two kinds of sensors you can use – a light sensor, and a distance sensor. To use a sensor, drag a ‘Condition’ into one of the boxes – with my current sequence I’ll drag Sensor 1 – Dark into the ‘HandsUp’ box. Now the sequence will wait to put the arms down again until my light sensor sees very little light.

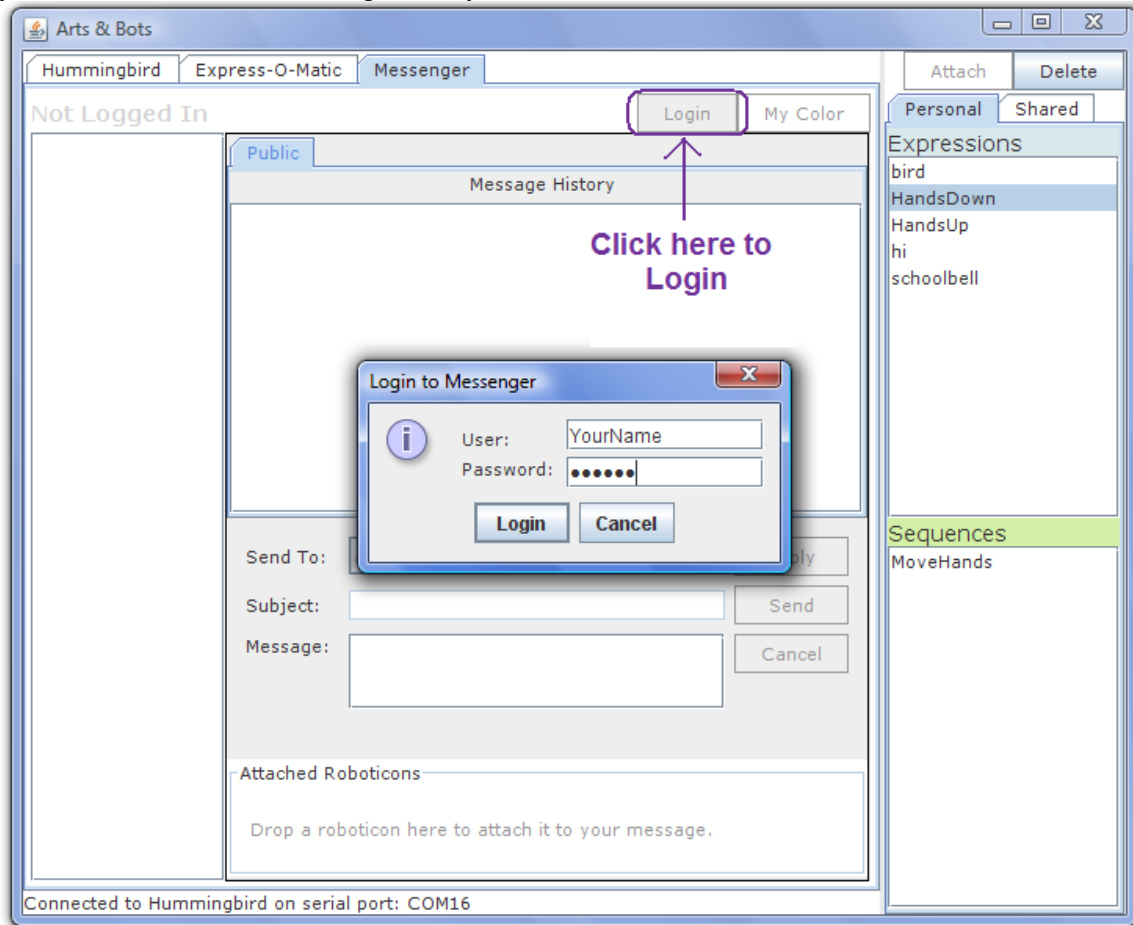


Now that you have a sequence that you like, you can share it with your friends, allowing them to play the sequence on their robots. To do so, you'll need to go to the Messenger screen.

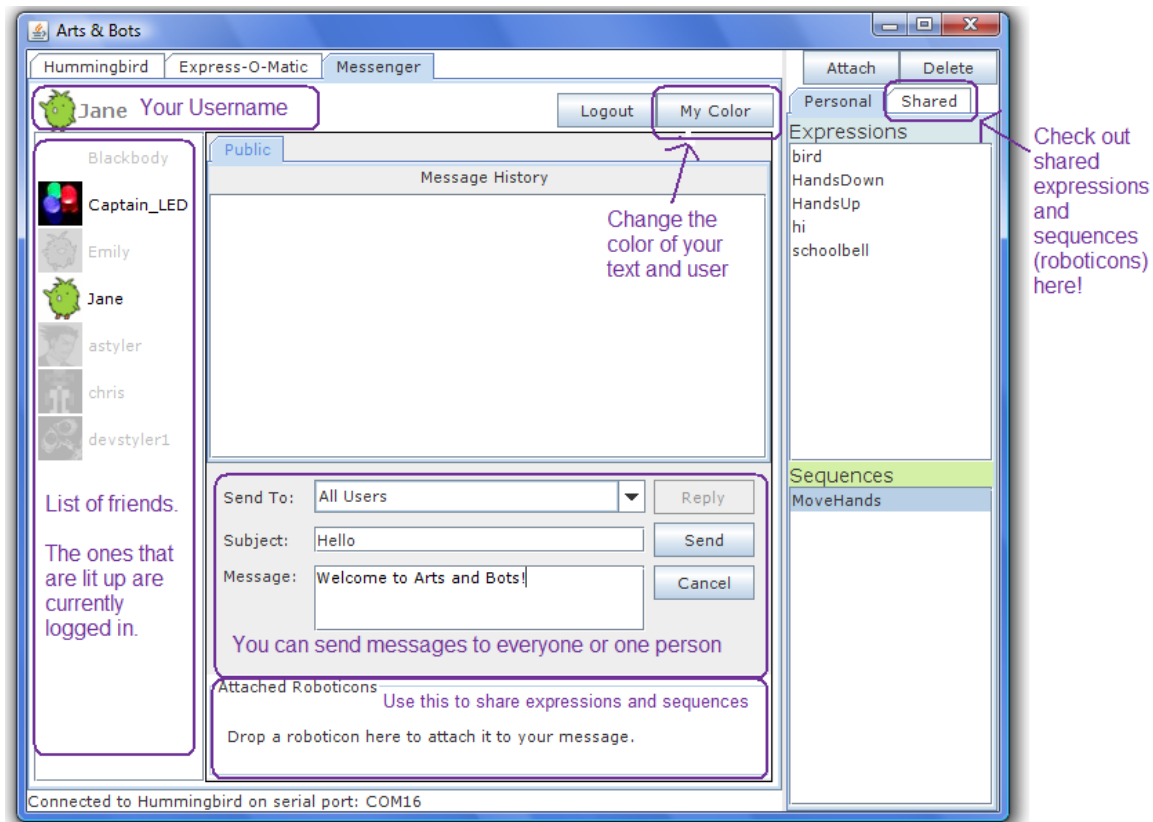
Messenger



The first thing you need to do is to login. Login with the user name and password that you created (or that was assigned to you).



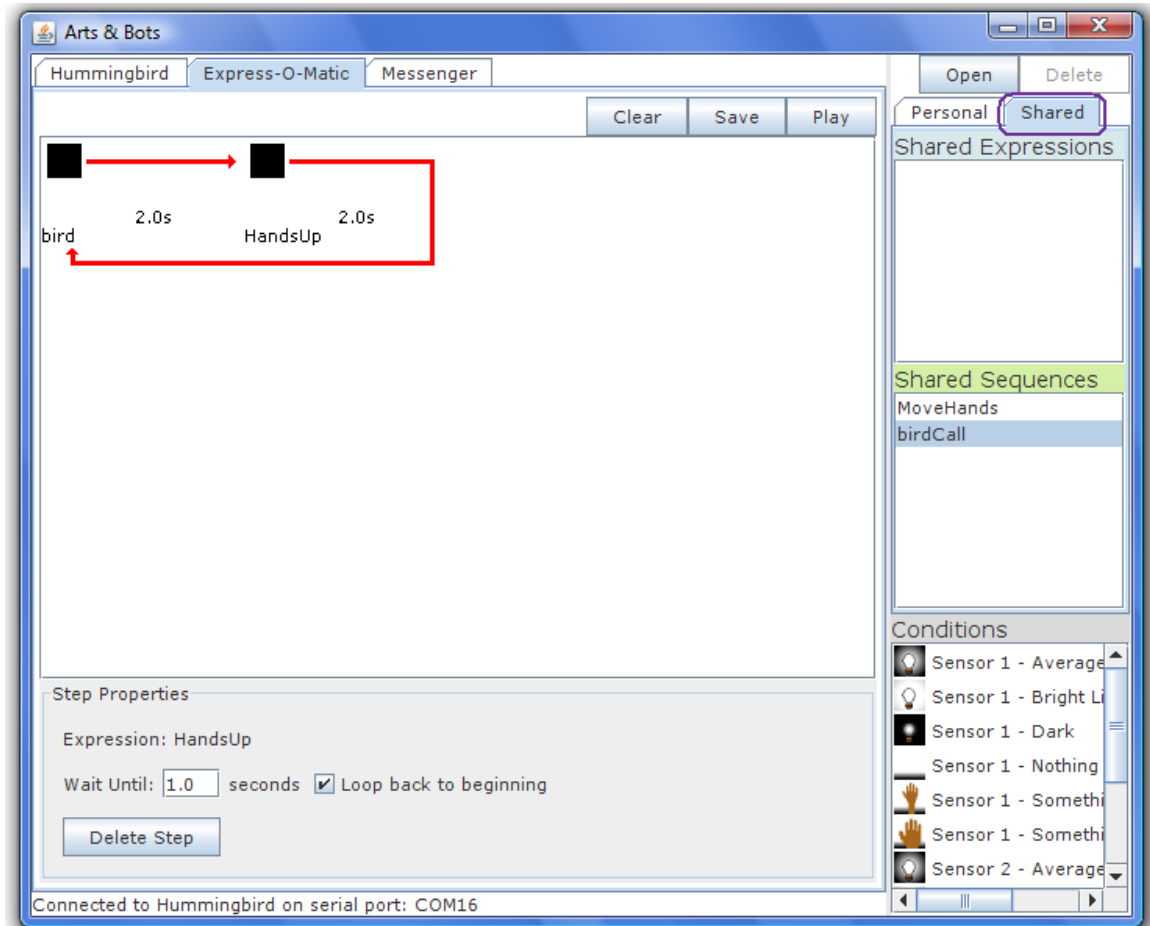
Once you've logged in, many things on the Messenger screen will become active. Your username will be displayed in the top left, above a list of all the people who you can talk to. User names that are lit are currently online – so in the following screen capture you can see that Captain_LED and Jane are online.



You can use the message sending box to send messages – use the 'Send To:' box to choose who to send it to. You can also reply to messages you've received.

Along with sending your messages, you can also attach Roboticons – these are Expressions or Sequences that you've created that you want to share. You'll share these with whoever you send the message to.

You can also check out Roboticons that have been shared with you – click on the ‘Shared’ tab to look at these. If you want to open them up, you can do so in the Express-O-Matic and Hummingbird sections of the software. For example, Captain_LED just shared the ‘birdCall’ expression with everyone, so now I can go to Express-O-Matic and open it up:



That’s all the instructions you’ll need to start using the Arts and Bots software. We hope you enjoy it!

A.1 Dispositional Goals

Top-level goal: **Engagement** The program will help girls see technology as interesting and deeply relevant to their lives, and help motivate their continued engagement and exploration of technology. There are a number of ways in which we sought to measure movement towards this goal:

- Students show interest in further workshops.
- Students express interest in continuing to use the robot after the workshop ends.
- Students persist in the face of technical difficulties.
- Students use the chat software, create complex programs, or make custom robot parts outside of workshop times.
- Finally, students integrate technology with their social life. For example, one girl made a Halloween robot that would say trick or treat.

Top-level goal: **Confidence** The program will help girls to develop confidence in their own ability to create with, modify, or troubleshoot the technology in their lives. This could be demonstrated by students in a number of ways:

- They are not afraid to take something apart. Error messages don't scare or cause them to freeze into inaction.
- Students exhibit a willingness to try before asking for help, but are willing to ask for help when necessary.

Top-level goal: **Multiple “right” answers** Students have incorporated the idea that there is no one right answer to solving many real-world problems. This should lead to increased efficacy in practical problem solving, as students are more willing to try a variety of solutions.

Top-level goal: **Creative use of technology** Students understand that there are many ways to use technology (including in situations which do not appear to be technological), and that some of these constitute creative ways to use and think about technology.

A. Robot Diaries Curriculum

Appendix B

Assignments for CCAC CIT-111 and CIT-130

The assignments in this chapter were used in the Fall 2009 Finch Pilot and were written together by Don Smith (Faculty at CCAC) and the author. The Finch was introduced in assignment four in the introductory course and in assignment two in the intermediate course. All the assignments after the Finch was introduced used the Finch.

