

MATERIAL DESIGN FOR A ROBOTIC ARTS STUDIO

by

Casey Wayne Smith

B.S. Physics
Montana State University
Bozeman, MT
1999

SUBMITTED TO THE PROGRAM IN MEDIA ARTS & SCIENCES, SCHOOL OF
ARCHITECTURE & PLANNING, IN PARTIAL FULLFILLMENT OF THE
REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE
IN
MEDIA ARTS AND SCIENCES
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2002

© Massachusetts Institute of Technology 2002
All Rights Reserved

Author - Casey Smith
Program in Media Arts & Sciences
May 10th, 2002

Certified by - Mitchel Resnick
Associate Professor of Learning Research
Thesis Supervisor

Accepted by - Andrew B. Lippman
Chairperson
Departmental Committee on Graduate Students

MATERIAL DESIGN FOR A ROBOTIC ARTS STUDIO

by

Casey Wayne Smith

SUBMITTED TO THE PROGRAM IN MEDIA ARTS & SCIENCES, SCHOOL OF
ARCHITECTURE & PLANNING, ON MAY 10TH, 2002, IN PARTIAL FULLFILLMENT OF THE
REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE
IN
MEDIA ARTS AND SCIENCES
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ABSTRACT

A growing number of artists are using new electronic and computational technologies for the creation of interactive, kinetic, and behavior-based art. However, users without technical backgrounds often find that there is no simple way to begin creating with these new materials without first learning a wide range of programming and electronic skills. This thesis discusses a set of technologies and activities designed for an introductory robotic art course that enable art students with little technical background to experiment with computation as a medium. The thesis presents case studies to highlight how students engaged with these technologies and discusses how the ideas represented in the course make possible a new model for artist/engineer collaboration.

Thesis Supervisor: Mitchel Resnick

Title: Associate Professor, MIT Media Lab

This work was supported in part by:

The LEGO Company

The National Science Foundation (NSF Grant # ESI-0087813)

IBM

Learning Lab Denmark

Things That Think consortium, MIT Media Lab

Digital Life consortium, MIT Media Lab

MATERIAL DESIGN FOR A ROBOTIC ARTS STUDIO

by

Casey Wayne Smith

Thesis Committee

*Thesis Advisor - **Mitchel Resnick***
Associate Professor, Lifelong Kindergarten Group
MIT Media Lab

*Thesis Reader - **Bakhtiar Mikhak***
Research Scientist, Learning Webs Group
MIT Media Lab

*Thesis Reader - **Chris Csikszentmihályi***
Assistant Professor, Computing Culture Group
MIT Media Lab

*Thesis Reader - **Arthur Ganson***
Artist

Acknowledgements

I would like to thank Mitchel Resnick for the opportunity to study under him at the Media Laboratory. I am extremely grateful for this experience and am honored have been a member of his group.

Bakhtiar Mikhak has been an enormous influence in my own educational and personal philosophy and will continue to influence my direction in the future. I would also like to thank his gracious family for their support and friendship.

Gretchen Skogerson deserves much credit for the creation of this research project and her constant energy, humor, and always interesting art.

Thanks to Lisa Fraser for everything she has done for my life.

Mark Givens, Joe Robb, and Alexis Foreman deserve credit for keeping me grounded in the important things in life.

My family continues to amaze me with their constant love and support. Only now am I beginning to fully recognize and appreciate the things that they have done for me and the ideals they stand for.

Finally, to Rahul Bhargava, Daniel Kornhauser, Tim Hirzel, and Nell Bryer for being the best friends an Ugly Duckling could ever have.

Table of Contents

Chapter One - Introduction	6
1.1 Motivation	6
1.2 Problems with Artist/Engineer Collaborations	8
1.3 Approaching Materials at Multiple Levels	10
1.4 Changing Levels	13
Chapter Two - Extended Example	15
Chapter Three - Background and Context	18
3.1 Origins and History of Robotic Art	18
3.1.1 Foundational Work	18
3.1.2 Contemporary Work	20
3.2 Constructionism and Constructionist Tools	21
3.3 Discipline-Based Art Education	23
3.4 Successful Scaffolding in Practice	24
3.4.1 6.270 (Electrical Engineering and Computer Science Dept.), MIT	24
3.4.2 Robotic Design Studio (Wellesley College)	26
3.4.3 Design By Numbers (ACG, MIT Media Lab)	27
3.4.4 Digitally Mediated Design (Architecture Department, MIT)	28
Chapter Four - Design of the Tools and Curriculum	30
4.1 Overview	30
4.2 Tools	31
4.2.1 The Cricket System	32
4.2.2 Extensions to the Cricket Toolkit	33
4.2.3 The Bus Device To Think With	35
4.2.3 The Bus Device to Think With in Use	38
4.3 Curriculum and Structure	40
Chapter Five - Evaluation	44
5.1 Evaluation Methodology	44
5.2 Aspects of Student Engagement	44
5.2.1 Experimentation and Iteration	45
5.2.2 Conceptual Engagement	47
5.2.3 Entry Point	49
5.2.4 Materiality	51
5.2.5 Extensible	53
5.3 What Didn't Work	54
Chapter Six - Conclusion	57
References	59
Appendix A	60
Appendix B	80

Chapter One - Introduction

1.1 Motivation

Educators of the studio arts recognize the value of the creation of objects. Not only does this process serve to develop technical skills, but the object created serves as an artifact to anchor discussion regarding art history, critique, aesthetics, and the artist's individual process and approach regarding the concepts and materials of their work. This approach to art pedagogy forms the basis for most studio arts courses. However, certain materials demand a level of technical ability that makes this approach difficult as the acquisition of these skills overshadows the creation of art.

Many courses designed to serve as an entry point into the field of robotic art¹ borrow their tools and curriculum directly from the engineering context in which these materials are most often found. The complexity of the materials of this field, namely electronics and programming, demand that many technical skills are needed before students can begin creating. As a result, the approach of these courses is often centered on problem solving and the acquisition of technical skills needed to work with the materials.

1. The name 'robotic art' may be problematic. It is often used to describe art whose conceptual premise is robotics. I use it more broadly, to describe art that incorporates the materials of robotics (microcontrollers, sensors, motors, etc.) but doesn't necessarily use it as its conceptual basis. My use encompasses interactive sculpture, behavioral sculpture, and the more typical definition of robotic art. Eduardo Kac loosely defines the field by 'the principle of giving precedence to behaviour over form.' (Kac, 2001a)

While these skills are certainly necessary for success in the field of robotic art, I will argue that with properly designed tools and curriculum a course can be created that better matches the traditions of studio arts education. Such a course will allow novices to immediately create and experiment with these materials, acquiring technical skills in the process. The objects created in this course can then be used to ground reflection and discussion of the unique aspects of this field of art. I developed such a course, titled *An Introduction to Robotic Art*, and co-taught it at the Massachusetts College of Art (MassArt) in the fall of 2001. This thesis documents the tools, curriculum, approach, and results of that course.

This course relies on technological tools that temporarily hide the technical details of the materials so that users may more readily experiment and iterate with them. These tools allow novices to readily create interactive and behavioral sculpture, support increasingly complex designs as the user's knowledge grows, and develop a good foundation for continued exploration of the robotic art in the future.

It might be misconstrued that this approach is masking the technical details of technology because of a lack of faith in artists to learn them. However, quite the opposite is true. I believe that anyone, artist or engineer, can learn these skills given the right motivation. This course is intended to give people more interested in creating with these materials an opportunity to explore the expressive potential of this new media and give them motivation to learn the technical skills they will need in the future.

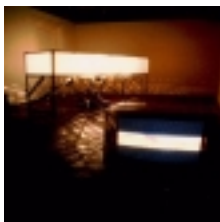


Figure 1: Gretchen Skogerson's *Lash..*

In the fall of 2001, I began helping a fine arts student at Rensselaer Polytechnic Institute, Gretchen Skogerson, learn to use Media Lab's current Programmable Brick technology (Resnick, Martin, Sargent, Silverman 1996), the Cricket (Martin et al., 2000). The Cricket is a small micro-controller that allows the user to author simple programs using sensors, infra-

red communication, and motor control as primitives. Having basic programming skills already, Gretchen took immediately to Crickets, eventually using them in her Master of Fine Arts thesis project, *Lash* (figure 1.1). The process was rewarding for both of us; she improved her programming skills and I gained interest in the field of robotic art. One year later, Gretchen accepted a faculty position at the Massachusetts College of Art. Through this collaboration the ideas represented here were developed. Gretchen co-taught *An Introduction to Robotic Art* with me, bringing her considerable knowledge of the field to the course.

1.2 Problems with Artist/Engineer Collaborations

As computer based art grows, art schools are offering more computer-related classes. As the community of practitioners grows, so will the sophistication of technical, critical, and aesthetic discourses in the field. Filmmaker and software engineer Andrew Stern explains this trend:

"There is no escaping the fact that to make an artwork interactive is fundamentally to build a machine with processes; anything less would simply be a reactive work without autonomy- 'push button' art. Artists must think procedurally to create truly interactive art, and fashion these procedures to express their artistic intentions. This requires the artist to have a firm foothold in both artistic practice and computer science. The most precocious 'new media' academic departments are requiring their students to become equally proficient in both disciplines" (Stern, 2001).

One approach for artists wanting to use technological materials but not wanting to spend the time learning their use is to collaborate with a technologist. The Experiments in Art and Technology (E.A.T) foundation, in fact, was established to match technologists with artists to meet this need. However, in complex applications of technology the technologist must make many decisions that will ultimately affect the outcome of the work.

In these instances “the programmer becomes less of a technician and more of a fellow artist” (Stern, 2001). The artist is then in danger of losing his vision of the project. However, collaborations can lead to interesting art but also to a better understanding of the medium for both artist and technologist. In the best possible case, the technologist has an understanding of art and the artist has an understanding of technology, leading to a shared language and process and perhaps to a new type of knowledge. This knowledge can be formed by the technologist and artist each practicing in the medium and process of the other.

However, these relationships can be problematic. MIT professor John Maeda explains:

"Although such collaborations can produce respectable artwork, they rarely lead to works of real power and inspiration. What is more, the situation is getting worse because relentless progress in information technology has widened the gap between artist and engineer: The artist has little understanding of the computer as a medium, and the engineer (who has no artistic training) is not allowed to unlock his creative potential in using the medium he has mastered" (Maeda, 1998).

Casey Reas, a graduate of Maeda's Aesthetics and Computation Group at MIT, is an artist by training and self-taught in many skills of the engineering domain. He calls for ways to make technology more approachable by artists:

"[U]ntil the technologies required for the production of this art become as ubiquitous as video cameras, there will not be a generation of artists who have an intuitive understanding of this medium. It is also necessary to develop tools to enable the manipulation of the raw materials at a higher level than the current practice of intricately machining custom parts and writing programs in C and assembly code¹" (Reas, 2001).

Regardless of whether the artist chooses to work with an engineer or learn the necessary skills themselves, the success of their work depends on their understanding of the computer as a medium. This understanding only comes with the experience of using it artistically.

1.3 Approaching Materials at Multiple Levels

Most students wanting to gain experience with computers and electronics must learn the necessary skills on their own or enroll in an engineering course that covers these topics. The "skills-based approach" found in most engineering courses may be favorable to students already familiar with the capabilities and limitations of the materials, and possessing a clear idea of the types of work they wish to create. They have the proper motivation to benefit greatly from a course of this nature. Without this background, students may find themselves wondering if the things they are learning are relevant to the creation of art. How the circuit they just built or program they wrote can be used expressively? Even if the course is rooted in art, which many good educators in this field do, students will likely not have the time to build interesting objects as they must move on to the next, harder concept. In these courses, the dominant process is often one of problem solving and debugging as the goal of a specific activity is a working circuit or program.

One difficulty of learning a skill very far outside of a learner's domain of knowledge is known as 'Schön's paradox.' It is summarized as follows:

-
1. The C language and assembly code are two very common computer programming languages.

" The paradox of learning a really new competence is this: that a student cannot at first really understand what he needs to learn, can only learn it by educating himself, and can only educate himself by beginning to do what he does not yet understand" (Schön, 1987).

As electronics and programming are often outside of an art student's domain of expertise, an introductory course in these subjects should serve to allow students to explore the potential of the field while providing them with an understanding of what they would need to learn to pursue what they want to build. Only when students understand what they want to pursue (the art they want to build) and what they need to learn should the learning of technical skills be the priority.

The act of designing is a "kind of experimentation that consists in reflective 'conversation' with the materials of a design situation." (Schön, 1992 p.135) In robotic art, designing then requires experimentation and iteration with electronics and microcontrollers¹, materials that do not necessarily lend themselves well to this approach as electronics usually only work or do not work, and while interesting accidents can happen due to 'logical' mistakes, more often a program will simply not work at all due to syntactical mistakes. Experimentation is often not rewarded and iteration is made difficult by the long revision times often needed for electronics and microcontroller programming. Designing is often a matter of appreciating and using qualities of materials as they emerge (Schön, 1992 p.138), but this

1. Microcontrollers contain a central processing unit combined with peripherals such as input/output (I/O) devices, memory, and timing references. They are packaged on a single chip and can be programmed by the user with a set of instructions loaded into their memory.

approach to design is limited to materials that readily support experimentation or to people already fluent in the use of the materials.

This challenge can be overcome by offering people many different levels at which they engage with the material. This is particularly important with complex materials such as electronics and computers. For instance, some users may want control over the lower-level details of the system and are willing to learn the technical skills needed to have control over these details. Others might be more interested in exploring the potential of electronics and programming in art. These users should engage with the materials at a higher level.



Figure 1.2: The Cricket.

This thesis documents a course designed to provide a higher-level entry point into the materials of robotic art. While many courses exist that focus on the lower-level, technical skills needed to manipulate and build with these materials, such as Carnegie Mellon University's *Robotic Art Studio* and New York University's *Physical Computing*, this course hinges on a toolkit that *scaffolds* the technical details of electronics and programming. Scaffolding, in this instance, refers to the masking of certain details of the materials from users so that they can explore more relevant aspects. In this way, "students have a chance to apply a set of skills in constructing an interesting problem solution before they are required to generate or remember those skills" (Collins, Brown, Newman, 1989 p.485). The Cricket System (*figure 1.2*) previously developed at the MIT Media Lab serves as the base kit for this course, although I created many new devices specifi-



Figure 1.3: A Cricket with Display Bus Device and Distance Bus Device.

cally for the creation of large-scale and interactive artwork. This system requires no electronics or programming knowledge to begin using. Creations built with it are easily changed and modified. Syntax is minimized so that when accidents occur they are more often logical errors that might lead to something interesting than syntactical errors that cause a program not to run. The Cricket has the ability to communicate with other special devices, known as Bus Devices, that extends its capabilities. For example, by plugging in the Display Bus Device into the Cricket a user can display sensor values or other numbers (*figure 1.3*). More details of the Cricket and its Bus Devices will be discussed later.



Figure 1.4: The Bus Device to Think With.

1.4 Changing Levels

Whenever scaffolding is used in educational settings, one must be very clear about what is being hidden and how to unravel the scaffolding to see the underlying details. In this manner, users wishing to explore the lower level details of the system may do so. To this end, I developed the Bus Device to Think With, a Cricket Bus Device that allows the user to gain understanding of electronics, lower-level microcontroller programming, and the underlying details of the Cricket System. Through a series of activities designed around this device, students can introduce themselves to these concepts within the familiarity of the Cricket System, allowing their skills to progress to lower-level electronic and programming concepts. These skills will allow students to move on to materials that allow them to escape the constraints of the Cricket System, use the Cricket System in

unexpected and unsupported ways, or become a developer of the Cricket
System itself

Chapter Two - Extended Example

In this section, in order to better illustrate the nature of the class, I will discuss the experience of one student in *An Introduction to Robotic Art*, Jen. Jen is a visiting lecturer at MassArt and an accomplished artist. The course was designed for undergraduates at MassArt, so while Jen does not represent a typical student in the class, I feel that her experience represents the ideas of the course well.



Figure 2.1: Bronze acorns made by Jen. It was modeled and unfolded in 3D software, cut on flat bronze, then re-folded by hand.

Jen works mostly with metal, specifically bronze, but in the last 3 years has begun to experiment with a new form of sculpture that has arisen due to advances in 3D modeling and printing technology (*figure 2.1*). In this field, sculpture is designed on a computer using 3D modeling software then 'printed', layer-by-layer, in plastic using a machine known as a 3D printer. While Jen almost failed a programming course as an undergraduate and claims to have "sworn off" computers since, she says that the visual nature of 3D modeling attracted her to the field: "The software has become so visual, so user friendly, that I've had renewed interest and successes; a smoother learning experience." Jen says that she has become "bored with [her] perceived limitations of the medium" and found that the sculpture was more interesting in terms of the process that she used to create it than as an object itself. She desired to "integrate the computer part of the work more directly with the studio work." This was her motivation for taking *An Introduction to Robotic Art*.