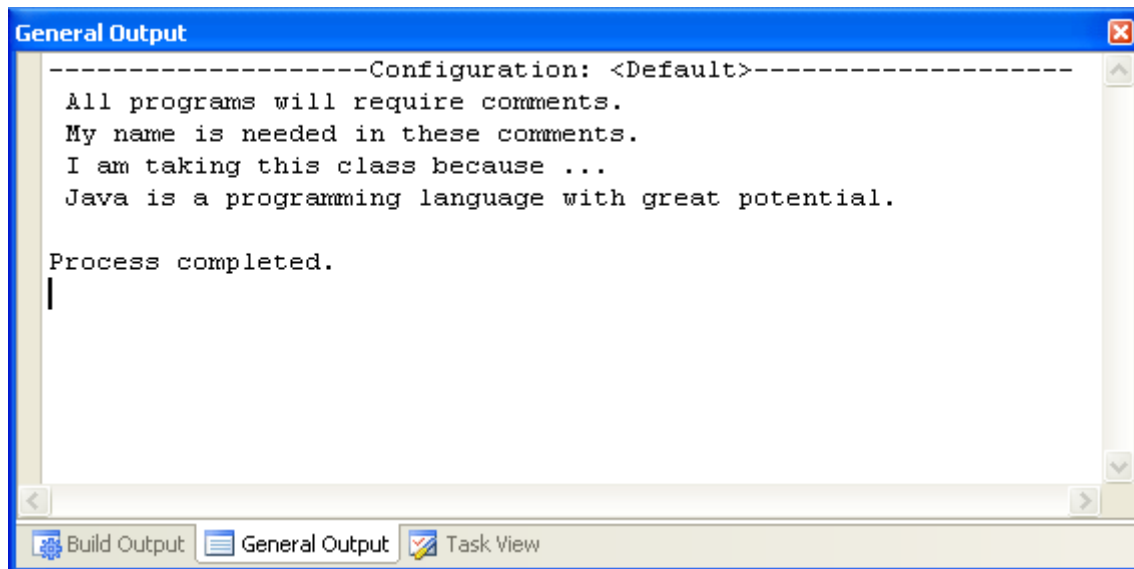
Lab Assignment 1

First Program

Due Date: 08/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign1.

Write, compile, and test a program that displays the following text, as shown below, to the console. Notice the spacing, spelling and case of the output. Be sure to explain why you are taking Java Programming. This explanation can be as short or as long as you want. Please do not use the reason shown in the example.



```
-----Configuration: <Default>-----
All programs will require comments.
My name is needed in these comments.
I am taking this class because ...
Java is a programming language with great potential.

Process completed.
|
```

If you need help.....ask.

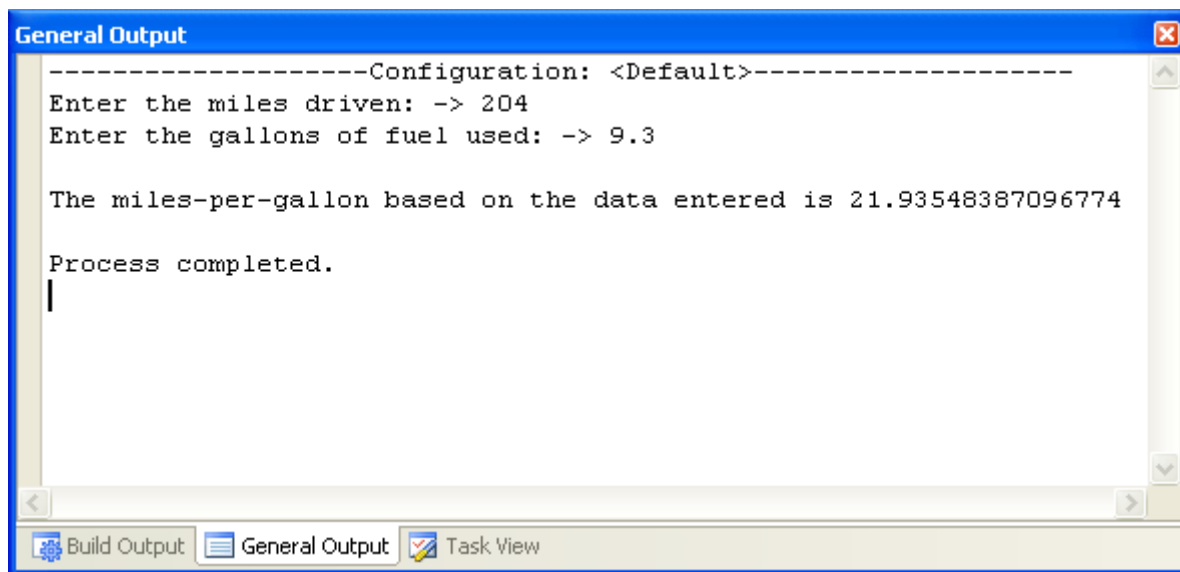
Lab Assignment 2

Using data in a program

Due Date: 08/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign2.

Write, compile, and test a program that prompts for input about a motor vehicle. The program asks the user for the number of miles driven (integer) and the gallons of gas used (double). With the input data the program calculates miles-per-gallon for the vehicle. An example of the program is shown below. (Hint) M.P.G. = miles driven divided by gallons of gas used.



The screenshot shows a 'General Output' window from a Java IDE. The window has a blue title bar and a scrollable text area. The text inside the window is as follows:

```
-----Configuration: <Default>-----  
Enter the miles driven: -> 204  
Enter the gallons of fuel used: -> 9.3  
  
The miles-per-gallon based on the data entered is 21.93548387096774  
  
Process completed.  
|
```

At the bottom of the window, there is a tabbed interface with three tabs: 'Build Output', 'General Output' (which is currently selected), and 'Task View'.

If you need help.....ask.

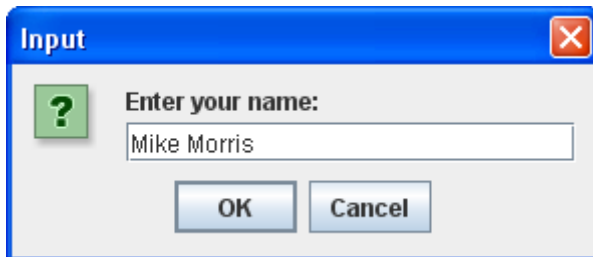
Lab Assignment 3

Variables and Numeric Operators

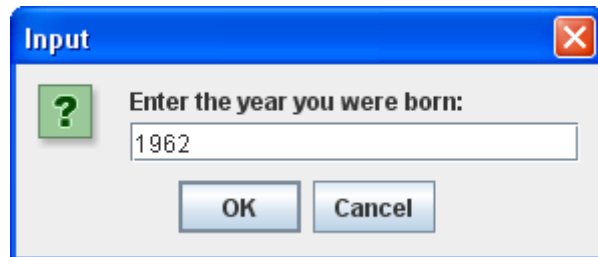
Due Date: 09/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign3.

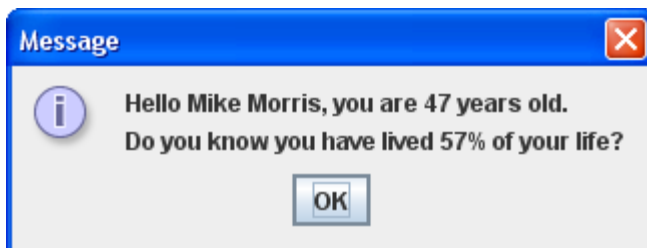
Write a program that prompts the user for input using the JOptionPane class dialog boxes. First prompt the user for their name. Next ask the user for the year they were born. Create two constant variables, one for the current year and the second for the average life expectancy of 82.3 years. Calculate the user's age and the percentage of the user's life lived. Display this information as shown below.



Input dialog box titled "Input" with a question mark icon. The text "Enter your name:" is displayed above a text field containing "Mike Morris". Below the text field are "OK" and "Cancel" buttons.



Input dialog box titled "Input" with a question mark icon. The text "Enter the year you were born:" is displayed above a text field containing "1962". Below the text field are "OK" and "Cancel" buttons.



Message dialog box titled "Message" with an information icon. The text "Hello Mike Morris, you are 47 years old." and "Do you know you have lived 57% of your life?" is displayed. Below the text is an "OK" button.

If you need help.....ask.

Lab Assignment 4

Finch Software and Selection Control Structures

Due Date: 09/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign4.

Write a program that tells the user if the Finch is currently placed with the beak up, beak down, or level. The program should speak the Finch's current orientation; print the orientation to the console. The Finch should also light up the LED red when the Finch is placed with the beak pointing up, blue when the Finch is level, and green with the Finch is oriented with the beak pointed down. The LED should stay lit up for at least five seconds.

Here are some methods from the Finch API that should help:

isBeakUp, isBeakDown, isFinchLevel
saySomething
setLED
sleep

If you need help.....ask.

Lab Assignment 5

Looping

Due Date: 09/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign5.

Program your Finch to wander around a room. When it encounters an obstacle, it should back up and turn away from it randomly for two seconds – you can do this by sending negative random values to the left and right wheel velocities. The program should generate two random values for the wheel velocities using the Random class. The Finch's beak should glow green when it is moving around, and change to be red when it sees an obstacle and it is moving away from it. Since Finches are diminutive and polite creatures, it should also apologize for nearly running into an obstacle when it sees one.

Have the Finch continue running this program until it is picked up and placed on its tail.

Here are some methods from the Finch API that should help:

```
setWheelVelocities  
saySomething  
setLED  
sleep  
isBeakUp  
isObstacleRightSide  
isObstacleLeftSide
```

If you need help.....ask.

Lab Assignment 6

Methods

Due Date: 10/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign6.

Teach your Finch to sing by writing a method that converts notes to tones to play. You have been provided with an extended skeleton file called LabAssign6Skeleton.java – add this to your project and rename it LabAssign6.java. You will need to complete the *notePlayer()* method to complete this program.

The method takes a character as an input – acceptable characters are A,a,B,b,C,c,D,d,E,e,F,f,G, and g. Based on the input, you will need to use the Finch's playTone method to generate the correct musical note. playTone takes two arguments – frequency, and duration. For duration, you can use 600 ms for lower case characters and 1200 ms for upper case characters (so upper case characters are long notes and lower case are short notes). For frequency, use the following mapping of characters to frequencies:

Aa – 440
Bb – 494
Cc – 262
Dd – 294
Ee – 330
Ff – 349
Gg - 392

When a note is played, you should also light the Finch's beak; each note of a different frequency should have a unique color, though notes with the

same frequency but different durations (like 'A' and 'a') can have the same color.

If a character is sent to the method that is not correct (for example, the letter 'J'), the method should return false. If a correct character is sent, it should return true – this functionality will be tested, so make sure you try sending incorrect characters.

Here are some methods from the Finch API that should help:

setLED

playTone

Here are some songs you might recognize:

ccggaaGffeeddCggffeeDggffeeDccggaaGffeeddC

bagabbbbaabdbagabbbbaabaG

CCCdEedefG

cdeccdecefGefG

If you need help.....ask.

Lab Assignment 7

Methods

Due Date: 10/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file as directed below.

Would we rather create R2D2 or the Terminator? That's the central question behind the notion of emotional robots - if we give our robots the ability to feel, they're much less likely to annihilate the human race with nuclear weapons (1) or to enslave us so that they can use us as batteries(2). Right now, the Finch doesn't have any emotions and a world dominated by small but coldly brutal USB tethered robots is only a few iterations of Moore's law away (3). Save us all from this fate by giving the Finch some emotions. In this assignment, you will create a new class, called `MoodyFinch`, which has an internal emotional state. You have been provided with a skeleton file and will need to fill in the following methods:

```
/* Sets the internal emotion variable. */  
public void setEmotion(String setting)
```

```
/* Plays the sequence of the current emotion - this is filled  
in for you */  
public boolean playEmotion()
```

```
/* Returns the Finch's current emotional state */  
public String getEmotion()
```

```
/* The Finch shows that it's angry */  
private void angry()
```

```
/* The Finch runs a happy routine */  
private void happy()
```

```
/* The Finch runs a sad routine */  
private void sad()
```

```
/* The Finch should act apathetic and uncaring */  
private void blah()
```

The angry, happy, sad, and blah routines are private, and so can't be called by any methods outside the MoodyFinch class. Instead, people using the MoodyFinch class will call *playEmotion()* to play whichever of these four routines the Finch is feeling right now. The routines should involve use of the beak LED, the motors, and the Finch should say something whenever one is called.

Two files have been provided to you - LabAssign7Main.java, which calls the MoodyFinch class and uses it, and MoodyFinchSkeleton.java, which you can use to get started writing your class. Good luck, and remember, the fate of the world depends on you!

1 - See Terminator. And Battlestar Galactica (though arguably those robots had emotions, but they should've been programmed with happier ones)

2 - See The Matrix

3 - Computers double in processing power roughly every 18 months

If you need help.....ask.