Figure 2.2: *Ludicrum*
v.1.



Figure 2.3: *Ludicrum*
v.3.

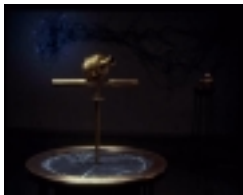


Figure 2.4: *Ludicrum*
v. 2.

Jen spent most of the course time writing programs for the Cricket. Each new Bus Device that I brought in or new concept that we discussed prompted Jen to write new programs to explore their use. Gradually, Jen's programming became more directly, focusing on stepper motors, optical distance sensors, and LED controllers. While I knew that she had begun to integrate these things into her metal work, I was very surprised when Jen told me of her upcoming exhibit of these pieces at the Montserrat College of Art! The show, titled *Ludicrum: naturalia, artificialia, scientifica v.1-5*, was a series of five different works, all relating to the presentation of the natural world of insects that her young son is fascinated by but often unnoticed by adults. Three of the five contained Crickets: Two of the pieces consisted of brass microscopes and rotary tables, on which different insects and objects were mounted. These pieces used optical distance sensors to detect a viewer and reacted by presenting them with different objects to view through the microscopes by rotating the table with a stepper motor (*figures 2.2 & 2.3*). The third piece consisted of a brass telescope with a large group of LEDs mounted on a far wall that invoked the image of a swarm of insects. As the viewer looked through the telescope, the LEDs would become active and display different patterns (*figure 2.4*).

Jen is currently using the Bus Device to Think With to further her knowledge and skills in microcontroller programming and electronics. As these skills develop, she hopes to build projects with more advanced microcontrollers and possibly re-design the LED Bus Device she used in *Ludicrum* to better meet her needs. She also plans on continuing to use the Cricket in her work. She summarized the experience of the course in a recent letter to me:

"I've seen a whole lot of really uninteresting artwork made under the umbrella of the digital, and I think this is because people get so caught up in the technology, the technical details, that the artistry suffers. Or perhaps I am less

impressed by technical feats than I am with a beautifully drawn simple black line.

"What impresses me about the Cricket, and the effort to make it somewhat user friendly, is that the device bridged a gap for me; the gap between the physical object, and the computer. So much of the programming involves a tactile experience. I mean, once you write your program, then you immediately go to the Cricket and turn it on, plug it in, download, and see if it works. I know I would have been scared off of the idea of using sensors and LEDs at first if I had had to have a good grasp of $V = I \cdot R$ ¹. Yeah, after some hands-on experimenting at the end of th class I see how it works, but I think having to learn minutiae about electronics would have been a big turn off. Not that I'm not interested, but my brain gets hit with a paralyzing wave of anxiety when overwhelmed by the abstract. Having the hands-on devices to play with allowed the trickier concepts to have a visual aid, and allowed me to understand some things at a superficial level before I really understood [the technical details] as far as I understand [them] right now."

1. $V = I \cdot R$, *voltage is equal to current multiplied by resistance*, is a canonical equation in electronics. It is often the first major topic presented in an introductory electronics course and forms the basis for much of the later topics.

Chapter Three - Background and Context

3.1 Origins and History of Robotic Art

This section describes the origins and history of robotic art. It is provided to familiarize the reader with the concepts, materials, and techniques unique to this form of art. After briefly discussing its origins, I will provide selected foundational and contemporary works. More detailed accounts can be found in Jack Burnham's book *Beyond Modern Sculpture* (Burnham, 1967) or in Eduardo Kac's article *Towards a Chronology of Robotic Art* (Kac, 2001b).

3.1.1 Foundational Work

In 1955, a show titled *Le Mouvement* opened at the Galerie Denise Rene in Paris. It was one of the first events showcasing movement as a fundamental means of expression. Marcel Duchamp and Alexander Calder, artists known for their early experiments in motion, were featured in this show. In 1966, Bell Labs physicist Billy Klüver organized *9 Evenings*, a public exhibit of collaborative performances created by teams of artists and engineers. In 1968, Jasia Reichhart organized *Cybernetic Serendipity* at London's Institute of Contemporary Arts. This show featured art from Nam June Paik, Nicholas Schöffer, and Jean Tinguely, among many others.

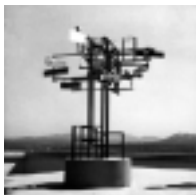


Figure 3.1: *CYSP 1*.

CYSP 1 - Nicholas Schöffer (1956)

Kinetic art, a field that started and grew rapidly starting in the 1950's, attempted to incorporate movement into what was previously static sculp-

ture and reintroduced technology into the debate. In 1956, Nicholas Schöffer built *CYSP 1* (Cybernetic Spatiodynamic Sculpture) (*figure 3.1*). The sculpture, held by a stationary base, contained sensors and analog electronics that produced different types of movements in response to the presence of viewers. The complexity of the intelligence or behavior was a landmark for the field, as the only previous work was mechanically controlled. This work is often regarded as the first example of cybernetic art. This work was made possible through the large technical and monetary support of the Philips Corporation.



Figure 3.2: *K-456*.

K-456 - Nam June Paik and Shuya Abe (1964)

A 21 channel, remotely controlled robot, *K-456* had rough human form, recited JFK's inaugural address, and excreted beans as it was guided through its performance space (*figure 3.2*). While intended to be a political piece, this work raised further questions in reactivity, remote control and audience interaction. It parodied the popular fear of robots displacing workers as it required constant attention by a large number of people. This work was also the result of a collaboration between an artist (Paik) and engineer (Abe).



Figure 3.3: *The Senster*.

The Senster – Edward Ihnatowicz (1970)

This large creature exhibited a very shy behavior, nodding its head to quieter viewers while moving away from louder ones (*figure 3.3*). The piece was commissioned by the Phillips Corporation and utilized a Philips digital microcomputer. Many different sensors were used, including microphones and motion-detectors, and the body contained six hydraulic servos to actuate its limbs. Its sensual behavior contrasted its stark appearance and provided the first example of behavior created by a computer in robotic art.

Ménage - Norman White (1974)



Figure 3.4: *Ménage*.

In 1974, Norman White created four ceiling-mounted robots and a fifth robot on the floor. Each robot contained a light source and light scanner. The light scanning apparatus would point itself in the direction of light sources and as a result the robots would point at one another (*figure 3.4*). However, more complex movements occurred as the robots were pulled away from each other by their own motors. The seemingly simple organization created very dynamic interactions, providing an early example one of emergent behavior in robotic art.



Figure 3.5: *Flock*.

3.1.2 Contemporary Work

Flock - Kenneth Rinaldo (1993)

The Flock consists of three robots that communicate their individual positions via audible telephone tones (*figure 3.4*). Each robot reacts to the presences of viewers but also to the positions and movements of the other robots. As a result, emergent and self-organizing behaviors are observed. Complex and organic motions are seen as well. This piece alludes to a collective intelligence posed by the robots that is reinforced by the audible language with which they communicated.



Figure 3.6: *Petite Mal*.

Petite Mal - Simon Penny (1993)

Petite Mal is an autonomous and mobile robot (*figure 3.6*). It contains ultrasonic and body-heat sensors that allow it to detect humans and sometimes pursue them. It contains no non-functional parts, save a flowered tablecloth that wraps its metal beams, giving it a primitive aesthetic. It gets its mobility from a pair of motor-driven bicycle wheels. It is meant to portray an intelligent machine but not an anthropomorphic one. This is achieved through its bare appearance and shape that does not resemble human nor animal form.

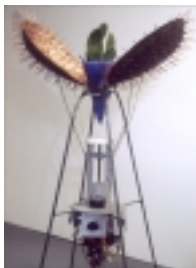


Figure 3.7: *Rearming the Spineless Opuntia*.

Rearming the Spineless Opuntia - Amy Young (1999)

This work attempts to give the opuntia plant back its spiny armor that evolution took from it many years ago (*figure 3.7*). Sonar sensors detect an approaching human, and raises large spiny leaves to shield the plant from the person approaching. As the person steps back, the shields lower again. This work ironically places technology in the position of a defense for nature, where it is usually seen as an attacker.



Figure 3.8: *Rover*.

Rover - Alan Rath (1998)

Rover is a small, wheeled robot that sits quietly on a recharging dock until it senses activity (*figure 3.8*). It then seeks out visitors and inspects them with its video camera. While sometimes comical, it can also seem intrusive or menacing. David Ebony of *Art in America* describes Rath's work as, "not aggressive monsters, nor are they passive or subservient beasts. While their movements hint at human behavior and social interactions, they are not anthropomorphic" (Rath, 1999 p.43).

3.2 Constructionism and Constructionist Tools

Constructionism is an epistemological theory that builds on Piaget's Constructivism (Piaget, 1963) by adding the idea that knowledge is constructed "especially felicitously" while engaged in the creation of a personally meaningful or public object (Papert and Harel, 1991). This idea places faith in people's ability to learn skills and ideas in natural ways given the right motivation, environment, and materials. "Better learning will not come from finding better ways for the teacher to instruct but from giving the learner better opportunities to construct" (Papert, 1990, p.3). The challenge for educators is then developing tools, materials, and activities to support these opportunities.

One project intended to extend what children can build and what ideas can be learned in the process is the Programmable Bricks project at the MIT Media Lab (Resnick et al., 1996). The Programmable Brick is a small computer programmed on a desktop, but then free of the desktop in all ways. It can read sensors, collect and store data, and control small motors. The Programmable Brick allows children to build a wide variety of creations from a rollerblade odometer to an automated hamster cage (Resnick et al., 2000).

While this project has foundations that are almost two decades old, the Media Lab continues to improve the Programmable Brick. The Cricket, the latest Programmable Brick, is much smaller than previous versions (about the size of a 9V battery), has built-in infrared communication support, and is extensible via an open bus line. Many ‘extension devices’, once known as X-Gadgets but now most commonly called ‘Bus Devices’, enable the Cricket with new functionality to support specific purposes or activities. Examples of Bus Devices are radio frequency communication devices, four-digit number displays, distance sensors, and LED controllers.



Figure 3.9: The LogoBlocks programming environment.

The Cricket is programmed in a number of environments. The easiest is LogoBlocks (*figure 3.9*), a graphical programming environment that allows the user to snap together primitives and control structures as if they were LEGO bricks (Begel, 1996). While this language is very simple to use, the more advanced user might find constructing programs slower than typing them and large programs hard to understand and manipulate. A few text-based environments exist, the newest being the Jackal programming environment, created by Rahul Bhargava (*figure 3.10*). Jackal simplifies the use of Bus Devices in procedures and allows advanced users to add their own primitives to those the Cricket ‘knows’. As the blocks in LogoBlocks are marked with their text representation, the transition from LogoB-



Figure 3.10: The Jackal programming environment.

locks to a text-based environment is easy for most users and the syntax highlighting of Jackal helps further.

While Constructionism is often a radical departure from pedagogy embraced by primary schools today, it might not seem radical at all within an arts context, as most studio courses are structured in exactly that manner. In fact, one of the original goals of Seymour Papert's use of Logo in schools was to make math classrooms look more like art classrooms (Papert, 1980, Papert and Harel, 1991). However, the success of the studio method in art is often a result of the materials; they simply lend themselves well to experimentation within the context of the activity. An Introduction to Robotic Art is an attempt to bring those qualities to the materials of programming and electronics.

3.3 Discipline-Based Art Education

The Discipline-Based Arts Education (DBAE) approach to studio art courses incorporates art history, art criticism, and aesthetics into a studio arts course (Smith, 1987). While this approach to studio arts has long, informal traditions, the Discipline-Based Arts Education pedagogy was formalized in the 1960's for many different reasons. One goal was to provide a structured way for teachers to go beyond the skills approach and allow the studio to introduce discussions of art history, critique, and aesthetics. It also helped schools justify arts programs and secure funding, as this approach allowed for testable results and core curriculum (Anderson, 1992). This unfortunate byproduct has resulted in a much more rigid approach to DBAE, forcing time allocations for each domain and demanding very specific, testable subjects covered in a systematic way. The central focus of artmaking as means of understanding art is also lost, instead replaced with compartmentalized and isolated sections focusing on history,

critique, aesthetics, and technical skills. However the instances of DBAE may fail, the foundations of DBAE are still admirable: The creation of art can anchor and promote discussion of art conceptually, aesthetically, and historically. This idea serves as the basis for *An Introduction to Robotic Art*.

3.4 Successful Scaffolding in Practice

In this section, I will summarize four courses that utilize technology for learning and expression. These courses are important to my approach as they depend on scaffolding to emphasize process and conceptual development through the construction of meaningful and expressive objects. Although very different from each other, each course described here has influenced the development of *An Introduction to Robotic Art*.

3.4.1 6.270 (Electrical Engineering and Computer Science Dept.), MIT 6.270 (The Robot Design Competition) is a one month long, intensive robot design course originally organized by MIT undergraduate Michael Parker but further developed by Fred Martin and Randy Sargent. It is offered during MIT's January Independent Activity Period. The course attempts to provide authentic design experience in a curriculum that has become increasingly focused on mathematics and theory and less on design (Martin, 1994). Martin thought the existing design courses within the Course 6 curriculum focused on learning about design and not necessarily about learning how to design. Much inspiration for 6.270 came from Professor Woody Flower's Introduction to Design course offered in the Mechanical Engineering Department, 2.70. Students are given a bag of scrap materials and specifications for a task to be performed by a machine. They then spend most of the semester building the machine, which will then compete with others in a contest at the end of the semester. Fred wanted to develop a similar course, but one that



Figure 3.11: A 6.270 fobot.

focused on *computational* design in addition to the mechanical design. Students participating in 6.270 are given a bag of materials, including batteries, motors, sensors, and LEGO pieces, and a pre-built microcontroller. They are also given documentation that describes the use of the materials and the specifications for the competition. Lectures and recitations are held to describe and discuss various ideas relevant to designing, building, and programming mobile, autonomous robots (*figure 3.11*). However, Martin considers the most important learning aspect of the course to be the work that happens in small groups, as students build, debug, and redesign their robots (Martin, 1994).

Martin developed special technology for 6.270. This development was based on three criteria:

“ - *Level of Abstraction* Any educational technology hides or isolates the user from certain phenomena while revealing or highlighting others. In developing tools to facilitate the design of robots, we paid special attention to the sort of technological ideas we were exposing. Since a robot is a system comprised of a variety of media- electronics, programming, and mechanics- it was necessary to be clear on which concepts we expected students to master and which others they could simply use.

-*Transparency* Even if a certain idea is encapsulated by the layer of abstraction, it should be easily accessible to students who are interested. For example, we determined that students should not need to have a deep understanding of digital electronics in order to build their robots. But we did not want to prevent or discourage students from exploring this topic as part of their robot-building. Quite the contrary, we hoped to invite them to do so through the design of our materials, while simultaneously taking pains not to intimidate students who might not be interested in this topic.

-*Interactivity* Central to our project pedagogy was the belief that people learn best by *exploring ideas in a playful manner*. This was the modus operandi of our technology development, and a key concern was creating materials that

would encourage this behavior in students” (Martin, 1994)

The technology currently used is the Handyboard, a powerful microcontroller programmed in Interactive C. Martin was also one of the chief designers of the Cricket System, and the above design criteria heavily influenced its design as well.

6.270 represents a successful use of scaffolding within the engineering domain. Martin was more concerned with allowing students to encounter design problems and conceptual approaches to engineering than the technical skills themselves, and the qualities of the materials used allowed the course to be structured to represent this focus.



Figure 3.12: *The Monkey Zipline robot from the 2002 Robotic Design Studio*



Figure 3.13: *The Whack-A-Mole robot from the 2002 Robotic Design Studio*

3.4.2 Robotic Design Studio (Wellesley College)

The Robotic Design Studio at Wellesley College was developed to give liberal arts students access to the important aspects of engineering without the emphasis on technical skills that usually accompanies engineering curriculum (Turbak & Berg, 2001). Turbak and Berg describe their motivation for offering an engineering course at a liberal arts school as fourfold:

- To provide designing and building activities that challenge students so that they are forced to reflect on their own learning and problem solving processes.
- To provide insight into the ‘big ideas’ in engineering so that students might better understand engineered systems but also social and natural systems as well.
- To have students cross disciplines and make connections between them.
- To make technology more understandable and less intimidating

The Robotic Design Studio is an explicit attempt to adapt the approach of 6.270 to a different audience. Instead of competition robots, student projects in the Robotic Design Studio typically involve storytelling and narrative robots (*figures 3.12 & 3.13*). The specifications of the final project are intentionally free of constraints, allowing for students to build

projects of interest to them. This course uses Crickets and Handyboards as the main materials, although projects usually involve many art and craft materials as well.