Lab Assignment 8

Arrays

Due Date: 11/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign8.

This assignment uses the **Finch** robot to collect data from its sensors. The data collected is stored in arrays. Once data collection is complete, the program displays the data in various forms as output to the console. Have your Finch follow a course as defined below. There are 5 data collection points. At each point have the Finch collect data with its right and left light sensors.

When gathering data use the methods *getRightLightSensor()* and *getLeftLightSensor()*. Store the data in two int arrays. One array for the right side light values and another array for the left side light values. While the Finch is traveling through the course collecting data have its beak turn red. Use a loop to have the finch collect data. The array below has the distances information for the Finch to follow. Use the *straight()* method of the Finch class to read in the distances. When the Finch has completed its data collecting, announce completion. Display the follow information using a method for each task:

- Display the light data, left and right sensor values separately
- Use the services of the Arrays class to display the formatted output
- Display the sum of the left and right sensor values by point location
- find the brightest point on the course and display the brightest point which is the largest value of both the left and right sensors summed
- Calculate the average value of the left and right light sensors

Distances:

Start to point 1 is 3 feet

Point 1 to point 2 is 2 feet

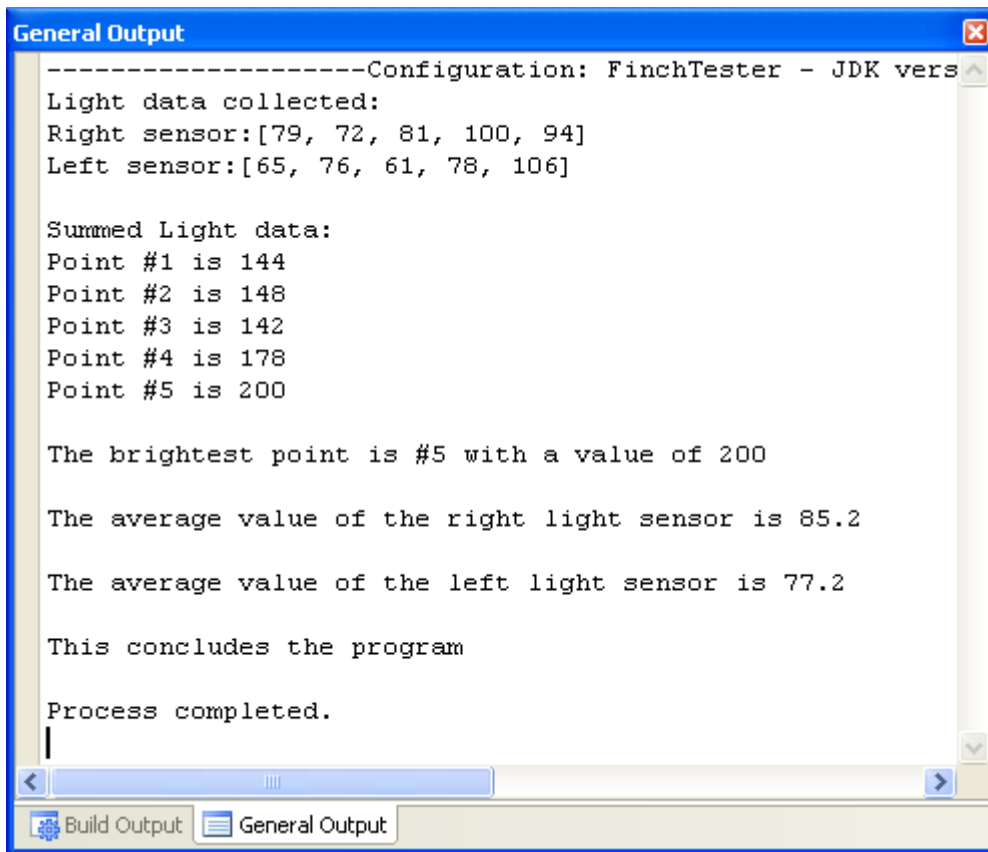
Point 2 to point 3 is 6 feet

Point 3 to point 4 is 2 feet

Point 4 to point 5 is 3 feet

```
double[] distance = {91.44, 60.96, 182.88, 60.96, 91.44 };
```

Example of console output:



```
-----Configuration: FinchTester - JDK vers
Light data collected:
Right sensor:[79, 72, 81, 100, 94]
Left sensor:[65, 76, 61, 78, 106]

Summed Light data:
Point #1 is 144
Point #2 is 148
Point #3 is 142
Point #4 is 178
Point #5 is 200

The brightest point is #5 with a value of 200

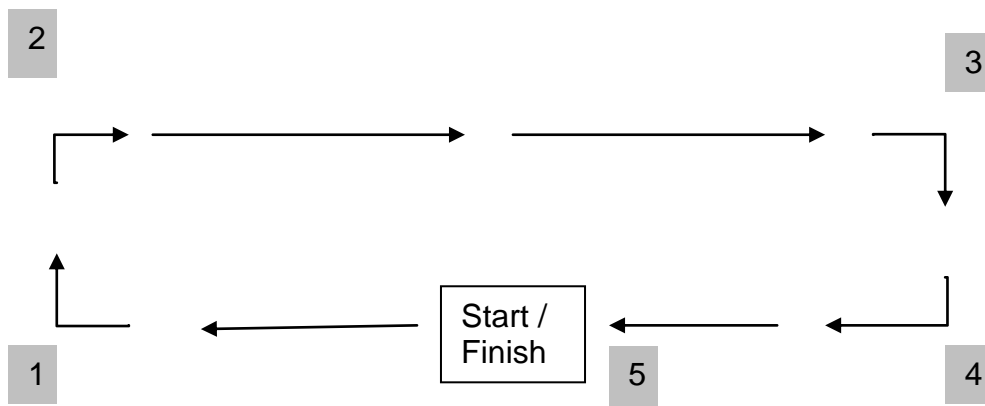
The average value of the right light sensor is 85.2

The average value of the left light sensor is 77.2

This concludes the program

Process completed.
```

Course map:



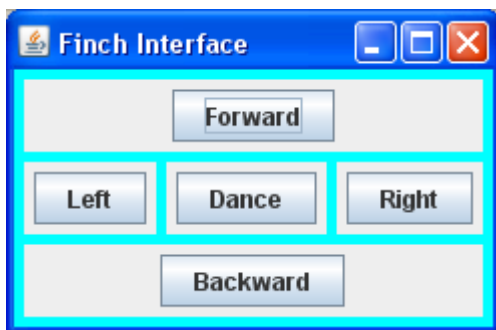
Lab Assignment 9

GUI with Layout Manager

Due Date: 11/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit the Java source code file as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign9.

Create a GUI that uses a JFrame. The GUI has 5 JButtons in the order and location shown below. These buttons are placed inside a JPanel and placed in a BorderLayout. When a button is pushed, the Finch responds by executing the button's label command. The distance for the Finch to travel forward or backwards is 10 — 20 cm. Turning should be in the range of 35 — 90 degrees. The dance button has the Finch perform a short 7 step dance routine. Have the Finch's beak change color for each button command. Below is an example of the GUI.



If you need help.....ask.

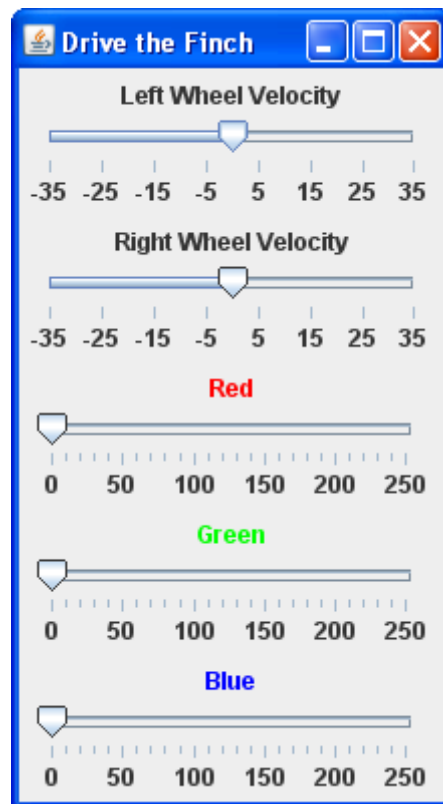
Lab Assignment 10

GUI with Sliders

Due Date: 12/??/09

Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit the Java source code file as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign10.

Create a GUI that uses a JFrame. The GUI has 5 JSliders in the order and location shown below. These JSliders are labeled and placed inside a JPanel. When a JSlider knob is dragged along its track, the Finch responds by moving its wheels or changing beak color. Below is an example of the GUI.



If you need help.....ask.

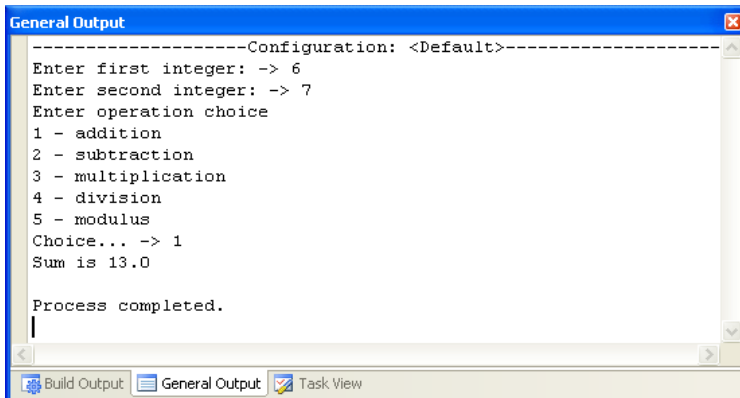
Lab Assignment 1

Java Data Types

Due Date: 08/??/09

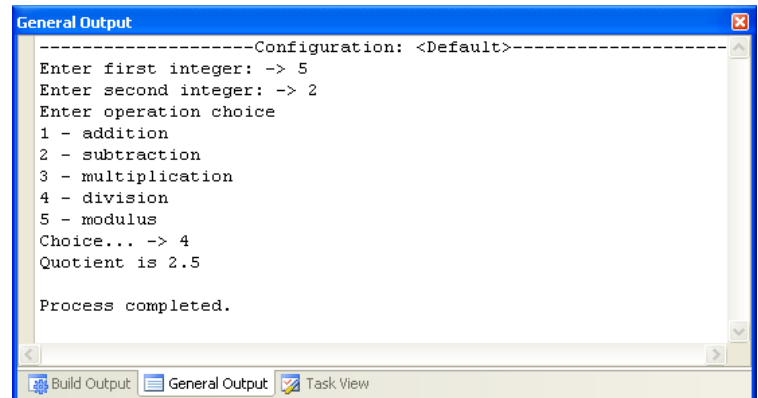
All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign1.

Write a Java program that accepts two integers. Use the Scanner class to read in the input. Ask the user for any of the five arithmetic operations to perform on the two integers. Output is displayed as a floating point number where appropriate. Below are examples of the program executing.



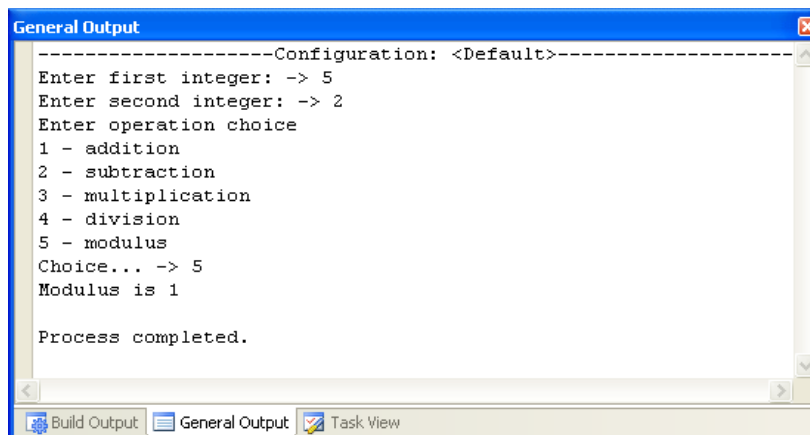
```
-----Configuration: <Default>-----
Enter first integer: -> 6
Enter second integer: -> 7
Enter operation choice
1 - addition
2 - subtraction
3 - multiplication
4 - division
5 - modulus
Choice... -> 1
Sum is 13.0

Process completed.
```



```
-----Configuration: <Default>-----
Enter first integer: -> 5
Enter second integer: -> 2
Enter operation choice
1 - addition
2 - subtraction
3 - multiplication
4 - division
5 - modulus
Choice... -> 4
Quotient is 2.5

Process completed.
```



```
-----Configuration: <Default>-----
Enter first integer: -> 5
Enter second integer: -> 2
Enter operation choice
1 - addition
2 - subtraction
3 - multiplication
4 - division
5 - modulus
Choice... -> 5
Modulus is 1

Process completed.
```

Lab Assignment 2

Using Classes, Dialog Windows, Finch Robot

Due Date: 09/??/09

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign2.

This lab assignment will use the JOptionPane class from the swing package and the Finch robot and software.

The Finch is applying for a job as a weatherman. Give your Finch the ability to talk about the weather by creating a program that:

- Accepts a location as an input from the user.
- Creates a WeatherReader object using this location (don't forget to import RSSReaders.WeatherReader!)
- Extracts the current temperature of the input location
- Has the Finch interpret this temperature by commenting on it and by setting the color of its LED.
 - So for example, if the temperature is 30 or below, you might turn the LED blue and have the Finch say "It's crazy cold out, time to head south". The Finch should say something different and have a different LED color for temperatures 30 and below, between 30 and up to 50, between 50 and up to 70, between 70 and up to 90, and over 90.
 - The Finch should always say the temperature, but it's up to you what else the Finch says or how you light up the LED, they just need to be different.
- Some hints:
 - Take a look at the javadoc for the RSSReaders, specifically the weather reader. Note that it is constructed with a string, and that the getTemperature method is crucial here.
 - You will need the following Finch methods - setLED, sleep, and saySomething; look for these in the Finch javadoc
 - The Finch talks kind of funny, so sometimes it helps to print out what it's saying to the console

- Add a sleep of at least 10 seconds to the end of your program.
- Because `saySomething` doesn't block while it's running, it's possible to call it and then immediately called `myFinch.quit()` - if that happens you'll only hear a fraction of a second of speech. The sleep is there so that you give your Finch time to say whatever it needs to say.
- Your computer must be on the internet for this program to work.
- Here's some typically cold, hot, and moderate cities if you want to test several of your conditions:
 - Barrow, AK
 - Fargo, ND
 - Phoenix, AZ
 - Las Vegas, NV
 - Death Valley, CA
 - San Francisco, CA
 - San Diego, CA

Lab Assignment 3

Methods and Loops

Due Date: 9/??/09

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign3.

Teach your Finch to sing by writing a method that converts notes to tones to play. You will need to complete a method named *notePlayer()* for this assignment. The declaration should be:
public boolean notePlayer(char note)

The method takes a character as an input – acceptable characters are A,a,B,b,C,c,D,d,E,e,F,f,G, and g. Based on the input, you will need to use the Finch's *playTone* method to generate the correct musical note. *playTone* takes two arguments – frequency, and duration. For duration, you can use 600 ms for lower case characters and 1200 ms for upper case characters (so upper case characters are long notes and lower case are short notes). For frequency, use the following mapping of characters to frequencies:

Aa – 440
Bb – 494
Cc – 262
Dd – 294
Ee – 330
Ff – 349
Gg - 392

When a note is played, you should also light the Finch's beak; each note of a different frequency should have a unique color, though notes with the same frequency but different durations (like 'A' and 'a') can have the same color.

If a character is sent to the method that is not correct (for example, the letter 'J'), the method should return false. If a correct character is sent, it should return true – this functionality will be tested, so make sure you try sending incorrect characters.

A second method is needed for accepting input from the user using a JOptionPane input dialog window. This method accepts a String as a series of notes. This method calls the *notePlayer()* method parsing each character of the input String.

The declaration should be:

```
public void parseNotes( )
```

If a bad note is in the input String, the Finch should stop and announce it is stopping.

In the *main()* method create an instance of the LabAssign3 class and use that object to call the *parseNotes()* method.

Here are some songs you might recognize:

ccggaaGffeeddCggffeeDggffeeDccggaaGffeeddC

bagabbbbaabdbagabbbbaabaG

CCCdEedefG

cdeccdecefGefG

If you need help.....ask.

Lab Assignment 4

Data Validation and Exceptions

Due Date: 09/??/09

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign4.

In this lab assignment you will program the Finch to play Midi files and dance to them. To help you do this, we have provided a wrapper class, called MidiPlayer.java, that you should copy in the FinchCode/finch directory and add to the project using Project->add files in JCreator. You can view the commented source of this file for information on how to use the methods of the class. To complete the program you will need to use the following methods from the MidiPlayer class:

public Sequence getSequence (File file) throws IOException, InvalidMidiDataException, Exception

Gets the sequence from the file object sent to the method. The sequence generated is then passed to the play method.

public void play(Sequence sequence, boolean loop) throws InvalidMidiDataException

Plays the Midi file encapsulated in sequence. The second argument is whether to loop play of the file, true for looping, false for no looping.

public int getBeatInterval()

This method returns the interval, in milliseconds, between beats of the song.

public void close()

This closes the sequence and shuts down the player.

To write your program, you should create two methods – *playMidi()*, which gets the midi filename from the user and starts the Finch dancing, and *dance()*, which contains a series of steps that play will select from. Their method declarations are:

```
public void playMidi()
public void dance(int step)
```

Your program should execute as follows:

The *main()* method should instantiate an object of the LabAssign4 class, call the *playMidi()* method, and invoke *System.exit(0)*.

The *playMidi()* method should instantiate globally defined *MidiPlayer* and *Finch* objects. It should request the name of a Midi file from a console prompt. Using the *Scanner* class, assign the file name to a *String*. The *playMidi()* method should then convert the *String* into a *File* object using the *File* class. At this point, you will need to use the *File* object as an argument in the *getSequence()* method of the *Midiplayer* class. This method returns a *Sequence* class object which is needed to call the *play()* method of the *MidiPlayer* class. Notice that *getSequence()* throws three exceptions. Your program should distinguish between these exceptions as follows:

- If an *IOException* is caught, the user entered in a bad filename, and the program should loop back to where the user was asked to enter a filename.
- If an *InvalidMidiDataException* is caught, the Midi file was corrupt. The program should exit using *System.exit(0)*; and print an error message like "Data corrupt".
- If a general *Exception* is caught, the program should exit and print an error message like "Unknown Error".

Once a *Sequence* class object is generated, you should call the *MidiPlayer*'s *play()* method. Note that it too throws an exception, and you will need to catch this in the same way you caught the *InvalidMidiDataException* for the *getSequence()* method.

The *playMidi()* method should use the *getBeatInterval()* method to determine the time between beats. This value will be used by the *Finch* to move to the beat. Start the *Finch* in a loop that occurs so long as its beak is not pointed up. The *dance()* method (see below) will be called using one of

six step values as an argument. Finally close the MidiPlayer and quit the Finch to end the *playMidi()* method.

Create a *dance()* method that switches between six different steps, and use the loop to go through these steps in order. The steps are up to you, but they should look fairly different and each should have its own LED color. Use the *setWheelVelocities()* and *setLED()* methods to accomplish this. Time the steps to go with the beats by moving to the next one at the interval between beats.

Once the Finch is picked up and pointed with beak up, the MidiPlayer should close and the Finch should quit Again this functionality is built in the *playMidi()* method.

We have provided you with four midi files and a corrupt file. Place these in the FinchCode directory. Try them all, and try invalid file names as well – we will test whether your program correctly catches both invalid filenames (IOException) and corrupt midi data (InvalidMidiDataException).

P.S. You can get more midi files at <http://www.mididb.com/>

Lab Assignment 5

Finch plays Simon

Due Date: 10/??/09

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign5.

Program your Finch to play a modified version of the memory game Simon. In the game you will write, the Finch will print and say an orientation, either Up, Down, Left Wing Down, or Right Wing Down. You will then need to move the Finch to that orientation. If you do this successfully, the Finch will print a new orientation – you'll have to move the Finch to the first orientation, then to the new one. This game goes on for as many moves as you can remember. When you finally mess up, the Finch will tell you how many moves in a row you got right, and either congratulate or insult you for it.

Here's the details:

Your *main()* method should instantiate an object of the LabAssign5 class and then call one method with that object – the *playGame()* method. *playGame()* has the following structure:

- Instantiate the globally defined Finch object
- Instantiate a Random class object
- Create an ArrayList of type String

Start a loop that ends when the player makes an incorrect move. The loop should:

- Use the Random class object to generate a random number from 0 to 3
- Use a control structure to add to the ArrayList a String containing the value “Up” for 0, “Down” for 1, “Left Wing Down” for 2, and “Right Wing Down” for 3
- Print and say what the move is
- Print and say that the user needs to repeat all of the moves done so far.

At this point, you should create a loop that goes through every move so far, starting with the first one created. This loop checks that a move is correct by calling a second method, *moveCorrect()*.

moveCorrect() returns a Boolean value, and is passed a String. For each of the four possible moves, check if the Finch is in the orientation suggested by the String (either Up, Down, Left Wing Down, or Right Wing Down). Give the player five seconds to get to the correct move – you can continue checking throughout the five seconds by constantly reading the Finch methods, like *myFinch.isBeakUp()* (for example, if you set a counter in the loop and had a sleep of 100 milliseconds, you could cause the loop to exit if either the *myFinch.isBeakUp()* method became true, or if the counter exceeded 50 (50*100 milliseconds = 5 seconds). If the player gets the correct move, the Finch’s beak should flash green for 1 second, the Finch should beep happily, and the *moveCorrect()* method should return true. If they don’t get it within five seconds, *moveCorrect()* should return false.

If the player correctly gets the whole sequence, your overall loop should continue and generate a new move for them to remember. If they mess up, the program should tell them the maximum number of moves they reached. The Finch should also render judgment – if the total moves are less than 4, insult them for having a bad memory, if moves are 4-8, tell them to try again next time, and if it’s 9 or more, congratulate them

Lab Assignment 6

Inheritance

Due Date: 10/??/09

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class as instructed below.

Would we rather create R2D2 or the Terminator? That's the central question behind the notion of emotional robots - if we give our robots the ability to feel, they're much less likely to annihilate the human race with nuclear weapons(1) or to enslave us so that they can use us as batteries(2). Right now, the Finch doesn't have any emotions and a world dominated by small but coldly brutal USB tethered robots is only a few iterations of Moore's law away (3). Save us all from this fate by extending the Finch class to give it some emotions; you will create a new class, called `MoodyFinch`, which has an internal emotional state. Because `MoodyFinch` is an extension of the `Finch` class, it will have all of the methods and capabilities of the `Finch` class, but you will add to those by writing several unique methods in `MoodyFinch`, specifically:

```
/* Sets the internal emotion variable.
 * Returns true if an acceptable emotion (Angry, Sad, Happy, or Blah)
 * was used, false otherwise
 */
public boolean setEmotion(String setting)

/* Plays the sequence of the current emotion */
public void playEmotion( )

/* Returns the Finch's current emotional state */
public String getEmotion( )

/* The Finch shows that it's angry */
private void angry( )
```

```
/* The Finch runs a happy routine */  
private void happy( )
```

```
/* The Finch runs a sad routine */  
private void sad( )
```

```
/* The Finch should act apathetic and uncaring */  
private void blah( )
```

The angry, happy, sad, and blah methods are private, and so can't be called by any methods outside the MoodyFinch class. Instead, people using the MoodyFinch class will call *playEmotion()* to play whichever of these four routines the Finch is feeling right now. The routines should involve use of the beak LED, the motors, and the Finch should say something whenever one is called.

The client class has been provided to you - LabAssign6Main.java, which creates and instance of the MoodyFinch. Good luck, and remember, the fate of the world depends on you!

1 - See Terminator. And Battlestar Galactica (though arguably those robots had emotions, but they should've been programmed with happier ones)

2 - See The Matrix

3 - Or so sci-fi authors would have us believe

Lab Assignment 7

Abstract and Interface

Due Date: 11/??/09

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class as instructed below.

The year is 2030. Robots have become common commodities. Our government has determined a need to standardize the robotics industry. They have provided a standard interface with methods that need apply to all robots on wheels. This interface is provided. Your task is to implement the interface class **Robot.java**. You will create a class named **StandardFinch**. This class will use composition rather than inheritance. The **StandardFinch** class will be composed of a Finch class object. Declare the Finch object as a class instance variable and instantiate it in the method *initialize()*. You will test the Java class completed with **StandardFinchDriver.java**.

Before you start this assignment, please read the information provided at Sun Microsystem's web site. This information should help you understand the task ahead. The link is provided.

<http://java.sun.com/docs/books/tutorial/java/landl/createinterface.html>

Lab Assignment 8

GUI

Due Date: 11/??/09

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign8.

Write a GUI application that powers the Finch. The application will use a JFrame. In the north position of the window a label appears and in the south position a series of movement buttons. Two graphic images are in the east and west regions of the GUI. The center position has a series of buttons when pushed use a group of RSS feeder classes. Write separate methods for these button routines.