Similarly to 6.270, this course represents the use of scaffolding in order to allow the focus of the course to be about engineering *as a field* and not about specific techniques or skills. However, the course also shows the expressive abilities of these materials and provides examples of students from diverse backgrounds to become interested and engaged in engineering.

3.4.3 Design By Numbers (ACG, MIT Media Lab)

John Maeda and his Aesthetics and Computation Group (ACG) at the MIT Media Lab have developed a software tool and course curriculum to give graphic designers an introductory experience in computer programming. The system consists of an interpreted programming language that is simple in syntax and structure and a very small (100 by 100 pixel) design area.



Figure 3.14: The Design By Numbers programming environment.

By removing the difficulty of syntax and the long development time associated with most graphics languages, Design By Numbers (DBN) allows students to explore the techniques and constraints associated with using the computer as a design tool. Also, the constraints of the 100 pixel-per-side graphics area and 100 shades of gray color domain provide structure for their designs in what might otherwise be an overwhelming or intimidating space (*figures 3.14 & 3.15*). While the programming language is unique to DBN, programming skills learned within its confines transfer to other languages, including C, Logo, and Java. In his book, also called Design By Numbers, Maeda warns, “Do not assume that fluency in the system described in this book will guarantee an easy transition to mainstream languages. Nevertheless, you will find yourself well prepared to create in

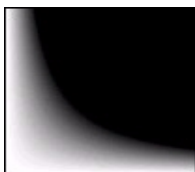


Figure 3.15: An example of a DBN program.

whichever language you ultimately choose” (Maeda, 1999). More importantly, users learn about computer-aided graphic design conceptually without the overhead of learning complex programming languages. Users interested in learning more and wanting to escape the limitations of the system will appreciate the background that Design By Numbers has given them.

Design By Numbers provides an example of using scaffolding to introduce users to the expressive abilities and constraints of the computer as a design tool without concerning themselves with the technical details of syntax or compilers. It provides a motivation for users fluent in DBN to learn other, more powerful graphics languages such as C++, OpenGL, and Java.

Design By Numbers is an excellent example of a course created to give students an opportunity to design in order to learn how to design, while providing them with general programming skills that will serve them in future.



Figure 3.16: A design project created by a student in Digitally Mediated Design.

3.4.4 Digitally Mediated Design (Architecture Department, MIT)

This course, developed by Megan Yakeley in the MIT Architecture Department, is based on the idea that “the development of ideas through iterative, experimental, incremental exploration” (Yakeley, 2000) will lead to greater understanding of the designer’s own creative process. The course depends on instructors willing to decrease the importance of the end product that is typically stressed in a studio course and also on design tools that are flexible and capable enough to allow students to iterate and experiment. It stresses the Constructionist notion that the creation of tangible artifacts that are meaningful to the creator will bring about a greater understanding of their creative and learning processes and also serve as a starting point for discussion and reflection among peers (*figure 3.16*).

This course takes advantage of the qualities that a computer brings to the design process by allowing students to engage in design in ways that they normally cannot with paper. For example, a student wanting to experiment or take a risk with a design can save the design under a new file name before altering it so that the original design is left unchanged. With paper, a student would start from scratch in the creation of the new design. Revisions to a design can be saved so that students can review the history of their design to better understand the processes they use and the influences of different sources in their design. The act of designing through programming is also significant, as it “permits the important step of drawing the students attention to the procedural nature of the design process” and focuses the students’ attention to “one small step at a time” (Yakely, 2000). It can also encourage exploration of design in new ways through the introduction of selective randomness that the computer allows.

While the tools of this course do not necessarily fit within the definition of scaffolding, his course demonstrates how the experimental and iterative abilities of a tool can affect a user’s acknowledgement and reflection of their personal process. This tenant is an important aspect *An Introduction to Robotic Art*.

Chapter Four - Design of the Tools and Curriculum

4.1 Overview

In this section I will discuss the design and implementation of both the technological tools and course curriculum and structure. These designs draw on Constructionism, concepts in robotic art, Discipline-Based Art Education, and the courses mentioned previously. While designed specifically for use within a semester-long course taught at the Massachusetts College of Art, they are applicable elsewhere and would appeal to anyone interested in learning about the creation of interactive or behavioral artwork.

An Introduction to Robotic Art is based on the creation of art as a focal point for the practice and discussion of technique, aesthetics, history, and concepts. This course uses experimentation and iteration to draw students into a reflection on their process with these materials. This is accomplished through the encouragement of prototyping and iteration, as well as by conversations with students regarding their approach to these materials versus the materials they normally create with. Technical skills, such as programming, soldering, and simple electronics, are introduced within the context of specific activities centered on the creation of art.

The Cricket System provides the scaffolding needed to allow this approach

to be successful. Below, I discuss the features of these tools and the extensions I created for these tools to better support robotic art and future learning.

4.2 Tools

I believe that the toolkit used in *An Introduction to Robotic Art* should:

- Offer a Low Entry Point: Allow students to immediately begin authoring movements, communications, interactions, and behaviors. This is important so that students are not intimidated by the materials or activities
- Enable Rapid Iteration: Allow for fluid modification in movement, interaction, and behavior to allow students to better understand the medium and to better benefit from peer review and critique. A playful approach is fostered by tools that allow for rapid iteration, flexibility, and experimentation. This feature also allows the work to be fine-tuned to meet exact desires.
- Support Interest-Centered Learning: Allow students to independently explore areas of their own interest. This describes a tool that has very different abilities that support many different possibilities. A good tool should interest many different people and support the creation of very different art. At any time, a student should be able to leave a general approach and focus in on a specific quality of the material.
- Be Extensible: Prepare students for future explorations in technology, programming, and robotic art as they move to tools with greater levels of user control. Ideally, a system would support users to de-layer the tool to reach more and more complex, and usually flexible, abilities. The metaphor of an onion is appropriate here, where students can pull off a layer as their understanding of it grows. Eventually the students might approach the inner layer of the onion, or they also may find an outer layer satisfactory to them. In any case, a good tool supports growth within it and transfer of knowledge to systems outside it.

Below I will describe the Cricket System and the extensions to it that I made to help it meet these ideals.

4.2.1 The Cricket System

The Cricket System consists of the Cricket, Cricket Bus Devices, and the programming environment. The Cricket itself is not much larger than a 9 Volt battery, contains two motor outputs, two sensor inputs, infrared input and output, and a piezo beeper that can play notes or 'beep'. Programs are written on a desktop computer, which is responsible for the compiling of the program and transferring it to the Cricket via an infrared interface connected to the serial port of the desktop. Once programmed, the Cricket no longer needs the desktop computer for operation.

As mentioned earlier, the Cricket can be programmed in two different environments, LogoBlocks and Jackal. Transferring from LogoBlocks to Jackal is made easier by using the CricketLogo keywords on the blocks of LogoBlocks, and while CricketLogo is a unique language, it provides users with necessary concepts that apply to popular programming languages such as BASIC, C, or Java. Both environments require no compiling, meaning that code can be executed as soon as it is written, providing for rapid iterations.

The Cricket has an open bus line which allows for other devices to plug into and be controlled by the Cricket. The current collection of these devices, referred to as 'Bus Devices', includes accelerometers, radio-frequency communication devices, and LCD displays. The large number of Bus Devices available for the Cricket (more than thirty have been developed) makes the Cricket System suitable for a diverse set of projects. Multiple Bus Devices can be connected to the Cricket at the same time.

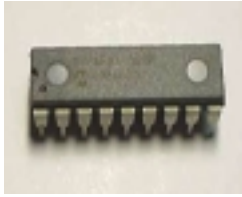


Figure 4.1: The MicroChip PIC microcontroller.

The Cricket and its Bus Devices are built with the MicroChip Peripheral Interface Controller (PIC) series of microcontrollers (*figure 4.1*). These devices, whose name still reflects their original role in industry control, are flexible and economic devices suited for a large number of tasks. The PIC on the Cricket runs a program known as the Cricket Core, which is responsible for interpreting and running the users program.

The Cricket, while meeting most of the design criteria, needed additional devices for its role in this course. First, the toolkit must be capable of creating large-scale, interactive artwork. I designed many hardware devices to meet the needs specific to robotic art. These devices are outlined below. Secondly, I will discuss a device intended to provide students an environment to explore concepts in electronics and microcontroller programming within the familiarity of the Cricket system.

4.2.2 Extensions to the Cricket Toolkit

I created the devices to better equip the Cricket system for the construction of large-scale, interactive artwork:

Relay Bus Device: The first focus for expanding the Cricket set was the ability to drive larger motors. As a first pass, a Relay Bus Device was created to allow for the use of any AC or DC motor with the Cricket (*figure 4.2*). It was set up so that the user could plug in any power supply desired and easily connect the leads of motors. Two relays were used with each motor so that the direction of DC motors could be changed. However, the lack of speed control was a drawback of this device, and the Big Motor Bus Device was created. The Relay Bus Device still proves useful for AC

motors.

Big Motor Bus Device: This device allows users to connect a DC motor and power supply (3-44V, <3Amps) and control the motors with 100 different power settings (*figure 4.3*). This ability is important for avoiding complex gearings.

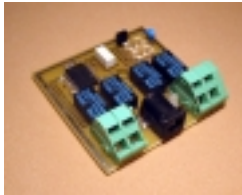


Figure 4.2: The Relay Bus Device.

Clock and Calendar: My own curiosities in art often involve concepts in time. The clock of the Long Now Foundation¹ and the work of Bruce Cannon² are two examples of art in this area that I find fascinating. I created a real time Clock and Calendar to give the Cricket the ability to create time-based pieces. It contains a small battery to keep the time while the Cricket is turned off.



Figure 4.3: The Big Motor Bus Device.

Capacitive Touch Sensor: Many interactive works depend on the ability to sense human presence. An optical distance sensor exists to detect presence at a distance; the ability to detect touch through materials was needed. Drawing on the recent release of single-chip, capacitive touch-sensing circuits, I created a simple board that gives the Cricket the ability to detect human touch through paper, plastic, or wood or on any metal object.



Figure 4.4: The Clock and Calendar Bus Device.

Stepper Motor Bus Device: Stepper motors are useful components that allow for precise stepping of the motor shaft. This is useful as it provides for precise positioning, like a servomotor, but can also turn around continuously. The Stepper Motor Bus Device allows the Cricket to control two stepper motors. It allows for the stepping speed and direction of the motor to be set, as well as discrete or continuous stepping.



Figure 4.5: The Capacitive Touch Sensor aboard a Cricket.

Multi-sensor Input Bus Device: This device allows users to take readings from 5 additional sensors. Many of the devices can be used together to allow the Cricket to use as many as 22 sensors simultaneously.

1. www.longnow.org
2. Bruce Cannon's work can be seen at <http://home.attbi.com/~bruceannon/>



Figure 4.6: The Stepper Motor Bus Device.

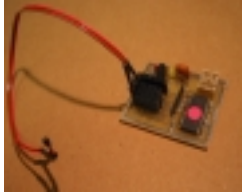


Figure 4.7: The Multi-sensor Input Bus Device.

4.2.3 The Bus Device To Think With

Whenever scaffolding is used in an educational environment, in tools or curriculum, ways must be provided to allow the user to reveal the concepts, techniques, and complexities behind the scaffolding. For instance, users of Web page development tools such as Dreamweaver can view the HTML and JavaScript source that is created in order to learn the details of authoring in those languages directly. The complexities of these languages are scaffolded so the user can instead focus on the design of the page, but interested users can view and understand the lower-level details.

Users knowledgeable with the underlying details of the Cricket system would be able to create new devices to better suit their individual needs. The ideal of user-designed Cricket peripherals is discussed in a paper by Martin, Mikhak, and Silverman:

“In this paper, we introduce *MetaCricket*, a hardware and software construction kit for building computational devices. *MetaCricket* has two crucial properties. First, it allows designers of all backgrounds, not just engineers, to create working prototypes of their ideas, ready for honest critique, analysis, and feedback. Second, the *MetaCricket* system is itself easily extensible. It is growing continuously, and is customizable by designers who have modest hardware and software backgrounds.” (Martin et al., 2000)

However, ‘modest’ in this case means having experience with digital electronics, designing digital circuits, and assembly or C programming; skills that most often only learned in engineering curriculums. The Cricket system, no matter how many Bus Devices are created or how well the designers of them consider their audience in designing them, will probably not meet the user’s needs for every project. A bridge is needed to provide expert users with experience in the areas necessary for them to become cre-



Figure 4.8: The Bus Device to Think With.

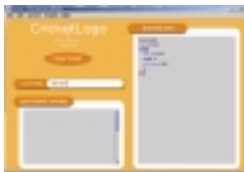


Figure 4.9: An extension of Jackal to program the BDTW.

ators themselves.

A first attempt at this bridge is the Bus Device to Think With (BDTW). I created this device with a very different goal in mind than most Bus Devices. Instead of giving the Cricket a new sensing or output ability, its purpose is to serve as an entry point to the basics of microcontroller programming and electronics. With it, users can explore the concepts necessary to understand lower-level microcontroller programming and basic electronics within the familiarity of the Cricket system and CricketLogo language (*figures 4.8 & 4.9*).

It was created for the ‘expert’ Cricket user, who is comfortable with Cricket programming to the point that it no longer provides much difficulty. Ideally, the user has also experimented with building their own sensors, motors, or power supply for the Cricket, although this is by no means a requirement. Mostly, the user should have confidence in using the Cricket. The ideal user also has a strong interest but no background in electronics and microcontroller programming.

The Bus Device To Think With is named in regard to Papert’s concept of the ‘object to think with’ (Papert, 1980). It provides activities that result in the creation of an interesting object whose behavior incorporates or represents an interesting or important idea in microcontroller programming or electronics. These activities are meant to not only provide users with experience and skills in these fields, but also to orient them to the literature and documentation common to the field of engineers and hobbyists so that they may progress past the activities supplied.

The native functionality of most microcontrollers usually includes one or more of the following: digital I/O, analog-to-digital conversion, serial communication, pulse-width modulation, and various timers and program interrupts. These microcontrollers are usually programmed with either the C or assembly programming languages, although more exist. They also almost always require external circuitry to run. Thus, the difficulty in learning to use these devices is threefold: Understanding its native functionality, understanding the language used to program it, and understanding basic electronics. The Bus Device to Think With allows users to detangle these three problems and attack them one at a time. This aspect of the device will be discussed in the section titled *The Bus Device to Think With in Use*.

The Bus Device to Think With is intended to allow students to explore electronics, microcontrollers, and the C programming language. The C programming language was chosen because of the large community of users, tools, and documentation centered on this language. This language is also the simplest and best documented way of developing new Bus Devices. While C does not have the ease of Logo or the power of assembly, it can be used to program many different types of microcontrollers, desktop applications, and computer graphics. The Bus Device to Think With is not meant as a tool to use in the creation of art. It is only an intermediary tool to bridge the gap between programming a Cricket in CricketLogo and programming a microcontroller in C.

The BDTW is similar in some ways to the LogoChip, a new microcontroller being developed by Bakhtiar Mikhak, Brian Silverman, and Robbie Berg. This microcontroller is programmed in Logo, allowing Cricket users

to encounter the concepts behind microcontrollers with the flexibility and familiarity of this language. This device was developed to introduce college students to microcontroller programming. However, part of the goal of the BDTW is to allow students to move from Logo to C in order to gain the immense resources, documentation, and assistance available from the large community of C programmers. The LogoChip does not support this need as it was developed with different goals and for a different audience.

4.2.3 The Bus Device to Think With in Use

The first step in using the BDTW requires it to be plugged into a Cricket. With the CricketLogo programming language, users can execute the native functions of the PIC microcontroller aboard the BDTW. For example, the following CricketLogo program utilizes the digital I/O capabilities of the PIC to cause one pin on it to alternate between a high state (5 Volts) and a low state (0 Volts) repeatedly, with a 1 second wait in between:

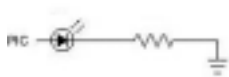


Figure 4.10: LED circuit from BDTW tutorial.

```
to turn_off_and_on ;the name of program
  bit_clear $A2 ;set the pin as output
  loop[
    pin_set $A2 ;set the pin 'A2' high
    wait 10 ;wait 1 second
    pin_clear $A2 ;set the pin 'A2' low
    wait 10
  ]
end ;end of program
```



Figure 4.11: The circuit built on the BDTW

After constructing the circuit (*figures 4.10 & 4.11*) with a LED and resistor, this program will cause the LED to blink.