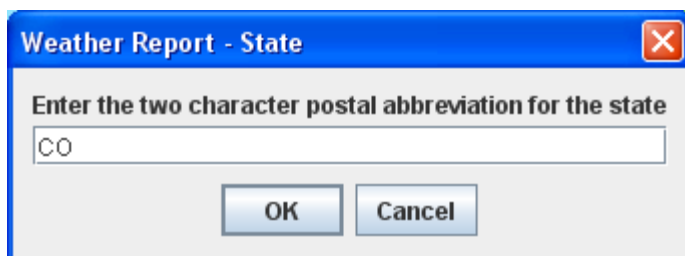
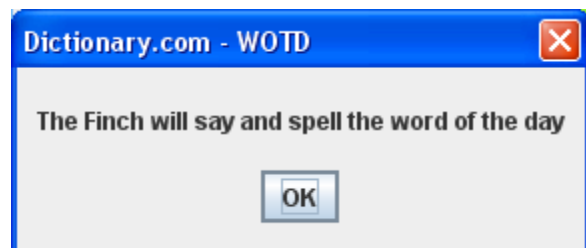
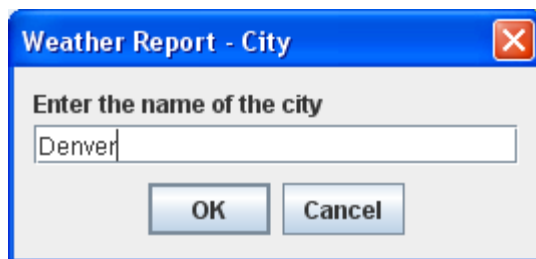
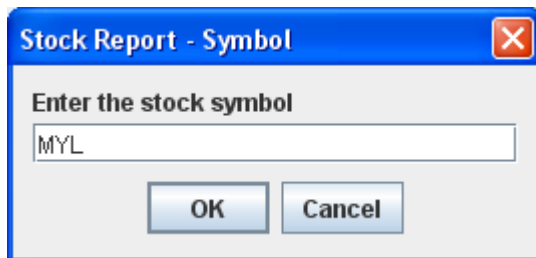
If the “Stock Price” button is pushed a dialog window prompts the user for a stock symbol and the Finch reports the stock name and current trading price for the stock symbol entered. The beak turns red if stock price is less than \$10.00 and turns green if stock price is \$50.00 or greater, otherwise the beak color is yellow.

When the “Weather Report” button is pushed a dialog window prompts for the city and state. The Finch reports the current condition for the location entered. If the temperature is 32 or less degree the beak of the Finch turns blue and if temperature is greater than 75 degrees the beak is red, otherwise the beak color is yellow.

The “Music Chart” button prompts the user for a chart position and the Finch reports the song and musical artist. The beak turns cyan in color while reporting the information.

When the “Dictionary” button is pushed a message dialog window appears stating the Finch will say and spell the word of the day and then does so. The movement buttons at the bottom of the GUI cause the Finch to perform the button’s label action.

Your GUI should look similar to the example below. The window has a set size that cannot be changed.



Lab Assignment 9

My Finch GUI

Due Date: 12/03/09(final deadline)

All programs should use standard Java naming conventions, have proper indentation and comments. Points will be deducted for programming assignments not having these elements. Your name and Lab assignment number must be included at the top of the Java source file in comments. Submit this Lab assignment as an attachment via e-mail. Be sure to include in the e-mail, your name, lab assignment number, and class section. My e-mail address is dsteach9@hotmail.com . Print a hard copy of the source code to be submitted in class. Name your file and class name LabAssign9.

Write a GUI application that powers the Finch. The application will use a JFrame. Use layout managers, and Swing class GUI components. Your grade for this assignment will be based on complexity, design and functionality.

Grading Criteria:

Three or more layout managers.....	30pts
Implement two or more interface classes.....	30pts
Text field input.....	20pts
Embedded graphics.....	10pts
Naming conventions, indents, comments.....	10pts

Appendix C

Finch Documentation

This Appendix contains documentation materials provided to students in the Finch pilots at CCAC. It includes four files:

- The Finch Manual - A description of the Finch hardware, covering the available sensors and outputs.
- Finch Quick Reference - A listing of the most frequently used program methods of the Finch class/API.
- Creating and Running Robot Programs - A guide to creating and compiling new programs in Jcreator. Jcreator is the software environment that was used in the courses for both Finch and standard programming assignments.
- Finch Driver Setup - A manual for installing the USB driver required for Finch operation.

Finch Manual

Introduction

This document provides a quick overview of the Finch, a new robot for use in computer science education. Finch is fully programmable in Java, and allows students to use a host of sensors and output devices not available to them if using just a computer. It is also very cute.

USB

The Finch uses an FTDI FT232R USB to serial converter chip to communicate with a computer. As with all USB devices, the Finch requires a driver to be installed on the computer before it is plugged in for the first time. Drivers are available here:

<http://www.ftdichip.com/Drivers/VCP.htm>

The Finch draws all power from USB, and is technically classified as a high-current USB device. It communicates at a baud rate of 57600, and when plugged in will show up as a serial or 'COM' port.

Sensors

The Finch is capable of sensing ambient light levels, temperature, obstacles placed in front of it, and acceleration. Refer to the annotated images of the Finch on the next page to see where sensors are placed on the Finch.

Light. The Finch uses two photoresistors to detect ambient light levels. The sensors are placed in the front of the robot on either side of the hump. They are placed to allow the robot to easily sense the direction of a bright light source, so as to turn towards or away from it.

Temperature. The Finch has a single thermistor sensor that detects the ambient temperature. The sensor can also determine the temperature of an object if the object is placed in contact with the sensor. The temperature sensor is accurate to within 2 degrees Fahrenheit.

Obstacles. The Finch can detect if obstacles are placed in front of it. The sensors face forward, and are on the left and right side of the Finch so as to detect obstacles placed on both sides of the Finch. The sensors can detect obstacles up to 1 foot away.

Due to the limitations of this type of sensor, some objects, primarily those made of black plastic, will not be sensed.

Accelerometers. The Finch uses a 3-axis MEMS accelerometer to detect acceleration in all three spatial dimensions. The sensor can detect accelerations of +/- 1.5 gees. The primary use of the accelerometer is to detect the direction of gravity, so as to know how the Finch is oriented (flat on the ground, upright, etc). It is also possible to detect spikes in acceleration caused by tapping the Finch, or quickly moving it by hand.

Motors

The Finch has two motors and uses its tail as a slide caster. It can turn in place around the axis of its two wheels. The wheels are pushed against the motor shafts, eliminating the need for a gearing system. Black and white encoder stickers are stuck to the inside of each wheel to allow the Finch to track and control its position and velocity. The encoders have a resolution of 0.75 cm or 0.3 inches per tick.

Light & Sound

The Finch has a full-color LED embedded in its 'beak'. This LED contains red, green, and blue elements. By setting the intensity of each element the LED can be controlled to make any color. There are 256 settings for intensity for each color element, ranging from 0 (off) to 255 (full on).

The Finch has an on-board buzzer which is capable of playing sounds with frequencies between 100 Hz and 10 KHz. Software that comes with the Finch also allows the programmer to control computer speakers so as to play synthesized speech, wav files, or musical notes.

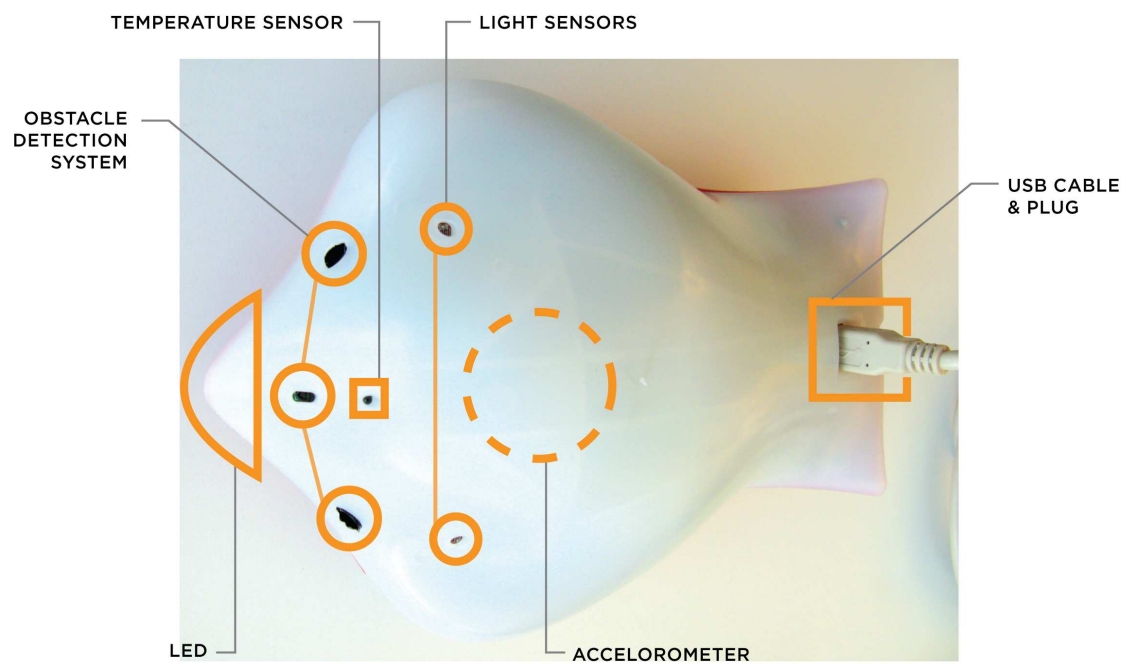


Image 1: Finch Sensor Placement

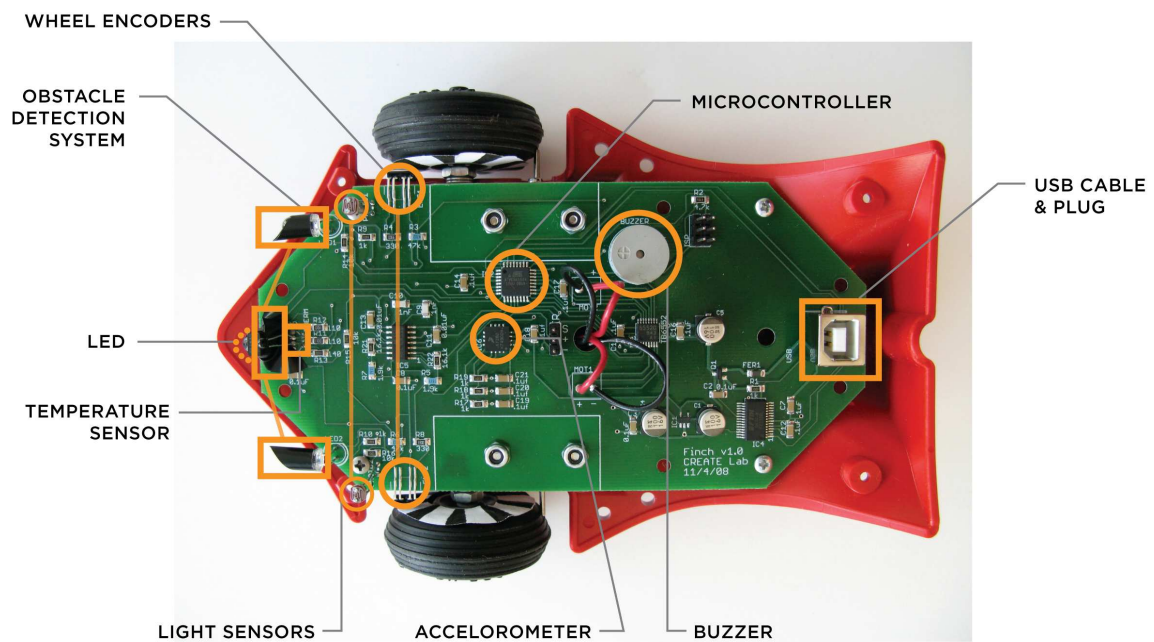


Image 2: Finch Interior

Finch Quick Reference

You can see full definitions of method calls and constructors for the various robot related classes you will use in this course by double clicking on the `FinchDocumentationQuickLink.html` file. This document is meant as a quick reference to some of the methods that you will be using most often.

Speakers:

```
void saySomething(String text)
```

Uses the Text To Speech library to synthesize audio from the `text` variable, and play that audio on the computer's speakers. Note that this program blocks for the amount of time that it takes to synthesize the audio - for example, synthesizing the sentence "Hello world" may block further program execution for two seconds.

```
void playTone(int frequency, int duration)
```

Plays a tone over the computer speakers or headphones at a given frequency (in Hertz) for a specified duration in milliseconds. Middle C is about 262Hz. Visit <http://www.phy.mtu.edu/~suits/notefreqs.html> for frequencies of musical notes.

Motors:

```
void setWheelVelocities(double leftVelocity, double rightVelocity)
```

Sets both Finch's wheels to the velocities specified. `leftVelocity` and `rightVelocity` are specified in centimeters per second. The range is from -35 to +35. To backup, send negative velocities to this method.

```
void setWheelVelocities(double leftVelocity, double rightVelocity,  
                        int timeToHold)
```

Same as above but only sets the wheels to spin at the velocity specified by `timeToHold`. `timeToHold` is in milliseconds.

```
void stopWheels()
```

Stops the Finch's wheels.

LEDs

```
void setLED(int red, int green, int blue)
```

Sets the color of the LED in the Finch's beak. The LED can be any color that can be created by mixing red, green, and blue; turning on all three colors in equal amounts results in white light. Valid ranges for the red, green, and blue elements are 0 to 255.

Sensors:

```
int getLeftLightSensor()  
int getRightLightSensor()
```

Returns the value of the left or right light sensor. Valid values range from 0 to 255, with higher values indicating more light is being detected by the sensor.

```
boolean isObstacleLeftSide()  
boolean isObstacleRightSide()
```

Returns true if there is an obstruction in front of the left or right sides of the robot.

```
double getTemperature()
```

Returns the current temperature reading at the temperature probe. The value returned is in Celsius. To get Fahrenheit from Celsius, multiply the number by 1.8 and then add 32.

```
double getXAcceleration()  
double getYAcceleration()  
double getZAcceleration()
```

This method returns the current X, Y, or Z acceleration value experienced by the robot. Values for acceleration range from -1.5 to +1.5 gees.

```
boolean isFinchLevel()  
boolean isFinchUpsideDown()  
boolean isBeakUp()  
boolean isBeakDown()  
boolean isLeftWingDown()  
boolean isRightWingDown()
```

These methods are used to get the Finch's orientation – they return true if the Finch is level, upside down, etc, and false otherwise. Only one of these methods will return true for any given orientation. For example, if you place the Finch flat, isFinchLevel will be true and the rest will be false.

Miscellaneous

```
void sleep(int ms)
```

This method uses Thread.sleep to cause the currently running program to sleep for the specified number of milliseconds.

Creating and Running Robot Programs

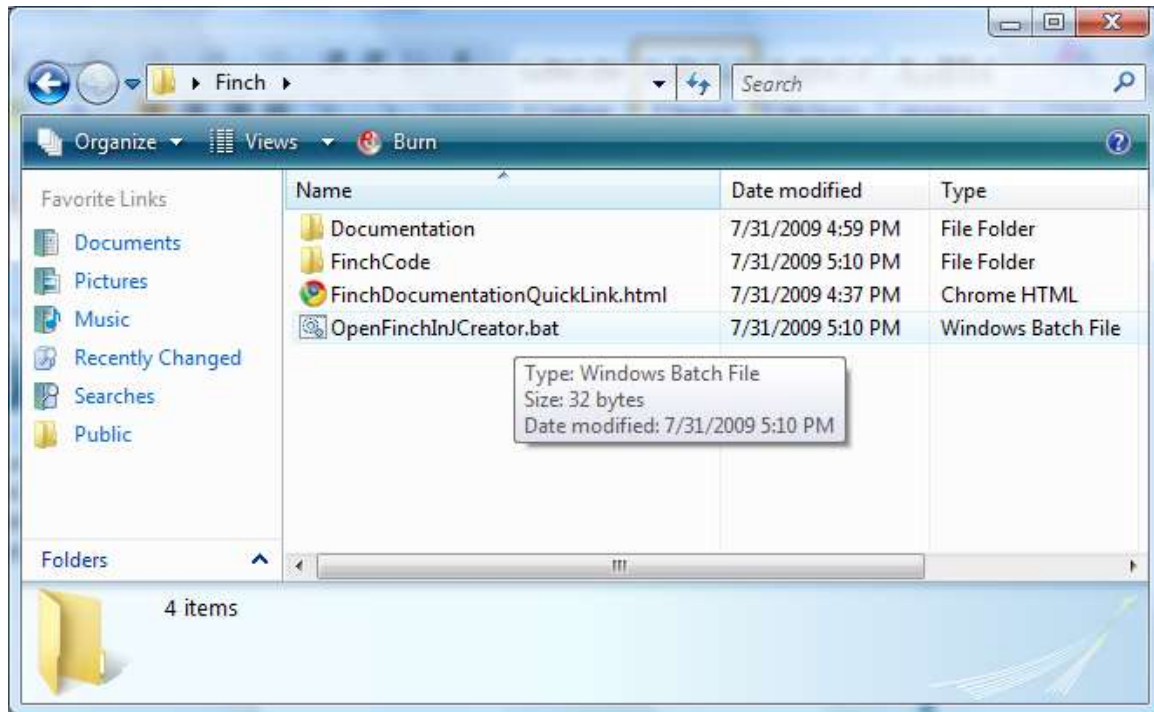
To write programs for the Finch, you will need to work within a Java project. For those who use JCreator, a project is already available with the Finch software package. The following is a step by step guide for how to compile and run programs for the Finch robot in JCreator. If you wish to use an IDE other than JCreator, there are instructions available at:

<http://csbots.wetpaint.com/page/How+to+Compile>
for several other IDEs.

If this is your first time using the robot, it is strongly suggested you do the following *in order*.

Opening the FinchProject

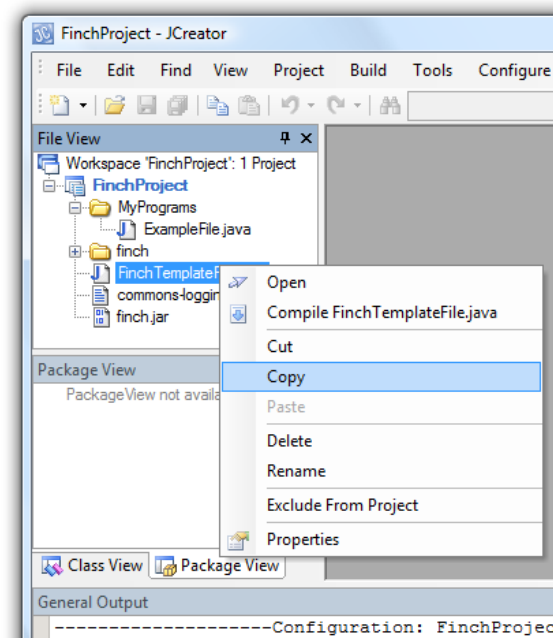
1.) Open the project by double clicking the OpenFinchInJCreator batch file in your class folder. The JCreator IDE should start.



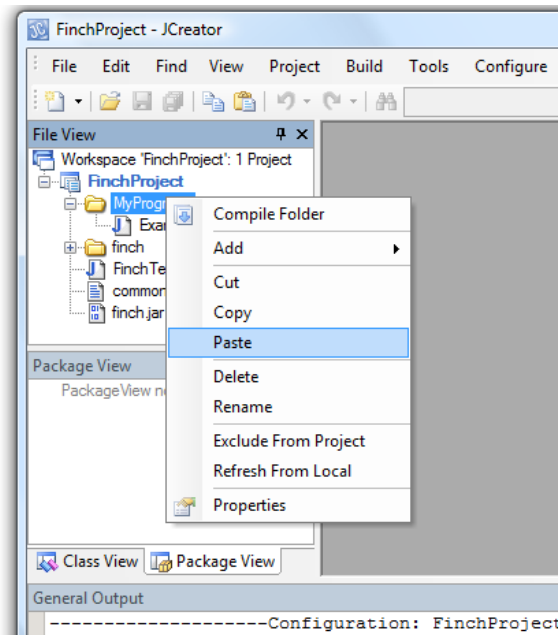
It is possible that JCreator will not open if you have not associated .jcp files with JCreator yet – if this is the case, you will need to go into the FinchCode directory, right-click on FinchProject.jcp, and select “Open With... “. Once you’ve done that, navigate to C:\Program Files\Xinox Software\JCreatorV4LE and click on JCreator.exe.

Adding new files to the project

2.) The file FinchTemplateFile.java is a skeleton file for you to copy and create new programs with. Instead of editing FinchTemplateFile.java directly, you should make a new file so that you always have a clean starter copy of FinchTemplateFile.java to fall back on. To make a new file from FinchTemplateFile.java, right click on FinchTemplateFile.java and select copy (1). Then right click on MyPrograms, and select paste (2).

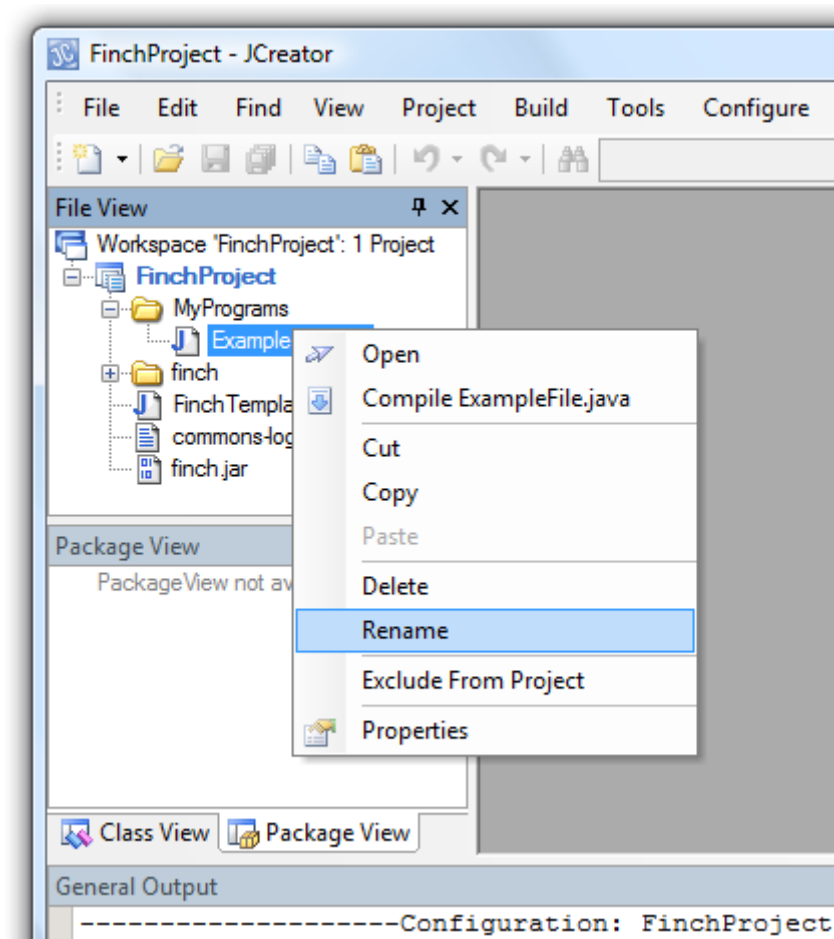


(1) Copying the File



(2) Pasting the File

3.) You will also want to rename the file. Right click on the new FinchTemplateFile.java file in the MyPrograms folder and click on rename. Rename the file myTest.java.



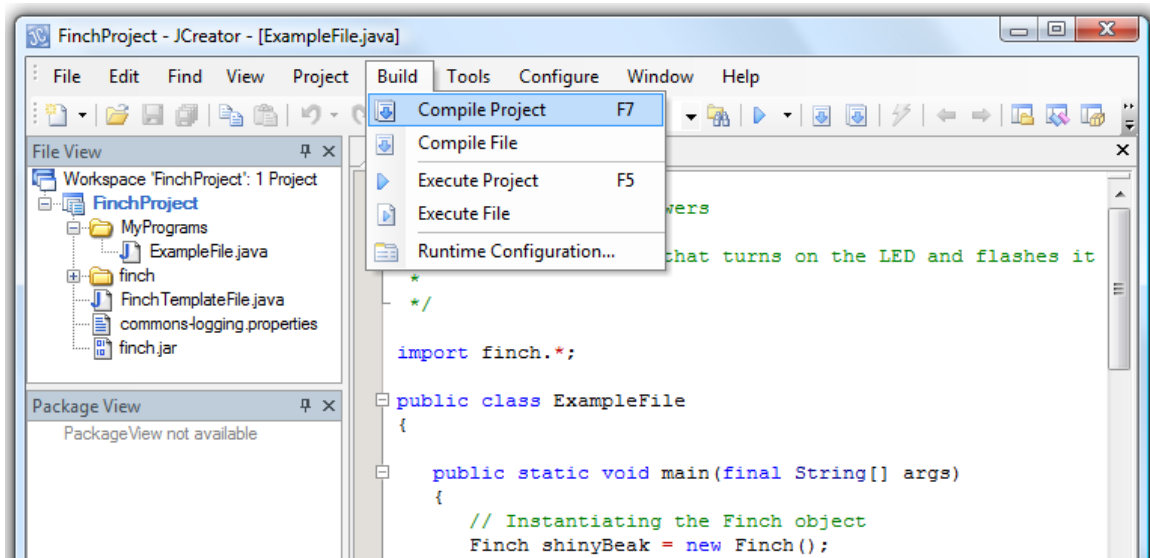
4.) Because Java requires that all class names and filenames match, you will also have to rename the class name. To do this, edit the line that reads “public class **MyFirstFinch**” to read “public class **myTest**”. You can now edit this file and write code to your heart’s content. For any future assignment, just follow the same procedure for adding files to the project.