In the example above, users are introduced to the concepts of pins, ports, and digital inputs and outputs. The electronic concepts of voltage, resistance, and current are briefly covered as well.

During the second step in using the BDTW, the user directly programs the PIC of the BDTW with the program below, written in C (the ‘setup’ portion of this program omitted for clarity):

```
void main(){ //the 'main' program

//set pin 'A2' as output
bit_clear(PORT_A_DDR, 2);

//loop
while(1){
    bit_set(PORT_A, 2); //set pin 'A2' high
    delay_ms(1000);    //wait 1 second
    bit_clear(PORT_A, 2); //set pin 'A2' low
    delay_ms(1000);    //wait 1 second
}
}
```

This program, running directly on the PIC, will behave just as the earlier CricketLogo program did. Because of this feature, users can understand the C program based on their knowledge of CricketLogo. For instance, the structure ‘loop’ in CricketLogo does not exist in C; instead we use the ‘while’ structure with a *condition* that is always true.

In this manner, users can focus on understanding the functionality of the PIC or on the electronics because the programming language is scaffolded, but remove the scaffolding and focus on language once they are comfortable with the electronics and functionality of the microcontroller. This ability represents the design ideal of ‘tearing down the scaffolding.’

The activities included in the Bus Device to Think With Manual (*see Appendix B*) include additional digital input and output exercises, techniques for controlling motors, using sensors and analog-to-digital conversion, and communicating with a desktop computer via serial communication. Each activity introduces new concepts in electronics and microcontroller functionality.

Lastly, the BDTW can serve as a platform for developing PIC programs and peripheral circuitry for prototyping for users comfortable with both. Users ready to start writing their own programs and constructing their own circuitry can reuse the BDTW for this purpose.

4.3 Curriculum and Structure

An Introduction to Robotic Art was structured around two weekly sessions, one consisting of an activity meant to draw out an interesting aspect of robotic art and the other was an open studio time in which students could continue to build on previous activities or create something of their own interest. The activities are drawn from a document I wrote, titled *Creating Behavior with Crickets* (see Appendix A), that highlights what I consider to be the most interesting possibilities of the use of Crickets in art. I chose to focus on the behavioral aspects of robotic art because this potential is normally not encountered when working with traditional materials. The manual begins with a clear definition of its purpose and expectation of the user's commitment:

This document has two goals. One is to provide the user with activities to provoke thinking and discussion around the field of robotic art. The second is to provide the user with useful programming skills that will serve as a starting point for future explorations. The user should be prepared to conduct research in both fields outside of this document as well as working on a long-term project outside of these activities in order for the concepts introduced here to become clear.

It is organized into six different activities, each focusing on an interesting ability of the Cricket. These activities are meant to introduce the student to the qualities of the Cricket and aspects of robotic art that I find most interesting. These activities are summarized below:

Activity One: Actuation – *Create a machine that provides movement to a found object in a way that suggests life, intelligence, or irony.*

Activity Two: Sensing – *Construct an object to provide an ambient display of an environmental factor.*

Activity Three: Time - *Create a work whose behavior changes by date or time, evolves, or ages.*

Activity Four: Communication – *As a group, create a ‘cascaded’ series of behaviors. Each person’s work should wait until an infrared signal is received, respond to that message through movement or sound, and then pass the message on.*

Activity Five: Organism and Machine - *Construct two separate works, one that attempts to mimic organism-like interactions and one that behaves in a machine-like way.*

Activity Six: Connecting to the Computer – *Send sensor values to a computer to manipulate on-screen images.*

Each activity includes an explanation of the activity and a sample Cricket-Logo program to get started with. Links to artists’ works relevant to the activity are also included.

During the first session of the week, we discussed the activity at hand and then worked to build a prototype. Related work in the field was discussed if applicable to the activity or to a student’s ideas. Gretchen’s extensive knowledge of work in this area was useful to students as they began to understand its history and potential. The activities also served to launch discussion of principles of electronics and programming. For example, the ‘communication’ activity was followed by a discussion of digital serial communication, the basis for the Cricket infrared and bus communication. The ‘sensing’ course had a follow-up discussion during the studio time in which we soldered together our own sensors and discussed the electronics

concepts of resistance, analog-digital conversion, and voltage dividers. In another class, we discussed the concept of pulse-width modulation that the Cricket uses to control the speed of motors and the electrical property of motors known as inductance.

The second session served as studio time, in which students were encouraged to experiment with the materials outside of the scope of the activity or work on a longer-term project. Many times, the activity would spill over into the studio time. As students developed ideas for projects outside the activities, they would discuss their ideas with Gretchen and myself, along with other students if desired. Gretchen and I would provide feedback on the idea, provide references regarding related work, and encourage them to construct a prototype. Prototypes were shown during class and used to prompt discussion. Many students took home the materials and worked outside of class.

At the middle of the semester, students were asked to propose a final project. These proposals were discussed with Gretchen and myself; we provided feedback conceptually and technically. These projects provided a focus for the remaining studio sessions.

Almost thirty students and faculty attended the initial meeting. Due to time constraints and misconceptions regarding the nature of the course, this number dropped to approximately fifteen by the second meeting. As this course was voluntary and no credits were awarded, it often was given lowest priority for students who had to worry about their for-credit courses. Four or five of the remaining fifteen students would attend occasionally (approximately every third session), and three others came approximately every other session. Six students, however, were very dedicated to the course and attended almost every session. The work and reflections of

some of these students will be discussed in the evaluation section that follows.

Chapter Five - Evaluation

In this section, I discuss the methodology used for the evaluation, the aspects of student engagement that point to successes in the course, and lastly its failures.

5.1 Evaluation Methodology

The evaluation was conducted using recorded interviews and conversations with students before, during, and after the course, students' notebooks and the art they created during and after the course, and in informal and formal questionnaires. The analysis of this data was focused on the evolution of students' approaches to the materials throughout the duration of the course, the ways in which their conceptual framework of robotic art changed, and their reflection of their own learning and artistic processes. Through this analysis, five distinct aspects of student engagement stood out. These aspects represent approaches and processes of the students indicative of a successful engagement with the materials, concepts, and/or technique of the field. These are discussed below in the following case studies. These studies point to approaches and processes found in student's engagement with the materials that contributed to their understanding of the materials, concepts, and techniques of robotic art.

5.2 Aspects of Student Engagement



Figure 5.1: Copper 'carapaces' made by Nathan.

5.2.1 Experimentation and Iteration

Nathan is a sophomore in the fine metals program at MassArt. Nathan's work has primarily focused on jewelry and knives, although he admits that is still too new to metalsmithing to have explored many of its possibilities. He recently exhibited some of his work at a student gallery at MassArt.

Nathan describes his process as an explorative one:

"It seems like a lot of the stuff that I make comes together as a result of lots of different experiments. I will become interested in certain kinds of processes or even a certain type of shape. I won't usually have a particular vision for it when I first start. For example, these (small paper 'carapaces') were a result of a lot of messing around. Then later, after I actually made some of them (out of metal), I will imagine them integrated into things and build ideas into them. It's an explorative process; one thing leads to another and I combine them all" (*figure 5.1*)

Clearly, the creation of prototypes is fundamental to Nathan's approach as they spark new directions and serve to focus his ideas. Midway through the Introduction to Robotic Art, Nathan explained how he thought this related to the Cricket:



Figure 5.2: Nathan's glass and cardboard prototype.

"This prototyping process is very important. The Cricket; I guess it's a bit different than 'prototyping', but it serves the same role in a way. Just by playing with these things; I didn't even know things like distance sensors existed, but the ability to play with them and see how they can relate to other attributes of a piece; that has started very different thoughts about what is possible. While I haven't really focused on building working things with the Cricket, just playing around with different behaviors and different modules...very small programs and simple things, but it has given me a few ideas that I could follow up on."



Figure 5.3: Copper leaf prototype. The leaf is actuated by a pull on the string.

Nathan's final project is a copper and glass flower that provides an individual viewer with a different experience than a viewer within a crowded gallery. He wants the flower to monitor the activity level in the room with motion and sounds sensors. If the gallery is busy, he wants the flower to remain fairly static and be "just another decoration" within the room. How-



Figure 5.4: An early drawing of Nathan's final project.

ever, when the room is quiet and an individual viewer is seeing the flower, the flower opens its petals and the bulb changes color. He says that the idea for this project grew out of many small Cricket constructions that he made, while the shape of the flower grew out of his experiments making insect carapaces from metal. This similarity means that the experimental qualities of the Cricket fit within Nathan's well-developed process of working with metal and allowed him to approach artmaking with computation in a similar way. In other words, he was able to use the same experimental process that he uses for creating with metal in authoring behaviors with the Cricket.

Nathan is a very talented illustrator and originally wanted to attend art school for drawing. However, a summer session at the Rhode Island School of Design in metalsmithing made him immediately change his mind. However, his drawing skills benefit him greatly in his design process as his drawings serve as the first step in the prototyping sequence. From his drawings he creates paper and cardboard models for designs requiring flat layers or clay for objects require casting. Much of this need for prototyping is due to the difficulty of working with metal; so much time is needed to create a finished work that Nathan wants to have a clear picture of the piece before he begins. He works with paper and cardboard until feels like he is ready to start fabricating parts from metal. Even after creating parts with metal, he chooses to use metal pins instead of rivets so that it can be easily disassembled. Rivets replace the pins when the piece is considered finished. For Nathan's final project, he considers the behavior of the object as important as the aesthetic and is confident that the Cricket will give him the ability to "fine tune" the behavior. He knows that the quality of interaction he is looking for is subtle and hard to achieve, but remains hopeful that with enough time he can achieve "this very personal interaction". He believes that he can iterate on his program in ways similar to how he iterates in metal.



Figure 5.5: Drawing of rings with movable parts.



Figure 5.6: Another concept drawing of movable rings.

Nathan says that working with these materials has given him new ideas with for use within his traditional materials. Nathan says that some faculty had encouraged him to incorporate moving pieces into some of his creations, yet he felt that “it felt like a lot of work just to make something with a hinge” and that the aesthetic value of his work was his primary focus. Working with motors has made him think about objects whose shape could change (*figures 5.5 & 5.6*). For an assignment in a different course, he designed “rings and bracelets that have the pieces that swing out; I guess in some way I had never thought about motion in my work, and just working with motors and controlling things with them has focused me on motion much more.”

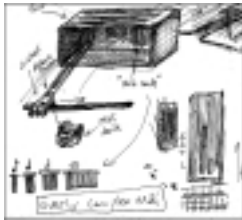


Figure 5.7: Early drawing of player piano.

5.2.2 Conceptual Engagement

David is junior in the 2D department at MassArt where he focuses on painting and drawing. He describes his approach to art as one that relies on chance and unexpected properties:

“I usually just start with making ideas in my sketchbook. If I like one, I will pursue. I like to draw a lot of characters and people. I will be on the T (Boston’s subway system) and I will draw random people. If they spark my imagination I will go further with that sketch and develop them in caricatures of the original.”

Early in the course, David built a system that mapped a distance value, read by the Cricket, to a series of musical notes played by the MIDI Bus Device. At first, these distances were mapped directly to the MIDI device, but he later changed the code so that only a subset of the possible notes would be played, allowing the system to stay within a key or song. This object then served as the focus of his thoughts for the rest of the semester. He liked the idea that holes in a card represented notes on a player piano and that this representation was very arbitrary. He wanted to build a similar system that drew upon the idea of arbitrary representation. His first sketch involved a “piano” that was played by arranging playing cards in different positions

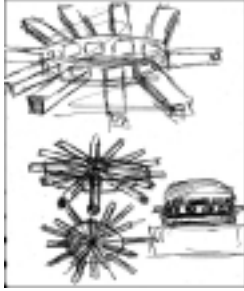


Figure 5.8: Rotary player piano.

(figure 5.7). A small, wheeled component containing the distance sensor would move across the rows of playing cards and play a note representing the distance between it and the card. While the viewer might hypothesize on the relationship between the cards and music because of the many uses of playing cards and the symbols that they contain, the relationship was as arbitrary as punch cards to a player piano.



Figure 5.9: Slide containing the different color slots.

His next revision to this idea was to change the linear arrangement of cards to a rotary one, where the distance sensor remained fixed but the table containing the cards could be rotated, either by hand or by a motor (figure 5.8). David thought that this might be a more aesthetically pleasing solution. Later, David drew a piano that consisted of a cutting board, crackers, and slices and cheese and meat. This design was similar to the original playing card design that contained a small movable component that moved back and forth relative to the crackers and slices of meat and cheese. This design extended the arbitrary relationship of the playing cards to a more ridiculous relationship between musical notes and hors d'oeuvres.



Figure 5.10: Unfinished player piano.

The last revision to this idea was to map colors to musical notes. He built long slides that contained small squares of colored film (figure 5.9). By sliding these slides into a box containing a light, photocell, Cricket, and MIDI board, the user could play different songs (figure 5.10). David was trying to illustrate the arbitrary representation of color and sound perception in humans by building a machine where these two phenomenon represent each other.

The playing card piano and the cheese and crackers piano were never constructed. These ideas remained sketches in David's notebook. The original system that David constructed stayed very similar to its original form until his final creation where he replaced the distance sensor with a light sensor. This construction served as an object to think with for David and motivated

him to explore many different conceptual ideas. Representing ideas with objects is the essence of artmaking and thus the ability to develop concepts based on objects or create objects based on concepts is foundational. The prototype that David first built would have been very difficult for him to build without the scaffolding of the Cricket. If the course had been focused on providing the necessary skills, David would have lost the experience of conceptual development that centered on his early creation.

5.2.3 Entry Point

Dawn is a student in the 3D department at MassArt. She is in her mid-forties and has limited experience working with computers. She works mostly with wood, clay, and metal. Dawn described one of her interests in art as being “visual puns.” The Cricket interested her because she felt that in most her work, the pun was obvious at first glance. With the Cricket, she could reveal the “punch line” of her pun to the viewer in ways of her own design. She explained,

“The whole process of having a Cricket is exciting to me because I like mechanical things, and by having the Cricket it adds a whole other dimension. It adds an element to surprise; instead of walking up to something and turning a crank, it turns on as you walk up! The opportunity to use those things and to be able to use...it’s the whole thing with the element of surprise. It’s one more way that I can make my pieces utilize that element of surprise. One pun I wanted to explore was an a refrigerator that contained a cockroach...when you opened the door, the light turned on and the cockroach moved around, but in a way that made you wonder what he did when the light was off. It’s playing to the whole ‘does the light turn off when the refrigerator door closed’.”

Dawn’s final project consists of a carved, wooden dung beetle on a tilting platform (*figures 5.11 and 5.12*). The dung beetle pushes a ball of dung up the slope of the platform until it reaches the top, where the platform tilts back the other way. The beetle then turns around and pushes the dung back

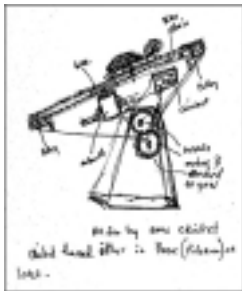


Figure 5.11: Drawing of dung beetle on tilting platform.



Figure 5.12: Carved wood dung beetle.

up the other side. The idea is based on the Greek legend of Sisyphus, who was fated to forever push stones up a large mountain, which would soon roll back down. Although the piece is not yet finished, she plans on titling it *Sisyphus Meets the Dung Beetle*.

I commented to her that a similar motion could be achieved with a motor and gear chain that synchronized the movement of the beetle and the tilting of the platform and that a Cricket was not needed. Dawn was aware of this and said that the mechanics of such a setup would be hard to build. Instead, a light sensor could be used to determine the position of the beetle and the Cricket could move the platform accordingly. This represents the idea of feedback, a powerful idea in computer science. We then discussed what the CricketLogo program would look like.

Dawn was concerned that the course would be too complex for her, but decided to come to the first class after the introductory meeting. She describes her thoughts after the first class:

“The thing for me was, the first class, I made the cardboard cricket. It was very simple, just some wheels and motors. I would say that I am computer illiterate. I am very illiterate. Being able to program something, no matter how simple, was like understanding the speed of light! Doing that was amazing. I remember telling my friend, ‘I programmed!’ Granted it was very easy, but that was always scary to me before. I realized that feeling limits my ability to understand; it’s why I have remained computer illiterate for so long.”

The simple act of exerting control over the computer in a comfortable environment provided the confidence to continue on in the course. After completing her project with the dung beetle, Dawn admitted she might explore a few more “simple ideas,” but is unsure as to what they might be. Most importantly, Dawn felt that computers are approachable:

“I don’t think I ever thought of using computers before, because I am so anxious about them. But now, it’s doable.

It's not so far beyond me. If anything, I left this class thinking this might be a tool that I could incorporate into my work. That's been real helpful for me. It's made another tool approachable, accessible. That's why the Cricket is good. It makes the tools approachable to those outside of the field. I'm in my forties; I remember when the first liquid crystal watch came out! I was in my thirties when I first used a computer; just to spell-check my thesis!"

5.2.4 Materiality

Near the end of the course, I asked Jen to describe her thoughts on the Cricket. Being an art educator, she responded in terms of its role as a learning tool.

"I like that the possibilities seem to be endless. Each new thing seems really simple, the ideas each seem really simple, but that doesn't mean that they can't be profound. I like that. It seems like there are a lot of doors it could open. I also love it as a learning tool. I don't think there is any better way to talk about or show people how a computer can respond to a human being, how you can have a dialog, an interaction, I don't think there is a better way than this instant thing; it's right here, that you feel like you have control of. It's this little device ...that you are actually 'collaborating' with. I think there is something very poetic right there about that. Outside of that, as a learning device or teaching device, I think it is very empowering for anyone who thinks about interactions."

Later in that interview, I asked her to describe other ways in which she would categorize computers as tools. She immediately thought of her work in 3D modeling: "The computer is a tool, for drawing or plans or design. Especially if I am going to be doing something that is going to be fabricated." Jen initially thought of the Cricket in similar ways: The Cricket was a *tool* that you used in order to achieve an idea. In later interviews, however, Jen began to describe the Cricket and Bus Devices as materials.

"In the beginning of the course I was completely confused by all the options. The more you would bring in; there is a clock, a touch sensor, then in that same kind of intuitive 'wouldn't it be cool if' way that is all about...[my] process when I work at my bench in metal, 'wouldn't it be cool if I

did this with this metal.’ You would add an ingredient (a new concept or device) and it would spark an idea in a similar way. ‘Let’s make that, that would be cool.’ It tapped into, being exposed to possibilities, tapped into the same intuitive process. You are not designing an idea, then finding the support materials to make it happen; you are responding to the materials. The class allowed us to respond to the materials.”

She described one instance of her response to these materials:

“My favorite moment was when you brought in the touch sensor. I wasn’t planning on doing that whole piece of the LED’s, but it sparked a ton of ideas that there is a way of gentle interactivity. I am really interested in touch, you make an object, somebody touches it, and how they interact with it is interesting to me. [The touch sensor] changed my thinking entirely. Initially, I was thinking that you motorize something and [a viewer’s] presence makes something happen. What I like about a lot of the work I make is that it has to do with you turning something by hand, a crank or something, touching something, making something happen; it was another entry into it that was softer. That really affected the design of the work and I am still thinking about it for other projects.”

She then described her use of the materials in the work *Ludicrum*.

“The motors ended up being part of the [piece]. They give the piece the behavior, it gives it its life, it makes them respond to it in a totally different way; I don’t know if I can categorize it as a tool anymore. The Cricket is kind of a facilitator, I think. It just becomes part of the vocabulary that you reach for.”

I believe the change in Jen’s categorization of the Cricket from a tool to a material signifies a successful experience in the course and has two underlying factors:

Understanding of expressive potential: Through the process of creating many different works with the materials, Jen understands what potential they have for her own work. She understands the conceptual and aesthetic qualities of the materials of robotic art.

Fluency with the materials: Early in the course, Jen described artmaking as, “this thing about an idea that you chase, the more fluid you are with the idea, the more fluid you are with materials, the more able you are to chase an idea in a way that you are OK with not catching it in the end.” Jen’s programming skills have reached a level where she is fluent with the materials, allowing her to accommodate these materials into the process she describes here. Her confidence in the materials allows her to use them in natural ways.

5.2.5 Extensible

Jen was the only student of the six who chose to explore lower-level micro-controller programming and electronics with the Bus Device to Think With. (One student started working independently with electronics halfway through the semester after buying some books from Radio Shack. He is now in the process of creating an installation using contact microphones that he built.)

Jen’s desire to learn more was driven partially by some of the limitations of the Cricket. At one point during the construction of *Ludicrum*, she asked me for help debugging a Bus Device intended to control 16 sets of LEDs, which appeared to be malfunctioning. During the debugging session, I mentioned that creating the effect she desired would be very simple with a PIC and that it would allow her to control 60 or more LEDs instead of 16. This comment sparked a new interest in learning more about PIC programming and electronics. She describes this interest:

“I think I have the ‘bug’ now for it. I would really like to understand it. One of the most exciting things about this is that it provided easy access into programming. Maybe not easy, even if it is ‘false’ easy, it makes it accessible. I love learning new stuff, and I think this is the next thing I want to delve into.”

The Bus Device to Think With provided Jen with a comfortable environment to begin the often-uncomfortable process of learning something new. While the birth of a son understandably interrupted her progress, Jen is now ready to begin PIC programming, using the Bus Device to Think With as her platform.

5.3 What Didn't Work

The large number of people that dropped out of the course through the semester is a concern. The course was not-for-credit and the majority of students that left certainly did so in order to focus on their other classes. A challenge for educators wishing to provide a free environment is to find ways of constraining certain aspects of the experience while leaving students free to explore their own interests. I would find ways of doing this in future courses based on these materials.

There were a few 'bugs' encountered during the course with the Cricket. Many students were working with these materials at home or at their own studio and these bugs were very frustrating, as they did not know if it was their fault or a flaw in the Cricket or programming environment. After the proper documentation of these issues, students did not necessarily mind them. They mostly wanted to know if it was their fault or whether they should move on or try to find a way around the bug. While it is unreasonable to expect learning tools to be flawless, it is completely necessary for them to be fully documented. The Cricket, its Bus Devices, and programming environments need further documentation at all levels.

Most people took a relatively long time to begin creating with the materials. While the scaffolding of the Cricket allowed students to begin creating in order to understand what they need to learn, the artist must still design in

order to learn how to design (Schön, 1985). The qualities of the material take experience working with to understand, but students were slow to get started because they felt they didn't properly understand the qualities of the material. More examples of art made with Crickets or related projects would have been valuable to help students get started. A database of projects in this area would be beneficial.

One student, John, attended class sessions only for the first half of the semester. He had a fairly knowledgeable background in electronics and was a recent graduate from MassArt. During the first session, he created a sophisticated 'clock' that consisted of a large wheel that oscillated back and forth. The oscillations were caused by feedback from light and magnetic sensors. While John's enthusiasm during the first few sessions was undeniable, he lost interest in the course after he realized that he knew most of the concepts (in art, programming, and electronics) covered. Clearly, a course more focused on technical skills would have better suited John.

The Bus Device to Think With is an explicit attempt to make the Cricket system deconstructable and understandable. While I feel that it is successful in this regard, it is far from ideal. Instead, the Cricket itself should be designed with this goal in mind. The Cricket virtual machine is a program written in a unique mnemonic of assembly language. Not only is assembly language a difficult one to master, this unique mnemonic lacks good example code and documentation. I believe that this core program should itself be written in C so that it is understandable by interested users. This would also help communities with specific needs modify the Cricket itself.

Additional future work in this area would include the creation for a friendly C programming environment for the Cricket. This environment could also be used to program PIC microcontrollers. It would contain features that make it easier for users to develop their own Bus Devices. In this manner,

the Cricket system could be customized by the Cricket experts within the community for use by others within that community. The Bus Devices built within the community would reflect the needs and desires of the collective users.

An alternative to learning C or another language to program microcontrollers would be to use the LogoChip being developed by Mikhak, Berg, and Silverman. This device, however, is still in development. It will be a valuable addition to the Cricket System.

Chapter Six - Conclusion

This thesis documents the design and implementation of a course intended to allow art students to explore electronics and programming through the authoring and creation of behavior-based robotic art. This course provides an entry point into the technical requirements of these materials but also provides students with design experience. This is important in order for them to develop concepts and processes regarding the use of these materials. The scaffolding of the complexities of these materials by the Cricket allows for students to approach the creation of art in an experimental and iterative way. A process and tool for removing this scaffolding as needed and desired is discussed.

One outcome of this work is a description of course materials designed to provide students experience with a new mode of expression in the context of an arts college. However, the approach outlined in this document will be relevant to anyone interested in introducing themselves to programming or electronics without first learning the technical complexities.

A second outcome is the development of an alternative relationship between artists and engineers. In artistic collaborations, the artist usually provides artistic vision and the engineer implements it within the constraints of her skills and materials. Another method would call for the engineer to scaffold the materials the artist is interested in using, provisionally hiding the technical complexities of the materials but allow for the behavioral programming to be done by the artist. In this manner, the artist is not

burdened with unwanted complexity but is able to make the subtle decisions that ultimately affect the outcome of the work. The artist would describe the overall guidelines needed for the piece and the amount of complexity the artist wants to control. The engineer then develops the hardware (one board or suite of connecting boards) and programming environment (software library, interpreter, or compiler) that satisfies these needs. The engineer is no longer trying to implement the artist's vision but rather scaffolding the technical details to a level the artist is comfortable with. The artist regains control over the implementation and has the ability to experiment with the materials at hand.



Figure 6.1: An example program written in MAX/MSP.

Some limited examples of this approach exist, but only for software tools. MAX/MSP¹ is a popular music and multimedia graphical programming environment originally developed by Miller Puckette (*figure 6.1*). It is often used in the creation of interactive music and video and live performance art. Users knowledgeable in C can add 'modules' to MAX for specific purposes. These modules are represented graphically for use within the normal graphical programming environment. The most successful modules are often bundled in future releases of MAX/MSP. An artist in need of a specific tool or algorithm in MAX can recruit a programmer to create it.

A similar system could exist for microcontroller hardware; an artist would list performance guidelines and an engineer would create the hardware and programming environment for the artist. This approach requires a dedication by the artist to program and commitment from the engineer to create tools that others can create with. While these needs may require more effort for both engineer and artist, the arguments contained in this thesis suggest that this model of collaboration could lead to a more satisfying relationship between the two.

1. For more information, see <http://www.opcode.com/products/max>.

References

- Anderson, Albert A. (1992) *Discipline-Based Art Education, American Craft*, V. 52 April/May 1992
- Begel, Andrew (1996) *LogoBlocks: A Graphical Programming Language for Interacting with the World*. Advanced Undergraduate Paper in Electrical Engineering and Computer Science. MIT, Cambridge, MA.
- Brown, J. S., Collins, A., and Duguid, P. (1989). *Situated cognition and the culture of learning*, *Educational Researcher*, 18(1): 32-42.
- Burnham, Jack (1967) *Beyond Modern Sculpture*. George Braziller.
- Collins, A., Brown, J. S., & Newman, S. E. (1989). *Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics*. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honour of Robert Glasser*, (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Kac, Eduardo (2001a) *The Origin and Development of Robotic Art* (p.76-86) *Convergence: The Journal of Research into New Media Technologies* (Vol. 7, No. 1).
- Kac, Eduardo (2001b) *Towards a Chronology of Robotic Art*. (p.87-111) *Convergence: The Journal of Research into New Media Technologies* (Vol. 7, No. 1).
- Maeda, John (1998) *The South Face of the Mountain*, *Technology Review*. July/August. 1998
- Maeda, John (1999) *Design by Numbers*, MIT Press, Cambridge, MA.
- Martin, Fred (2000) *Robotic Explorations: A Hands-On Introduction to Engineering*. Prentice Hall, Upper Saddle River, NJ.
- Martin, Fred (1994) *Circuits to Control: Learning Engineering by De-signing LEGO Robots*, Ph.D. thesis, MIT, Cambridge, MA.
- Martin, F, Mikhak, B, and Silverman, B (2000) *MetaCricket: A designer's kit for making computational devices*. *IBM Systems Journal* (Vol. 39, Nos. 3 & 4)
- Papert, Seymour (1980) *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York.

Papert, Seymour and Harel, Idit (1991) *Constructionism* Ablex Publishing Corporation

Papert, Seymour. (1991). *Situating Constructionism*. Constructionism, eds. Idit Harel and Seymour Papert.

Piaget, Jean. (1963) *The Psychology of Intelligence* (Paterson, N.J, Littlefield, Adams)

Rath, Alan (1999) *Robotics*. Smart Art Press, Santa Fe.
Catalog of an exhibition held at Site Santa Fe, Oct. 21, 1998-Jan.24, 1999

Reas, Casey (2001) *Behavioral Kinetic Sculpture*, Master's Thesis, MIT, Cambridge, MA

Resnick, M, Martin, F, Sargent, R, and Silverman, B. (1996). *Programmable Bricks: Toys to Think With*. IBM Systems Journal (Vol. 35, No. 3-4, p. 443-452).

Resnick, M., Martin, F., Sargent, R., and Silverman, B. (1996). *Programmable Bricks: Toys to Think With*. IBM Systems Journal, vol. 35, no. 3-4, pp. 443-452.

Resnick, M, Berg, R, and Eisenberg, M (2000). *Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation*. Journal of the Learning Sciences (Vol. 9, No. 1, pp. 7-30).

Schon, D.A. (1992) *Designing as Reflective Conversation with the Materials of a Design Situation*, Research in Engineering Design, 3, pp.131-147

Schon, D.A. (1983) *The Reflective Practitioner: How professionals think in action* (New York, Basic Books Inc.).

Smith, Ralph (1987) *The Changing Image of Art Education: Theoretical Antecedents of Discipline-Based Art Education*. The Journal of Aesthetic Education, Vol. 21, No. 2, 1987

Stern, Andrew (2001) *Deeper Conversations with Interactive Art Or Why Artists Must Program*. (p.17-24) Convergence: The Journal of Research into New Media Technologies (Vol. 7, No. 1).

Turbak, Franklyn, and Berg, Robbie (2001) *Robotic Design Studio: Exploring the Big Ideas of Engineering In a Liberal Arts Environment*. Submitted to the Journal of Sci-

ence Education and Technology.

Yakeley, M (2000) *Digitally Mediated Design: Using Computer Programming to Develop a Personal Design Process*. Ph.D. thesis, MIT, Cambridge, MA.

Web References:

Robotic Art Studio - <http://www-2.cs.cmu.edu/afs/cs/usr/mai/www/course/15-493.html>

Physical Computing - <http://fargo.itp.tsoa.nyu.edu/~tigo/pcomp/index.shtml>

Appendix A



Creating Behavior with Crickets

This document has two goals. One is to provide the user with activities to provoke thinking and discussion around the field of robotic art. The second is to provide the user with useful programming skills that will serve as a starting point for future explorations. The user should be prepared to conduct research in both fields outside of this document as well as working on a long-term project outside of these activities in order for the concepts introduced here to become clear.

Accompaniments to this document include the CricketLogo language reference (attached) and the Jackal programming environment.

Parts List:

1. Cricket
2. Interface Cricket
3. Light, temperature, touch sensors
4. Motors and motor cables
5. Distance, Clock, Tri-color LED, Big Motor, Stepper Motor, ??? Bus Devices
6. AC to DC power adapter (outside barrel positive)

Setting up:

1. Install Jackal.
2. Plug serial cable into computer serial port, Interface Cricket.
3. In Jackal, select Edit/Preferences and choose the serial port you are using.
4. Point the infra-red parts of the Interface Cricket and the Cricket towards each other.
5. In the Command Center in Jackal, type 'beep' and press return. The Cricket should beep.

Your first CricketLogo program:

Jackal has three windows- **Procedures**, **Command Center**, and **Run This**. The **Command Center** is for trying out single lines of instructions: type the instruction here and press return. Your programs will be written in the **Procedures** window. You can write as many procedures as you want in this window, each starting with 'to' and finishing with 'end', as shown below.

```
to test
  repeat 10 [note random 5 wait 2]
end
```

Now, in order for the Cricket to know what procedure to run when the white 'run' button is pushed, we must write the name of the procedure in the **Run This** window. Now, a few other points:

Primitives and Flow

Write a program to build a peanut butter and jelly sandwich. Here is part of mine (this isn't real, of course):

```
to make-believe-peanut-butter-sandwich
  grab-knife
  open-jar (peanut)
  if(jar-open)[
    repeat 10 [ dip-knife spread-substance]]
  close-jar (peanut)
  open-jar (jelly)
  if(jar-open)[
```

```
    repeat 10 [dip-knife spread-substance]]
  close-jar (jelly)
end
```

```
to grab-knife
??
??
end
```

```
to open-jar :n
??
??
end
```

```
??
```

What are the *primitives* (the native commands that the program is built out of, such as repeat)? What are the *procedures* (blocks of code starting with to ending with end)? For example, *make-believe-peanut-butter-sandwich* is one, the others are *grab-knife*, *open-jar.etc*, although they are not shown here. The *procedure make-believe-peanut-butter-sandwich* calls the other procedures, which do their thing and return. Are there collections of primitives that often get reused? What is the *flow* (how does the Cricket step through the program)? Does any process get *interrupted* by another? What are the *conditionals* (what conditions are checked, then acted upon)?