Compiling and Running a Test Program

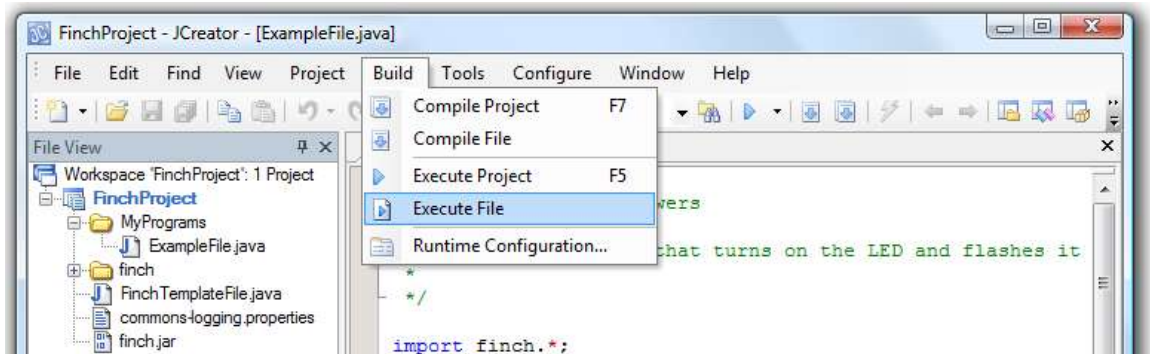
5.) Before you attempt to run a test program, make sure that the Finch is plugged into a USB port on the computer.

6.) Double-click on the file that you want to run by double clicking on it in the 'File View'.

7.) Compile the project by either hitting the F7 key or going to Build->'Compile Project' in the top menu.



8.) Now run the file that is by going to the 'Build' menu and clicking on 'Execute file'. DO NOT run the file by clicking on 'Execute Project' or by clicking on the blue run arrow.



Finch Driver Setup

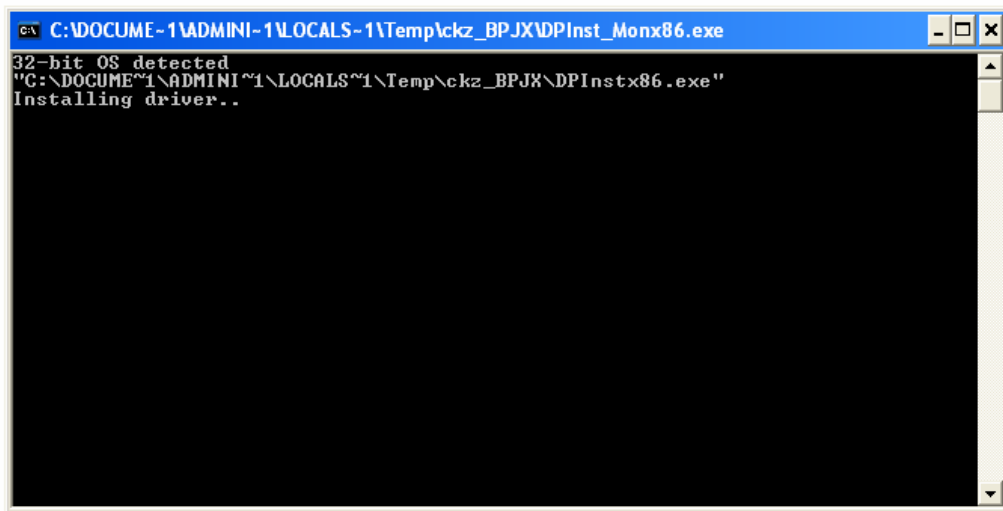
You must install the proper driver before plugging in the Finch. If you have already plugged the Finch in, just unplug it and follow the steps for your OS.

Windows XP or Vista

Go to the following URL and download the file at that webpage:

<http://www.ftdichip.com/Drivers/CDM/CDM%202.04.16.exe>

Run this executable. It will pop up a command line window that will look something like this:



```
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\ckz_BPJX\DPInst_Monx86.exe
32-bit OS detected
"C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\ckz_BPJX\DPInstx86.exe"
Installing driver..
```

This window may only appear for a few seconds. If it appears and then disappears, this means the driver was successfully installed. You may now plug in your Finch.

Mac OSX

Download the driver from the following link:

Intel Macs:

http://www.ftdichip.com/Drivers/VCP/MacOSX/UniBin/FTDIUSBSerialDriver_v2_2_10.dmg

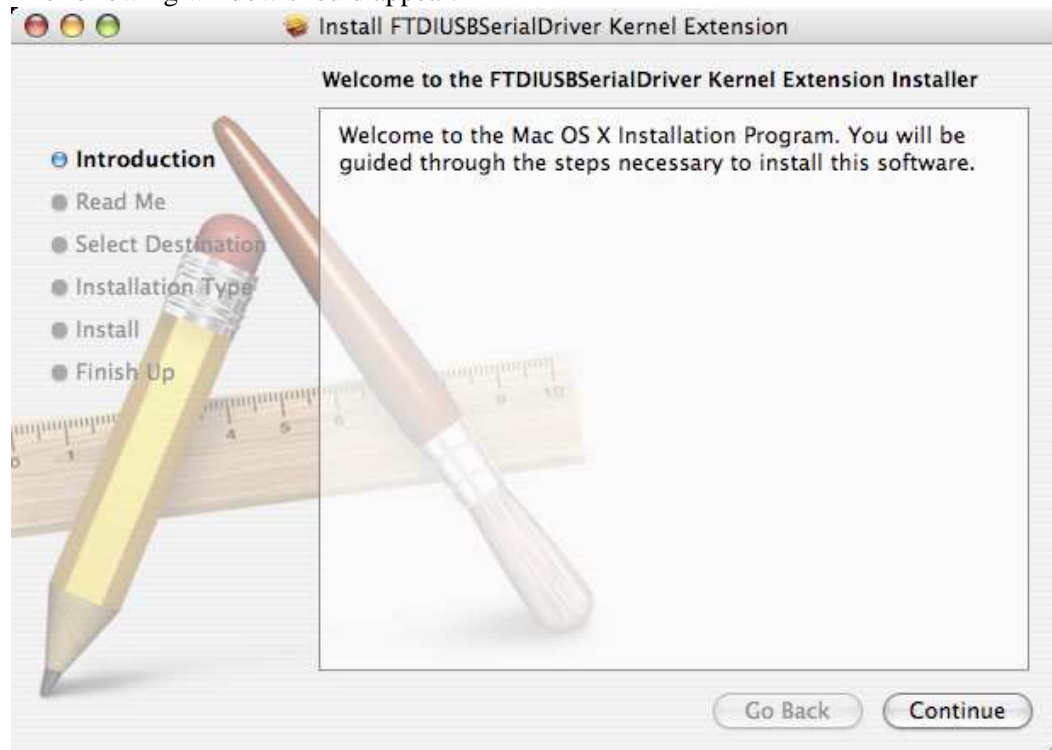
Power PC (before 2007):

http://www.ftdichip.com/Drivers/VCP/MacOSX/FTDIUSBSerialDriver_v2_1_10.dmg

Before you take the following steps, note that installation will require you to restart your computer.

Run the .dmg file that is downloaded.

The following window should appear:



Click through the various Continue and Install menus to install the driver – you should not need to use anything but the default settings.

Other Operating Systems (Linux, etc):

Go to the following website and select the appropriate driver:

<http://www.ftdichip.com/Drivers/VCP.htm>

Bibliography

- AAAS (2007). *Atlas of Science Literacy*. Project 2061 and the National Science Teachers Association.
- Abimbola, R., Alismail, H., Belousov, S. M., Dias, B., Dias, M. F., Dias, M. B., Fanaswala, I., Hall, B., Nuffer, D., Teves, E. A., Thurston, J., and Velazquez, A. (2009). istep tanzania 2009: Inaugural experience. *CMU Tech Report CMU-RI-TR-09-33*.
- Acroname (2009). Brainstem overview. [Http://www.acroname.com/brainstem/brainstem.html](http://www.acroname.com/brainstem/brainstem.html).
- Adams, J. C. (2007). Alice, middle schoolers & the imaginary worlds camps. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*.
- Arsenault, J., Godsoe, S., Holden, C., and Vetelino, J. (2005). Integration of sensors into secondary school classrooms. *Proceedings of Frontiers in Education*.
- Atmel (2006). Atmega88 datasheet. [Http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf).
- Banzi, M. (2008). *Getting Started with Arduino*. O'Reilly.
- Barnes, D. J. and Kolling, M. (2005). *Objects First with Java. 2nd edition*. Pearson Education.
- Bell, P. (2004). On the theoretical breadth of design-based research in education. *Educational Psychologist*, 39:243–253.
- Bell, T. and Fellows, M. (2010). Cs unplugged. [Http://csunplugged.org](http://csunplugged.org).
- Bernstein, D. (2010). *Developing technological fluency through creative robotics*. University of Pittsburgh Doctoral Thesis.

BIBLIOGRAPHY

- Bielaczyc, K. (2006). Designing social infrastructure: Critical issues in creating learning environments with technology. *Journal of the Learning Sciences*, 15:301–330.
- Biggs, S. (1989). *Resource-poor farmer participation in research: a synthesis of experiences from nine national agricultural research systems*. International Service for National Agricultural Research.
- Bjerknes, G., Ehn, P., and Kyng, M. (1987). *Computers and democracy: A Scandinavian challenge*. Aldershot.
- Blank, D., Kumar, D., Marshall, J., and Meeden, L. (2007). Advanced robotics projects for undergraduate students. In *Symposium on Robots and Robot Venues: Resources for AI Education*.
- Bloom, B. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. David McKay Co Inc.
- Blum, L., Cortina, T., Lazowska, E., and Wise, J. (2008). Special session: The expansion of cs4hs: An outreach program for high school teachers. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2008)*.
- Botball (2009). Botball website. [Http://www.botball.org/](http://www.botball.org/).
- Brand, P. and Schwittay, A. (2006). The missing piece: Human-driven design and research in ict and development. *Information and Communication Technologies and Development, 2006. ICTD '06*, pages 2–10.
- Brown, A. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, 2:141–178.
- Brown, B. (2007). *A Guide to Programming in Java; 2nd edition*. Lawrenceville Press.
- Buechley, L. and Eisenberg, M. (2007). Fabric pcbs, electronic sequins, and socket buttons: Techniques for e-textile craft. *Journal of Personal and Ubiquitous Computing*.
- Bunnell, T. H., Yarrington, D. M., and Polikoff, J. B. (2000). STAR: articulation training for young children. In *International Conference on Spoken Language Processing*.

BIBLIOGRAPHY

- Cannon, K. R., Panciera, K. A., and Papanikolopoulos, N. P. (2007). Second annual robotics summer camp for underrepresented students. *Proceedings of the 12th Annual Conference on innovation and Technology in Computer Science Education*.
- Chamot, J. (2006). Electronic braille tutor teaches independence. *National Science Foundation Press Release*.
- Class, B. (2009). *A Blended Socio-Constructivist Course with an Activity-Based, Collaborative Learning Environment Intended for Trainers of Conference Interpreters*, PhD dissertation 420. Facult de psychologie et des sciences de l'education, Universit de Genve.
- Cohen, S. A. (1987). Instructional alignment: Searching for a magic bullet. *Educational Researcher*, 16:16–20.
- Collins, A. (1992). *Toward a design science of education*. Springer-Verlag.
- Cooper, S., Dann, W., and Pausch, R. (2003). Teaching objects first in introductory computer science. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003)*.
- Cornwall, A. and Jewkes, R. (1995). What is participatory research? *Social Science and Medicine*, 41:1667–1676.
- Crowley, K. and Jacobs, M. (2002). *Buildings islands of expertise in everyday family activity*. In G. Leinhardt, K. Crowley & K. Knutson (Eds.), *Learning Conversations in Museums.*, pages 333–356. Lawrence Erlbaum Associates.
- Deitel, H. and Deitel, P. (2005). *Java How to Program. 6th edition*. Pearson Education, Inc.
- Dias, M. B. (2010). Techbridgeworld website. [Http://www.techbridgeworld.org](http://www.techbridgeworld.org).
- Dias, M. B., Dias, M. F., Belousov, S., Rahman, M. K., Sanghvi, S., and El-Moughny, N. (2009). Enhancing an automated braille writing tutor. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Dijkstra, E. (1988). On the cruelty of really teaching computing science. [Http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html](http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html).
- DiSalvo, C. (2007). Neighborhood networks website. [Http://www.neighborhood-networks.net/index.html](http://www.neighborhood-networks.net/index.html).