Apply the same questions to this CricketLogo programs. These *will* work on your Cricket, so plug in some motors, type the code in the **Procedures** window, type *dance* into the **Download This** window, and download them!

```
to dance
  step-forward
  step-back
  repeat 2 [step-forward]
  repeat 2 [step-back]
end
```

```
to step-forward
  note 20 10
  thisway
  ab, onfor 5
end
```

```
to step-back
  note 30 10
  thatway
  ab, onfor 5
end
```

Bus

A bus is a communication method that allows computers to talk to other devices. In Crickets, using the bus is as simple as plugging in a device and remembering the right primitive that controls the device. For instance, plug in a digital display bus device and run the following program:

```
to test
  display 99
end
```

The primitive *display* tells the digital display to display a number, in this case 99.

Syntax

Computer languages have syntax just as written languages do. Syntax is a burden we just have to deal with. Luckily, CricketLogo has little.

Procedures must start with 'to' followed by a procedure name of your choosing and end with 'end'. Math operations must be spaced and brackets and parenthesis are used in many primitives.

Consider:

```
to blah
  if (sensora * 2 > 100) [beep]
end
```

Input/Output

Computers can have both inputs and outputs, such as the mouse or monitor that we are all familiar with. The Cricket has three 'primitive' outputs (motors, speaker, and infra-red.) and two inputs: (sensors and infra-red).

Sensora values are read with the primitives 'sensora' or 'sensorb'. Sensor values read from 0 to 255. Crickets can also use bus devices as an input or output device.

Please read through the attached CricketLogo reference to get a better understanding of the syntax and commands of CricketLogo. Through the following activities, take some time to look up the commands used in the sample programs and also try to understand the structure of the programs. Analyze each sample program line-by-line to understand their flow. At anytime, break away from these activities to explore something in-depth.

Activity One: Actuation and Motors

Movement is the first quality we will explore that alludes to behavior. This activity focuses on animating a found object in ways that suggest life, intelligence, or irony in its movement. Please read the below choices, play with each, then build something, focusing on one quality of movement.

Built-in Motor Ports: The Cricket allows us to control the duration, direction, and speed of motors. Try this program after plugging in a motor to *Motor Port A* (see Cricket Diagram) on the Cricket:

```
to motor-test
  a, on           ;turn motor a on
  wait 10        ; wait one second
  a, rd          ;reverse the motor direction
  wait 10
  a, setpower 2  ;set the power lower, to 2 out of 8
  wait 10
  a, setpower 8  ;bump it back up to 8
  wait 10
end
```

Motor port B works the same way. Another motor command is *a, onfor*. This command turns on a motor for a given amount of time. However, the Cricket will move on to the next command while the motor is still on, waiting to be turned off. This is different than the *wait* command, which pauses the execution of the program for the duration of the wait. For example, the two below programs function very differently.

```
to motor-one           ;turns on motor a, quickly changes its
                      ;direction, after
  a, onfor 30         ;3 seconds shuts it off
  rd
end
```

```
to motor-two          ;turns on motor a, waits 3 seconds, changes
                      ;its direction
  a, on
  wait 30
  rd
end
```

Gearing is still sometimes necessary, as are mechanisms for converting a motor's circular motion to linear motion. Fred Martin's *The Art of LEGO Design*

(<http://handyboard.com/techdocs/artoflego.pdf>) is a good introduction to prototyping mechanisms with LEGO.

Also, as a motor's turning rate is unreliable, we will also want to incorporate feedback to ensure precise positioning. This could be accomplished by having the moving object, or some part of the mechanism in motion, trip a touch sensor to stop the motion, change the speed or direction, or start a completely different behavior as well. Suppose we have a motor turning a lever of some sort, which at some point will run into a switch. Code for this situation might look like the following:

```
to motor-feedback
  a, on                                ;turn motor a on
  loop [ if(switcha)
          [a,off]
        ]                               ;continually check the state
                                           ;of switch a- if
end                                     ; pressed, turn off the motor
```

Servo Motors: Some motors have this feedback ability built into them- these motors are called 'servo' motors. The Cricket has a special board (bus device) that can control servo motors. Instead of turning them on or off (the servo motors used with this board cannot turn in full circles), we tell them what position to go to. This is accomplished with the command *turn-servo*, which we pass the servo motor number (labeled on the device) first, followed by the position we want it to go to. This position number must be experimented with in order to obtain the position you want. With the Servo Motor bus device connected to the Cricket and a servo motor plugged into it (black wire should be closest to edge), run the following program:

```
to servo-test
  turn-servo 1 20                       ;turn servo 1 to position 20
  wait 5
  turn-servo 1 50
  wait 5
  turn-servo 1 90
  wait 5
  turn-servo 0
end
```

Stepper Motors: Stepper motors are another type of motor that can be quite useful. Stepper motors turn in angular steps, allowing the precise positioning of servo motors but can also turn all of the way around. The Stepper Motor bus device can control two steppers with the following commands:

```
a-step-speed :n                        ;set the speed of stepper
                                           ;a 0-100
```

```

a-step-forward           ;step forward forever
a-step-off               ;stop stepping
a-step-brake             ;lock up motor so it won't
                        ;turn
a-step-back              ;step backward forever
a-step-forwardfor :n    ;step forward of :n number
                        ;of steps 0-255
a-step-backwardfor :n   ;step backward for :n
                        ;number of steps 0-255

```

Note: Match up the color abbreviations on the Stepper Motor bus device when connecting a stepper motor.

Big Motors: One more actuation option is the use of big DC motors. While the Cricket can run DC motors on its own, you are limited to motors that run on less than 9V and draw less than .4 Amps. For people interested in controlling larger DC motors (needed for more power), the Big Motor bus device is needed. This device needs its own power supply, which needs have a connector with the outside barrel the positive supply, and inside barrel ground. The power supply can be anywhere from 5V to 40V. Once this and the motor are connected, the following commands can be used:

```

a-setpower :n
a-thisway
a-thatway
a-brake
a-off
b-setpower :n
b-thisway
b-thatway
b-brake
b-off

```

See the appendix for places to find cheap DC motors.

Artists and their work to view and discuss:

Marc Bohlen - <http://www.contrib.andrew.cmu.edu/~bohlen/salt.htm>
<http://www.contrib.andrew.cmu.edu/~bohlen/alarm.htm>
 Gregory Barsamian - <http://www.concentric.net/~Venial/sculptur.html>
 James Seawright - <http://www.seawright.net/jamesseawright/motion.html>

Activity Two: Sensing

Another interesting aspect of the Cricket is its ability to sense conditions of the physical world. In this activity, we will build an ambient display of an environmental factor. Four sensors plug directly into the sensor ports on the Cricket- they are light, temperature, touch (switch), and capacitive touch. These sensor values are accessed by the *sensora* or *sensorb* command. The values range from 0 to 255. Try the following program after plugging in a sensor to Sensor Port A and a digital display.

```
to sensor-test
  loop[ display sensora ]
end
```

Try out each sensor to get an idea of its range. For details of using the capacitive touch sensor, please see the appendix. Other sensors are bus devices, such as the distance sensor and the clap sensor. Plug in the optical distance sensor and try the following program:

```
to distance-test
  loop[ display ods-get-distance ]
end
```

Strange results? Probably. These sensors have a point at which the number displayed will reverse- one nice way around this problem is to make them into a motion sensor with a simple algorithm. It looks like this:

```
global [dist1 dist2]

to motion-sensor
  loop[
    setdist1 ods-get-distance
    setdist2 ods-get-distance
    if (dist1 > 30)[
      if (((dist1 - dist2) > 10) or ((dist2 - dist1) >
10 )) [beep]
    ]
  ]
end
```

It uses *global variables* to store two different distance readings at slightly different times. It then checks to see if they are different by 10 or more. If so, motion has been 'detected' and it will beep. See the attached CricketLogo reference for the details of global variables.

Now, attach the clap sensor and a motor and try the following program:

```
to clap-test
  when[clap?][a, onfor 5]
```

end

Play with the dial to adjust the sensitivity of the device.

Now, design a way to display sensor information in a way suggestive of the information itself or another quality related to measured factor. Feel free to use the MIDI board here, as there is a long heritage of mapping sensor information to music. See the list of bus-device commands for the MIDI commands.

Here is one to get you started: It uses the Tri-color LED bus device and two sensors control the red and blue values of the LED.

```
to sensor-display
  loop[ cLED sensora 255 sensorb]
end
```

(The function cLED controls the Tri-color LED, with three arguments- the red, green, and blue value to display).

Artists and their work to view and discuss:

Amy Young - <http://www.ylem.org/artists/ayoungs/index.html>

Rania Ho –

<http://www.ok-centrum.at/english/ausstellungen/cyberarts00/ho.html>

Activity Three: Time

An interesting ability of the Cricket is its ability to keep track and respond to the date and time. This ability gives us the opportunity to create work whose behavior changes by date or time, evolves, or ages. This activity centers on a creation whose behavior changes during the course of the day. The clock needs to have constant power in order to keep the time, so make sure the backup battery is in place. The following functions are used to control the clock:

```
clock-init                ;get the clock ready to write
to
set-time :hr :min         ;set the time
set-date :day :mth :year :dow ;set the day, month, year,
and day of week

get-day                   ;returns the day
get-mth
get-year
get-dow
get-hr
get-min
get-sec
```

For example:

```
to time-test
  clock-init
  set-time 12 30
  set-date 8 3 02 1
  do-something
end

to do-something
  loop[
    if ((get-time = 24) and (get-dow = 4))[a, onfor 20]
  ]
end
```

Notice that the first program, time-test, is used to initialize the clock. That program then *calls* another program, do-something, that constantly checks the time and reacts at midnight on Wednesdays.

Artists and their work to view and discuss:

Bruce Cannon - <http://home.attbi.com/~brucecannon/>

Ken Feingold - http://www.kenfeingold.com/docs/KF_01_2002.pdf

Activity Four: Communication

A unique ability of the Cricket is its ability to communicate to other Crickets via infra-red light. The primitive *send* sends a number (0-255), while the primitive *newir?* returns true if there has been a new IR reception, and the primitive *ir* returns the number received (and also sets *newer?* back to false). With only two Crickets talking back and forth, you do not have to be too concerned with protocol. One Cricket sends a number that represents a command to the other Cricket, which is expecting the number and knows what to do when it gets it. However, in order to take full advantage of this ability with many Crickets, we need to come up with a protocol to provide a bit more order.

Master/Slave

One possible protocol relies on having one Cricket being the 'master' and every other Cricket taking commands from it (the slave). A slight variation of this has each Cricket receive, execute, and pass on commands...acting at first like a slave and then like a master. For each of these protocols, a Cricket needs a unique identity and each command also needs a unique ID.

A sample program:

```
;setup variables to store ID, command
global [identity command1 command2 ir_val]

to setup
  setidentity 1           ;set identity to 1
  setcommand1 101       ;arbitrarily assign 101 to
                        ;command 1
  setcommand2 102
  receive_command      ;jump to new procedure
end

to receive_command
  when [newir?][        ;interupt on new ir
    if (ir = identity) [ ;are they talking to us?
      waituntil [newir?] ;get the next ir
      setir_val ir       ;save it
      if ir_val = command1 [do_this]
      if ir_val = command2 [do_that]
      send identity + 1   ;pass it on
      wait 5
      send ir_val
    ]
  ]
end
```

Who starts this process? What happens at the end? Can you get it to repeat itself?

We will now use the first protocol to create a collaborative work. Each person is responsible for creating a surprising or dramatic behavior for their particular piece that happens when their identity is received. Using the above example of code as the basis, everyone must choose an identity (no repeats!) and write two procedures (*do-this* and *do-that*) that will execute depending on whether the person before you sends you a 101 or a 102. The person with identity 1 will start the process. Make sure that the person after you is within the line of sight so that the IR signal will be received successfully.

Tips:

The primitive *newir?* reports a 'true' if there has been a new ir value come in since the last time you checked the value of *ir*, in a statement such as `if(ir = 2)`. In other words, checking the value of *ir* clears the state of *newir?*. Store the value of 'ir' into a global if you want to use its value more than once, as it might change if another Cricket is sending you more numbers.

Here is another master/slave example- figure out what it does!

```
global [identity ir_val]
```

```
to master
  send 63
  wait 5
  if (newir?) [
    setir_val ir
    repeat ir_val [beep wait 5]
  ]
end
```

```
to slave
when[newir?][
  if (ir = 63)[send identity]
]
setidentity 5
end
```

Artists and their work to view and discuss:

Simon Penny - <http://www.telefonica.es/fat/vida2/alife/apenny.html>

Eduardo Kac <http://www.ekac.org/dialogical.html>

Activity Five: Organism and Machine

In this activity, we will explore two modes of interaction: organism-like and machine-like. We will construct two separate works, one that attempts to mimic organism-like interactions and one that behaves in a machine-like way. This area is difficult as there are many complexities to grapple with. Artist Alan Rath, on the topic of behavioral sculpture, said, “what is ‘interesting’ behavior lies between doing nothing and randomness.” Here is a very simple example of something machine-like that takes advantage of the computers ability to store and recall information quickly.

The Cricket can store 2500 points of data. There are 1440 minutes in a day- lets keep track of the light level once a minute all day long, then play it back in less than a minute!

```
to record-light
  erase 2500      ;erase all data
  repeat 1440 [record sensora wait 600] ;record light
values
end

to play-light
  resetdp
  repeat 1440 [cLED 0 0 recall]
end
```

(To try this out, use a light sensor in sensor port A and a Tri-Color LED for playback. Remove the *wait 600* and run *record-light*. Then run *play-light*.)

Artists and their work to view and discuss:

Jenn Hall - <http://www.dowhile.org/physical/projects/acupuncture/index.html>

Simon Penny –

<http://www-art.cfa.cmu.edu/Penny/works/stupidrobot/stupidrobotcode.html>

Edward Ihnatowicz -

<http://members.lycos.co.uk/zivanovic/senster/index.htm#The%20Senster>

Activity Six: Connecting to a Computer

The Serial Bus Device allows us to send and receive information with a computer. This allows us to send sensor values to control onscreen graphics or have the computer send commands to control the Cricket. Plug in the serial board to the Cricket, and connect it to the computer via a serial cable. In the below example we will use the Proce55ing graphics environment to receive sensor values from the Cricket and manipulate graphics based on these numbers. However, various other applications like Director (video) and Max MSP (sound) can receive and handle serial data as well.

Run this program on your Cricket:

```
to serial-test
  loop[
    send-serial sensora
  ]
end
```

Now, shut down CricketLogo open up Proce55ing, click run, and open up Sketchbook/Standard/SimpleSerialDemo.pde. Press the right triangle to run this program. If all is successful, the square onscreen should change its hue based on the sensor values.

Artists and their work to view and discuss:

Kenneth Rinaldo – <http://www.ylem.org/artists/krinaldo/index.html>

Odds and Ends

Where to get motors:

<http://www.goldmine-elec.com/>

<http://www.allelectronics.com/>

<http://www.mpja.com/>

<http://www.sciplus.com/>

Old washing machines, dryers, coffee grinders, disk drives, toys, computer fans...try running them from the Cricket at first, then from the Big Motor Bus Device. Look on the motor for voltage and current ratings printed on the motor. Also, the Cricket can control AC motors with the use of a relay. Buy one, and read about it. Use the motor port at full power (*setpower 8*) to flip the relay. Be careful with AC, and only work with someone who has done something similar before.

Look for 'gearhead' motors for slow, powerful DC performance.

A little about Analog and Digital:

At the simplest level, analog means a system that uses continuously variable voltages to relay information, while digital means that voltages are 'rounded off' to a high level and a low level (in the Cricket and a lot of other systems, this is 0V for low and 5V for high). As said before, the sensor ports on the Cricket measure an analog voltage. The Cricket then converts this analog voltage to a number value (the familiar 0-255). This process is called A/D conversion (for analog to digital).

Digital systems pass information by codes consisting of highs and lows. Remember high school math, when you learned about base-2 numbers? That is finally important! Numbers in base two (binary) are represented by 1's and 0's. This is the core of digital systems; numbers are passed around in base two, where the zero's are the low (0V) and the ones are the high (5V). The Cricket speaks digitally to the bus devices. The Cricket communication system is based on codes consisting of 8 ones or zeros. The ones and zeros are referred to as bits, and 8 bits is called a byte. Back in decimal representation, 8 bits (1 byte) are capable of representing a number between 0 and 255. Aha! The explanation of the mysterious 255!