BIBLIOGRAPHY

- Doerschuk, P., Liu, J., , and Mann, J. (2007). Pilot summer camps in computing for middle school girls: from organization through assessment. *Proceedings of the 12th Annual Conference on innovation and Technology in Computer Science Education*.
- Druin, A. (1999). Cooperative inquiry: developing new technologies for children with children. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 592–599, New York, NY, USA. ACM.
- Druin, A. and Hendler, J. (2000). *Robots for kids: exploring new technologies for learning*. Morgan Kaufmann.
- Elia, J. (1986). *An alignment experiment in vocabulary instruction: Varying instructional practice and test item formats to measure transfer with low SES fourth graders*. PhD thesis, University of San Francisco.
- Fagin, B. and Merkle, L. (2003). Measuring the effectiveness of robots in teaching computer science. In *SIGCSE*, pages 307–311.
- Fahey, P. (1986). *Learning transfer in main ideas instruction: Effects of instructional alignment and aptitude on main idea test scores*. PhD thesis, University of San Francisco.
- FIRST (2009a). For inspiration and recognition of science and technology (first) website. [Http://www.usfirst.org/](http://www.usfirst.org/).
- FIRST, I. (2009b). Innovation first inc. [Http://www.innovationfirst.com/](http://www.innovationfirst.com/).
- Fitzpatrick, D. (2008). City to have a robotic birthday party. *Pittsburgh Post-Gazette*. [Http://www.post-gazette.com/pg/07124/783255-85.stm](http://www.post-gazette.com/pg/07124/783255-85.stm).
- Forouzan, B. A. and Gilberg, R. F. (2001). *Computer Science: A Structured Programming Approach Using C. 2nd edition*. Brooks/Cole.
- Frei, P., Su, V., Mikhak, B., and Ishii, H. (2000). curlybot: designing a new class of computational toys. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 129–136, New York, NY, USA. ACM.
- Frost, D. (2007). Fourth grade computer science. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*.
- Hamner, E., Bernstein, D., Lauwers, T., Nourbakhsh, I., and DiSalvo, C. (2008a). Robot diaries: Broadening participation in the computer science pipeline

- through social technical exploration. *Proceedings of the AAAI Spring Symposium on Using AI to Motivate Greater Participation in Computer Science*.
- Hamner, E., Bernstein, D., Lauwers, T., Stubbs, K., Crowley, K., and Nourbakhsh, I. (2008b). Robot diaries interim project report: Development of a technology program for middle school girls. *CMU Tech Report CMU-RI-TR-08-25*.
- Hamner, E., Lauwers, T., and Bernstein, D. (2010). The debugging task: Evaluating a robotics design workshop. *Proceedings of the AAAI Spring Symposium on Beyond Robotics: Design and Evaluation*.
- Hoadley, C. M. (2004). Methodological alignment in design-based research. *Educational Psychologist*, 39:203–212.
- Horstmann, C. (2005). *Java Concepts. 4th edition*. John Wiley & Sons, Inc.
- Horstmann, C. (2006). *Big Java, 2nd edition*. John Wiley & Sons, Inc.
- Hylton, P. and Otoupal, W. (2005). Preparing urban secondary school students for entry into engineering and technology programs. *Proceedings of Frontiers in Education*.
- ITEA (2000). Standards for technological literacy: Content for the study of technology. [Http://www.iteaconnect.org/TAA/PDFs/xstnd.pdf](http://www.iteaconnect.org/TAA/PDFs/xstnd.pdf).
- Kalra, N., Lauwers, T., and Dias, M. (2007a). A braille writing tutor to combat illiteracy in developing communities. *AI in ICT for Development Workshop, International Joint Conference on Artificial Intelligence*.
- Kalra, N., Lauwers, T., Stepleton, T., Dewey, D., and Dias, M. (2007b). Iterative design of a braille writing tutor to combat illiteracy. *Proceedings of the conference on International Conference on Information and Communication Technologies and Development*.
- Kim, H. J., Coluntino, D., Martin, F. G., Silka, L., and Yanco, H. A. (2007). Artbotics: community-based collaborative art and technology education. In *SIGGRAPH '07: ACM SIGGRAPH 2007 educators program*, page 6, New York, NY, USA. ACM.
- Koczor, M. L. (1984). *Effects of varying degrees of instructional alignment in post treatment tests on mastery learning tasks of fourth-grade children*. PhD thesis, University of San Francisco.

BIBLIOGRAPHY

- Koedinger, K., Anderson, J., Hadley, W., and Mark, M. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 9:30–43.
- Krathwohl, D. R., Bloom, B. S., and Masia, B. B. (1973). *Taxonomy of Educational Objectives, the Classification of Educational Goals. Handbook II: Affective Domain*. David McKay Co Inc.
- Lahiri, A., Chattopadhyay, S., and Basu, A. (2005). Sparsha : A comprehensive indian language toolset for the blind. *Proceedings of the 7th International ACM SIGACCESS conference on Computers and Accessibility*, pages 114–120.
- Lauwers, T. and Nourbakhsh, I. (2007). Informing curricular design by surveying cs1 educators. In *Proceedings of the 4th International Symposium on Autonomous Minirobots for Research and Edutainment*.
- Lauwers, T., Nourbakhsh, I., and Hamner, E. (2009). Csbots: design and deployment of a robot designed for the cs1 classroom. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, pages 428–432.
- Lauwers, T., Nourbakhsh, I., and Hamner, E. (2010). A strategy for collaborative outreach: Lessons from the csbots project. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*.
- Lego (2009). Mindstorms nxt website. [Http://mindstorms.lego.com/](http://mindstorms.lego.com/).
- LeGrand, R., Machulis, K., D., M., Sargent, R., and Wright, A. (2005). The xbc: a modern low-cost mobile robot controller. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3896–3900.
- Lewis, J. and Loftus, W. (2005). *Java Software Solutions: Foundations of Program Design. 4th edition*. Addison Wesley, Pearson Education, Inc.
- Malik, D. (2004). *C++ Programming: From Problem Analysis to Program Design. 2nd edition*. Course Technology, Thomson Learning.
- Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., , and Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2008)*.
- Martin, F. (1988). *Children, Cybernetics, and Programmable Turtles, master's degree thesis*. MIT Media Lab.

BIBLIOGRAPHY

- Martin, F. (1994). *Circuits to Control: Learning Engineering by Designing Lego Robots*. PhD thesis, MIT.
- Martin, F. (2000). *Robotic Explorations: A Hands-on Introduction to Engineering*. Prentice Hall.
- McCauley, R. and Manaris, B. (2002). Computer science education at the start of the 21st century - a survey of accredited programs. In *Proceedings of 32nd ASEE/IEEE Frontiers in Education Conference*.
- McNally, M. (2006). Walking the grid: Robotics in cs2. In *Proceedings of the 8th Australian conference on Computing education*, pages 151–155.
- Melchior, A., Cohen, F., Cutter, T., and Leavitt, T. (2005). More than robots: An evaluation of the first robotics competition participant and institutional impacts. *Heller School for Social Policy and Management, Brandeis University*.
- Miller, D. P. and Stein, C. (2000). so that’s what pi is for! and other educational epiphanies from hands-on robotics. pages 219–243.
- Mitchell, F. M. (1999). All students can learn: Effects of curriculum alignment on the mathematics achievement of third-grade students. In *Paper presented at the Annual Meeting of the American Educational Research Association*.
- Morris, H. H. and Lee, P. (2004). The incredibly shrinking pipeline is not just for women anymore. *Computing Research News*.
- Mostow, J., Roth, S., Hauptmann, A. G., and Kane, M. (1994). A prototype reading coach that listens. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*.
- Norman, D. and Draper, S. (1986). *User Centered System Design*. L. Erlbaum and Assoc.
- Nourbakhsh, I., Crowley, K., Wilkinson, K., and Hamner, E. (2003). The educational impact of the robotic autonomy mobile robotics course. Technical Report CMU-RI-TR-03-29, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
- Nourbakhsh, I., Hamner, E., Lauwers, T., DiSalvo, C. F., and Bernstein, D. (2007). Terk: A flexible tool for science and technology education. In *Proceedings of AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*.

BIBLIOGRAPHY

- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.
- Parallax (2009). About the basic stamp. [Http://www.parallax.com/tabid/-295/Default.aspx](http://www.parallax.com/tabid/-295/Default.aspx).
- Peaco, F. L. (2004). Braille literacy: Resources for instruction, writing equipment, and supplies. *NLS Reference Circulars*.
- Perkins (2006). Perkins brailler. Catalog of Products; Howe Press of the Perkins School for the Blind.
- Raffle, H. S., Parkes, A. J., and Ishii, H. (2004). Topobo: a constructive assembly system with kinetic memory. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 647–654, New York, NY, USA. ACM.
- Reshko, G., Mason, M., and Nourbakhsh, I. (2002). Rapid prototyping of small robots. Technical Report CMU-RI-TR-02-11, Robotics Institute, Pittsburgh, PA.
- Resnick, M., Martin, F., Sargent, R., and Silverman, B. (1996). Programmable bricks: Toys to think with. *IBM Systems Journal*, 35.
- Resnick, M. and Ocko, S. (1991). *LEGO/Logo: Learning Through and About Design*. Ablex Publishing.
- Rittle-Johnson, B. and Koedinger, K. R. (2001). Using cognitive models to guide instructional design: The case of fraction division. *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, pages 857–862.
- Roberts, E. (1995). *The Art and Science of C: A Library-Based Introduction to Computer Science*. Addison Wesley.
- Rowe, A., Rosenberg, C., and Nourbakhsh, I. (2002). A low cost embedded color vision system. In *Proceedings of IROS 2002*.
- Rusk, N., M., R., Berg, R., and Pezalla-Granlund, M. (2008). New pathways into robotics: Strategies for broadening participation. *Journal of Science Education and Technology*, 17:59–69.
- Saettler, P. (1990). *The Evolution of American Educational Technology*. Libraries Unlimited.
- Sandoval, W. A. (2004). Developing learning theory by refining conjectures embodied in educational designs. *Educational Psychologist*, 39:213–223.

- Sandoval, W. A. and Bell, P. (2004). Design-based research methods for studying learning in context: Introduction. *Educational Psychologist*, 39:199–201.
- Savitch, W. (2006). *Absolute Java. 2nd edition*. Addison Wesley.
- Schon, D. A. (1984). *The Reflective Practitioner: How Professionals Think In Action*. Basic Books.
- Schroeder, F. (1989). Literacy: The key to opportunity. *Journal of Visual Impairment and Blindness*, pages 290–293.
- Schuler, D. and Namioka, A. (1993). *Participatory design: Principles and practices*. Lawrence Erlbaum.
- Schweikardt, E. and Gross, M. D. (2006). roblocks: a robotic construction kit for mathematics and science education. In *ICMI '06: Proceedings of the 8th international conference on Multimodal interfaces*, pages 72–75, New York, NY, USA. ACM.
- Schweikardt, E. and Gross, M. D. (2008). The robot is the program: interacting with roblocks. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 167–168, New York, NY, USA. ACM.
- Shamlian, S., Killfoile, K., Kellogg, R., and Duvallet, F. (2006). Fun with robots: A student-taught undergraduate robotics course. In *ICRA*, pages 369–374.
- Sierra, K. and Bates, B. (2005). *Head First Java. 2nd edition*. OReilly.
- Simon, H. (1957). *Models of Man*. John Wiley.
- Simon, H. (1996). *The Sciences of the Artificial - 3rd Edition*. MIT Press.
- Soloway, E., Guzdial, M., and Hay, K. E. (1994). Learner-centered design: the challenge for hci in the 21st century. *Interactions*, 1(2).
- Soloway, E., Jackson, S. L., Klein, J., Quintana, C., Reed, J., Spitulnik, J., Stratford, S. J., Studer, S., Jul, S., Eng, J., and Scala, N. (1996). Learning theory in practice: Case studies of learner-centered design. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 189–196, New York, NY, USA. ACM.
- Tabak, I. (2004). Reconstructing context: Negotiating the tension between exogenous and endogenous educational design. *Educational Psychologist*, 39:225–233.

BIBLIOGRAPHY

- Tallarico, I. (1984). *Effects of ecological factors on elementary school student performance on norm referenced standardized tests: Nonreading behaviors*. PhD thesis, University of San Francisco.
- Tallyn, E., Stanton, D., Benford, S., Rowland, D., Kirk, D., and Paxton, M. (2004). Introducing escience to the classroom. In *Proceedings of the UK e-Science All Hands Meeting*, pages 1027–1029.
- Vegso, J. (2005). Interest in cs as a major drops among incoming freshmen. *Computing Research News*.
- Vegso, J. (2006). Drop in cs bachelors degree production. *Computing Research News*.
- Vernier (2009). Vernier website. [Http://www.vernier.com/](http://www.vernier.com/).
- Weber, G. and Brusilovsky, P. (2001). Elm-art: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education*, 12(4):351–384.
- Wiggins, G. and McTighe, J. (2005). *Understanding by Design, Expanded 2nd Edition*. Prentice Hall.
- World Health Organization (2004). Fact sheet 282: Magnitude and causes of visual impairment.
- Yim, M., Chow, M. D., and Dunbar, W. L. (2000). Eat, sleep, robotics. pages 245–292.
- Zweben, S. (2005). 2003-2004 taulbee survey: Record ph.d. production on the horizon; undergraduate enrollments continue in decline. *Computing Research News*.