Powering a Cricket from the wall socket:

To rid yourself of the need for batteries, you can build a power adapter for your Cricket. Go to Radio Shack and buy a 5V to 12V AC-to-DC adapter and a 9V battery clip (looks like the top of a 9V battery with a red and black leads coming from it. With the adaptor unplugged, cut the end off and strip the leads. With the leads separated as not to touch each other, plug the adaptor in and figure out with lead is positive and which is negative with a voltmeter. (Ask someone to help you here, especially if they own a voltmeter!).

Solder the negative end to the lead from the clip that is connected to the flanged side of the 9V connector. (It is probably the red one, although don't trust me here. Why red? The 9V battery clips are usually meant to clip a 9V into, not to take the place of a 9V! So, what we expect to be black (negative) is red (positive).) Solder the other lead of the adaptor to the lead connected to the smooth connector on the clip. Now, with the multimeter, make sure that the connector is the same as a 9V battery (flanged connector is negative, smooth one positive)! Be careful here, as a mixup here might kill your Cricket. With that confirmed, tape up your solder joints and plug it in!

Making your own sensors:

The Cricket can read a value between 0V and 5V. If you have a sensor that puts out an analog voltage between these values, you can plug it directly into the Cricket. The first slot up from the edge on the sensor port is the input. The second one is the ground. Plug ground from your sensor into ground of the sensor port (2nd slot up), and signal from your sensor into the input (1st slot up). However, many sensors are *resistive* sensors, which means that their resistance changes with environmental factors. The light sensor is an example of this, as its resistance changes with the amount of light hitting it (try it with an ohmmeter). For these types of sensors, we plug one end of the sensor into the 3rd slot on the sensor port, which is held steady at 5V. The other end then goes into the input (the 1st slot). The sensor port has a built in *voltage divider* that converts a variable resistance to a variable voltage when wired in this fashion. Try finding a *force sensitive resistor* or *bend sensor* at Radio Shack and using in this manner.

Things you know already:

This section provides analogies in CricketLogo to words you might come across in other programming languages:

- *Procedures, functions, methods*

These are segments of code that perform some function. We have been calling them procedures...anything that starts with a *to* and finishes in *end*.

- *Variables*

A variable allows for storage and manipulation of numbers in an abstract form. In CricketLogo, these are: *global [variable1 variable2]* where variable1 and variable2 are variables, of course.

- *Recursion*

Recursion is the method of calling a procedure from inside that same procedure.

```
to test
  beep
  test
end
```

Or better, but slightly more complicated:

```
global [variable1]

to test2
  recurse 1
end

to recurse :n
  setvariable1 :n + 1
  display variable1
  recurse variable1
end
```

-Arguments

A procedure can take one or more arguments as inputs or output one argument. Below, :n is the argument.

```
to beep-thismany :n
  repeat :n [beep wait 2]
end
```

To use this procedure, we write:

```
to test
  beep-thismany 12
end
```

- Conditionals or Condition checking

Checks a Boolean condition. Allows events to be conditional.

```
to check1
  loop [if (switcha) [beep]]
end
```

or:

```
to check2
  loop[if (sensora > 200)[beep]]
end
```

- Commenting and Reuse of Code

Commenting (text that isn't part of the code, but explains what a piece does) allows for other people to understand what your code does. Compartmentalizing code and using

arguments helps make your code reusable. In CricketLo go, comments are started with the ‘;’.

- Threads/Multitasking

Multitasking allows the computer to perform several tasks simultaneously. In CricketLogo, our only multitasking ability is the *when* command.

```
to thread          ;interrupts the loop to beep when condition
is true
  when [switcha][beep]
  loop[a, on wait 4 a, off wait 5 rd]
end
```

Useful books for going further in electronics, robotics, and sensors:

Robotic Explorations: A Hands-on Introduction to Engineering
Fred Martin
Addison-Wesley
ISBN: 0130895687

Electronic Circuit Guidebook: Sensors
Joseph J. Carr
PROMT Publications
ISBN: 070610981

McGraw-Hill Benchtop Electronics Handbook
Victor Veley
McGraw-Hill
ISBN: 0070674965

TAB Electronics Guide to Understanding Electricity and Electronics
G. Randy Stone
Tab Books
ISBN: 0070582165

The Art of Electronics
Paul Horowitz, T. Hayes
Cambridge University Press
ISBN: 0521377099

Handbook of Modern Sensors
Jacob Fraden
Springer
ISBN:1563965380

Interesting reads regarding interactive sculpture:

Beyond Modern Sculpture
Burnham, Jack
George Braziller, New York

Art of the Electronic Age
Popper, Frank
Thames and Hudson
ISBN 0-500-27918-7

Appendix B



the bus-device to think with

The following activities are designed to provide an intermediate or expert Cricket user with useful concepts in electronics and microcontroller programming. An introductory text of electronics is a necessary companion to this guide, such as the McGraw-Hill Benchtop Electronics Handbook or Practical Electronics for Inventors. Use these books to fill in the gaps or to learn more about a particular subject. A *Bus-Device to Think With* is needed, of course, along with a copy of the BDTTW Edition of Jackal. A digital multimeter is very useful and can be purchased cheaply at www.mpja.com. For users that push further into microcontroller programming, a few more resources are needed: A basic text of the C programming language, a CCS C compiler, and a Microchip PIC Start programmer. Users committed to the exercises presented here and willing to spend many hours hovering over solder fumes or in front of glaring CRTs may find themselves soon programming microcontrollers and even designing their own Bus Devices for the Cricket system.

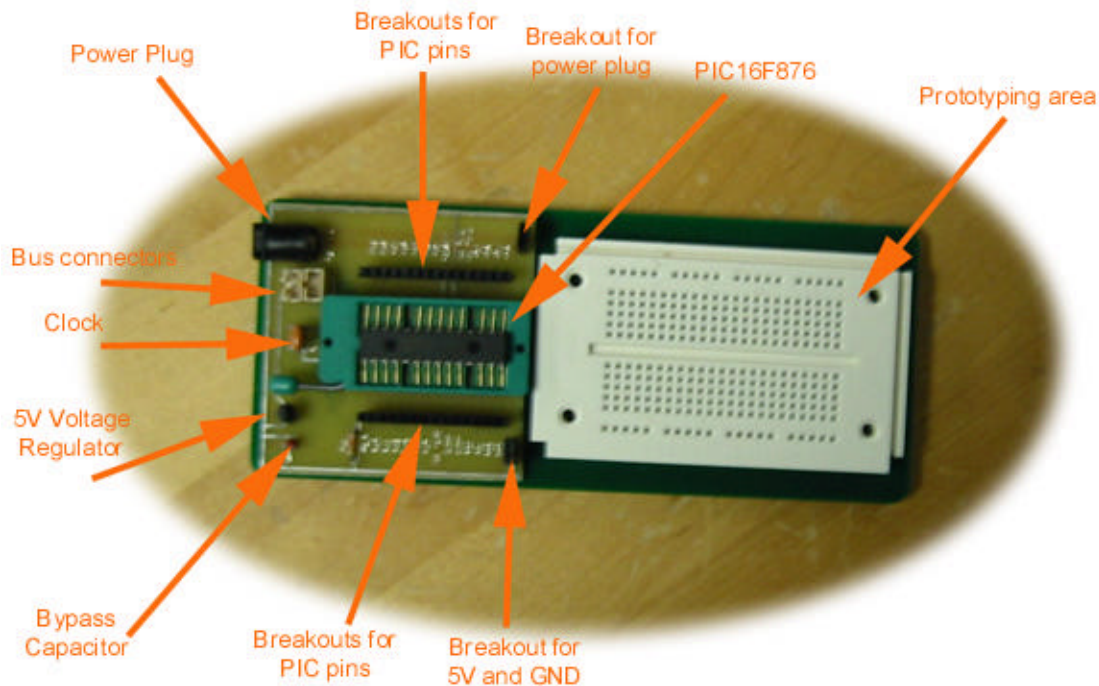
Modern microcontrollers are quite capable and flexible devices. The MicroChip PIC16F876, used in the Bus-Device to Think With, has 22 I/O pins, two PWM modules, three timers, a RS-232 port, and five analog-to-digital converters. The following exercises will discuss each of these capabilities and guide the user through their use. The goal is to provide enough experience with the basic concepts in microcontrollers and electronics¹ to let the user begin playing with ideas and to initiate them into the enormously helpful community of artists and engineers using these materials in their work. This is a heavy document that glances off many complex topics. Do not expect to understand all of it, but do not underestimate the time and commitment that these topics deserve. Most of all, build something fun with these exercises. It makes it all worthwhile.

Depending on your experience level, you may want to read up on the basic concepts in electricity and electronics (try to understand the concepts of voltage, current, resistance, circuits, capacitors, and the relationships between them). The activities that follow

¹ One important skill to master is the ability to read electronics specification sheets. One must wade through endless technical nonsense when all one really wants is a simple explanation. Included with this manual are spec. sheets for almost every component used. For some circuits in this manual, you might even have to read them.

assume little previous knowledge, but they also assume that you have some outside resources and motivation to seek out additional ones as needed.

The Bus Device to Think With

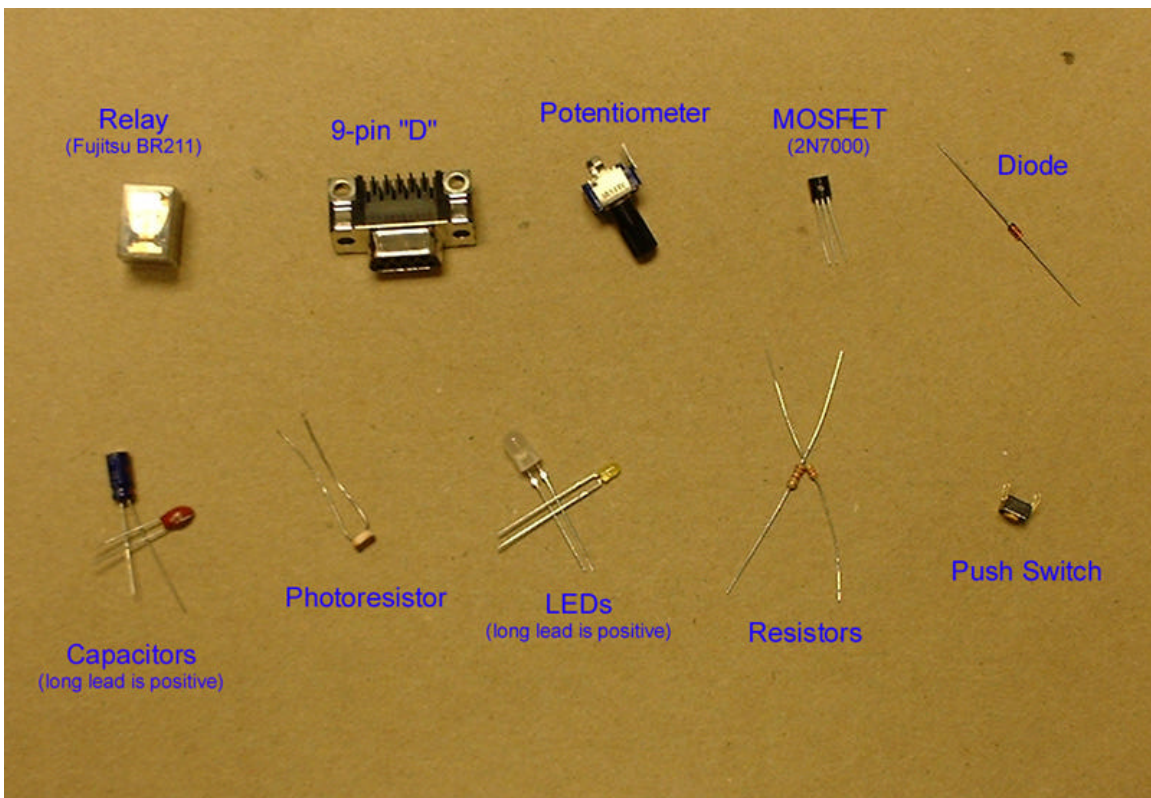


The Bus-Device to Think With is a device that allows you to play with the functionality of a PIC microcontroller through the familiar Cricket Logo language. The code running on the PIC16F876 is written in C for the CCS compiler (<http://www.ccsinfo.com/picc-referall.shtml>). It should be studied and tinkered with once you are comfortable with the basic functionality of the PIC. It contains two functions to handle communications with the Cricket, and a large table of commands. It compares the instruction sent from the Cricket to this table, then executes that function. Whenever possible, the CricketLogo commands are named and used similarly to the C commands, hopefully making the C code more understandable for the Cricket user.

The board itself contains only parts needed for the PIC to run (clock, MCLR resistor), for power (5 volt regulator, bypass capacitor), and for Cricket communication (bus headers). The PIC is powered off of the power supplied by the Cricket Bus line. The 5V regulator cuts the 9V of the Cricket down to a safe 5V, and the bypass capacitor reduces the noise in the power supply. The bus headers contain the 9V power, ground, and the bus signal line. This signal line is connected to the RB0 (discussed later) on the PIC.

The board schematic is included in this document for your use. *Take notice to the breakout plugs on the bottom, labeled +5 and GND. These are your sources of 5V and ground whenever a circuit diagram calls for them. The external supply plug is not connected to anything other than the other two breakout plugs on the lower right side of the board. These are useful if you want to use a different voltage supply for the circuit you build, such as 9V, 12V, or 24V.*

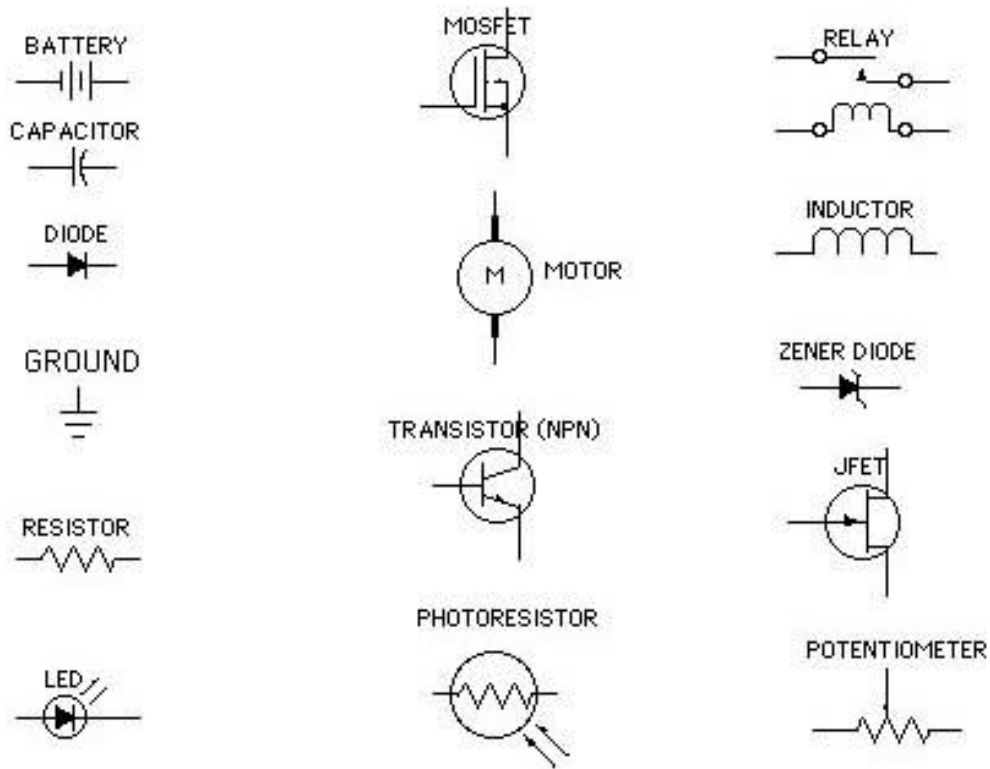
Parts:



All parts and data sheets from www.digikey.com.

Additional parts included: 9V motor, slide switch, multiple stripped wires of different lengths

Common symbols used in electronic schematics:



These parts can be found at any Radio Shack or online at www.digikey.com.

Activity One: LED array!

The PIC16F876 has 22 *pins* available for use as an input or an output (I/O), divided among three *ports*. A port is a group of pins, intended to give the user control over an entire collection of pins at a time. These ports are names A, B, C. For instance, pin 5 of the PIC16F876 is named RA4, meaning pin 4 of the A port, while pin 12 is named RC1, meaning pin 1 of the C port. Don't think too hard about this, its simply an imposed order. Before a pin can be used, we must tell the microcontroller whether it is an input or an output. We will start by making the pin RA2 an output with the function *bit_clear* \$A2². The name *bit_clear* has origins that we will discuss later, but now is not the time. For now, understand that *bit_clear* makes a pin an output (capable of producing 0 Volts or 5 Volts) and *bit_set* makes a pin an input (capable of reading 0 Volts or 5 Volts). Once the pin is set as an output, we can set its value low (0 volts) or high (5 Volts). This is accomplished with the function *pin_set*, which makes the pin high, and *pin_clear*, which makes the pin low. The simplest function looks like:

```
to turn_a_pin_on_briefly
    bit_clear $A2
    pin_set $A2
    wait 10
    pin_clear $A2
end
```

Lets add a LED to make this more visually exciting. We know that pin A2 will be set at 5 Volts. Most LED's will draw too much *current* at this voltage, so we will add a resistor to our circuit that will act as a *current limiter*. This will bring us to the first equation of the activity, the famous Ohm's Law:

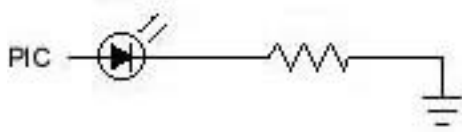
Voltage = Current * Resistance ($V = I * R$) (Units: V = Volts, I = Amps, R = Ohms)

We know that pin A2 will be at 5 Volts. This forms the left hand side of the equation. LED's are rated by the amount of current they can draw so that they don't burn out. The included LED's are rated at 20 mA. This is the current term. Then, we can assume that the LED itself has negligible resistance and solve for R.

$$5 \text{ V} = 20 \text{ mA} * R$$
$$R = 250 \text{ Ohms}$$

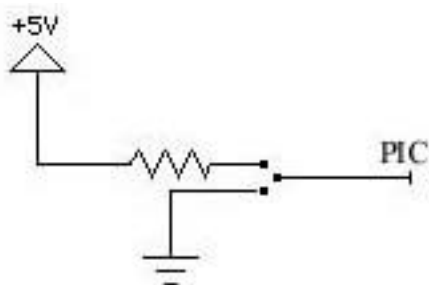
² What's the \$ sign all about? See appendix on Hex numbers.

The schematic of the circuit you will build looks like the following:



Now, to turn this schematic into a circuit: On the prototyping area (known as a breadboard), use the small segments of wire included in your kit to connect pin on the PIC (A2 in this case) to the long lead (positive side, called the *anode*) on the LED. Then take the resistor (in this case, 250 Ohms) and connect to short side of the LED (negative side, called the *cathode*) to the ground (Ground can be found on the bottom left corner of the board. One header is 5V, the other ground. If you look closely, they are labeled as such). After building this circuit, run the CricketLogo program. For more information on using a breadboard and creating circuits from schematics, please see the appendix on this topic.

Next, we will add a slide switch that will signal the microcontroller to turn on this LED. To do this, we will make pin A3 an input to check the state of our switch. We also will choose an input of 0V to be off and an input of 5V to be on, but this could be flipped. Once configured, the status of an input pin is checked using the *pin_test* function. This function returns a 1 at 5V and a 0 at 0V. This is the convention used in most digital systems (1 = high voltage level, 0 = low voltage level). Before wiring the below circuit, use the multimeter³ to probe the pins of the switch. To do this, use the resistance measuring function of the multimeter (called an *ohmmeter*, because it measures in ohms) to check which pins are connected with the switch in a given position, and which ones are connected when the switch is in the other position. Drawing a diagram of the switch might help. The pin that gets connected in either case will go to the PIC (probably the center pin). One of the remaining pins will go through a resistor to 5V, the other to ground.



³ See appendix for more details of using a multimeter

We write this function:

```
to lil_switch_program
  bit_set $A3                ;set A3 as an input
  bit_clear $A2             ;set A2 as an output for LED
  loop[
    ifelse ((pin_test $A3) = 1)
      [pin_set $A2]
      [pin_clear $A2]
  ]
end
```

Try it out!

As a challenge, figure out how to safely use the push button to turn on the LED. Hints: There should be a resistor between 5V and any given path to ground or the PIC. Probe the switch...unlike the slide switch, you only have two pins which get connected on push, and nothing which is connected otherwise. This circuit is included in the appendix .

Now you are own your own. Go crazy- you have 20 pins on 3 ports to use any way you wish. Warning: Four pins on the PIC16F876 are special: **1.** Pin A4 doesn't have the ability to go high by itself. We must 'tie it high', as they say. Here is how it works: We place a large resistor (~10 kiloOhm) between the pin and 5V. This will allow the pin to go into its high state. However, when the pin is set low at 0V, the resistor is large enough that not much current flows. In fact:

$$5V = I * 10 \text{ kOhm}$$

$$I = .0005 \text{ amps}$$

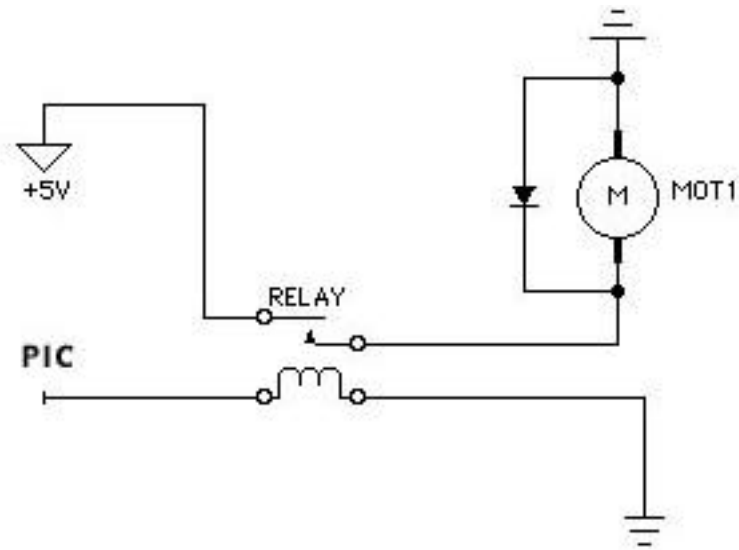
2. Pin B0 has been designated to serve at the communication line between the PIC16F876 and the Cricket. In fact, every Cricket and Bus Device use pin B0 as the communication pin. It is more than just protocol, however. Pin B0 is an 'interrupt' pin, which means that a change of voltage level will cause the microcontroller to pause the process that it is currently doing and execute some other functions. For instance, when a Cricket sends a command to a Bus-Device, the microcontroller on the Bus-Device pauses what it is doing to listen to the Cricket. It then either carries out a different task or continues on doing what it was doing before, depending on what the Cricket told it to do. This interrupt ability is a special trait of B0. Note that if you set B0 as an output, it can no longer listen to the Cricket and the microcontroller will no longer respond to the Cricket. You must then turn off the Cricket to reset pin B0's state, as it needs to be an input. Try it...I dare you. **3&4.** I have reserved RC4 and RC5 for serial communication. Normally, these pins are fully functional; however, the code residing on the PIC right now reserves these for the RS-232 exercise later on. These pins may not function properly for normal I/O as a result and should be avoided.

Activity Two: Relays!

Try the below circuit to turn on and off a motor with a *relay*. A relay is a switch turned on and off by a voltage applied to the *coil* of the relay. Basically, current flowing through the coil produces a magnetic field that closes two pieces of metal together, thus completing a separate circuit. This separate circuit contains the load to be switched, in our case a motor, and a power supply. The battery in the diagram can be any direct current source, the motor can be any load that falls within the specifications of the relay used. Remember, a PIC pin can only source 20mA, so choose your relay to fit within that specification. Also, make sure that the motor and power supply are compatible. The diagram below may be a bit confusing. Let's divide it into two parts: The first is the coil, the little loop-dee-loop thing. Find this part on the relay spec. sheet. Wire one side to the PIC, and one to ground. When you run the program below, you should here the 'click' sound of the switch flipping over.

```
to turn-on_and_off
  bit_clear $A2
  pin_set   $A2
  wait 20
  pin_clear $A2
end
```

Now, let's look at the other part of the schematic, which is the switch. Your relay has one pin that is the *lead*, one pin that is *normally open*, and may have one pin that is *normally closed*. Without current through the coil, the *lead* is connected to the *normally closed*. With current flowing through the coil, the *lead* is connected to the *normally open*. With schematic in-hand and multimeter fired up, verify this. Then wire up the rest of the circuit as shown below. The pins shown in the diagram are the lead (with the little arrow) and the normally open.



The diode that bridges the motor is to protect the components from harmful voltage that builds up when the motor switches off. This is because motors have a high *inductance*.

Relays are just a switch, making them useless if you want to control the speed of a motor or dim a light. However, they are very useful at switching AC loads or very large DC loads. We will control the speed of a motor in activity 4.


```

output_c    85          ;(1111111111 in binary, $FF in hex)
                ;turn on even pin numbers, odds off
                ;( 01010101 bin, $55 hex)
end

```

Hook up the bi-color LED, through a resistor between pins A0 and A1. (That is, run a wire from A0 to the bi-color LED, then a resistor from the other pin on the LED to A1.) Then try this program:

```

to tris_example
  set_tris_a 0
  output_a 1
  wait 10
  output_a 2
  wait 10
  loop [ output_a 1 output_a 2]
end

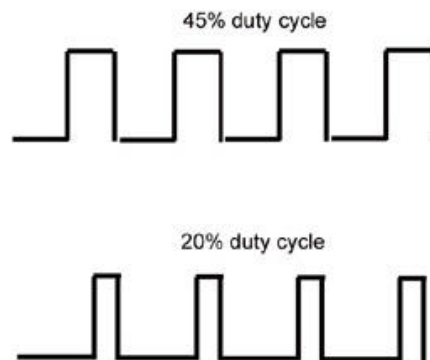
```

This concept takes a while to fully wrap your head around. It is a trick, mostly. It takes advantage of a way of representing numbers and uses that representation for a completely different reason, like setting up pins. There is no reason that tricks like that should make sense- you just accept them at first and then appreciate them later.

Activity Four: Motors and MOSFETS!

In this activity, we will use one of two available Pulse Width Modulation (PWM) modules to power and control a DC motor. Later, we will also use this module to produce a continuously variable DC analog signal (from 0-5V).

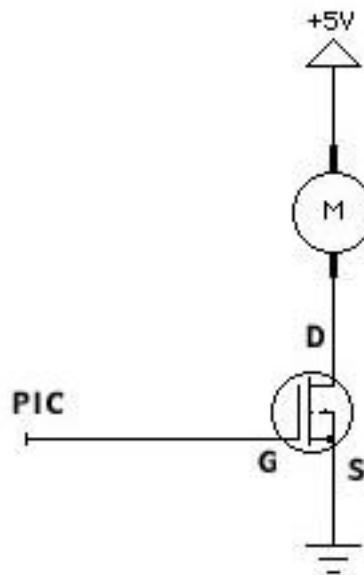
Pulse Width Modulation refers to a continuous series of square pulses produced by a pin on the PIC. The width of the pulses is variable, as illustrated below. The *duty cycle* of a PWM signal is the amount of time the pin is held high in each cycle. For instance, a duty cycle of 50% means the pin is low for the same amount of time as it is high, while a 10% duty cycle means that the pin is high for 10% of the time. A PWM signal also has a frequency, which is the number of high-low cycles per second. This frequency is not important in our examples. A PWM signal fed into a motor effectively turns the motor on and off. However, this is happening very fast when compared to how long the spindle of the motor takes to slow down. In the end, the motor ‘averages’ the highs and lows of the PWM signal, and its speed is proportional.



DC motors of reasonable size (big enough to do something useful with) are power-hungry devices. They draw anywhere from 100 mA to 10 A or more. A pin on our PIC can only *source* 20 mA. While the relays used in Exercise 1.5 can turn a motor of this type on, the mechanical switching action of the relay is not fast enough to keep up with the high-speed PWM signal. For this exercise, we will use a high-current MOSFET (metal oxide field effect transistor, not that it matters) that acts just like a relay, but can switch faster since it does not use moving parts to switch. The amount of current that can run through a MOSFET depends on the MOSFET, but we will use .4 A devices. Note that the pins of the MOSFET are labeled as D,G,S (in the spec. sheet). These stand for Drain, Gate, and Source, respectively. The Drain is the *drain of electrons*, or where the electrons flow to, which is the positive voltage supply. The source is the source of electrons, which is the ground. This is confusing! Current flows from positive voltage to ground, right? Yes, but only because current was defined wrong by Benjamin Franklin way back when. Current does flow from higher voltage to lower voltage (by definition), but what is actually flowing are electrons (from lower voltage (or ground) to higher voltage). To avoid confusion, always think of current flowing from high to low, except when reading spec. sheets and seeing the terms *drain* (or *V_{dd}*, which is the drain voltage)

and *source* (V_{ss} , source voltage). Don't blame Ben. He knew something was flowing and guessed wrong. He did bring us bifocals and chimneys, after all.

The PWM module on the PIC16F876 relies on an internal timer, specifically Timer2. The first step in using the PWM module is setting up this timer. We use the function *setup_timer2* with a 1,2 or 3 passed as an argument. Passing a 1 will setup the timer near its fastest rate and 3 slowest, with 2 somewhere in between. Verify this with an oscilloscope once the PWM module is running. The next step is turning the PWM module on. The PIC16F876 had two modules, PWM1 on RC2 (pin 13) and PWM 2 on RC1 (pin 12). The functions to call are *setup_pwm1* or *setup_pwm2*. Then, we set the duty cycle of the module. The function is *set_pwm1_duty* or *set_pwm2_duty* followed by an argument 0-255. One would imagine that the duty cycle should be 0-100. I would think that too. Microchip doesn't. Use 0-255. The circuit is sketched below:



```
to motor_program
  setup_pwm2
  setup_timer2 1
  set_pwm2_duty 10
  wait 10
  set_pwm2_duty 50
  wait 10
  set_pwm2_duty 150
  wait 10
  set_pwm2_duty 200
end
```

After successfully driving the motor, remove it and the MOSFET. Place a 100 μF capacitor (mind the polarity) in its place, with the positive side of the cap on the PWM

line and the negative side to ground. Write a CricketLogo program to sweep the duty cycle from 0 to 255 repeatedly. While running this program, measure the voltage level of the positive side of the capacitor⁵ with a voltmeter or oscilloscope. Analog out! Neat trick, although it turns out it isn't necessarily useful in most circuit designs. Some interesting things *might* be possible, as there are voltage-controlled amplifiers and filters used in musical synthesizers that might be interfaced to a PIC using this technique. One more PWM trick: Replace the capacitor with an LED (wire a resistor and LED from the PWM pin to ground). Change your CricketLogo program to ramp from 0% duty to 30% and back down again. You should see a nice 'heartbeat' from the LED.

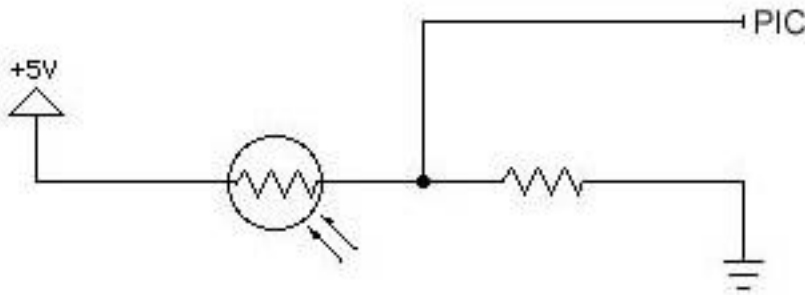
⁵ The value of the capacitor here should be played with. Too small of a capacitor and you will still be able to see individual spikes of the PWM signal with an oscilloscope. Too big, and it will take a mighty long time to discharge itself. You can also place a resistor in series from the PIC to the capacitor and see what effect it has. Watch the difference on an oscilloscope.

Activity Five: Analog to Digital (A/D) Conversion:

Digital systems have many advantages and have led to many great advances. However, the sensors used to sense or detect an aspect of the physical world measure an analog signal, and we are forced to convert an analog signal to a digital one for it to be useful. Fortunately, this is largely done for us in the PIC. It contains 5 pins (A0, A1, A2, A3, A5) that can be configured to perform direct AD conversion on a signal between 0V and 5V. The functions we will use are `adc_setup` and `adc_read`. The function `adc_setup` gets passed a number (0,1,2,3,5) for the pin of Port A to perform the conversion on. The function `adc_read` returns a 1-byte number (0-255) proportionally to the signal, with 0 representing 0V and 255 representing 5V.

```
to sample-adc
  adc_setup 0
  loop[ display adc_read]
end
```

Many sensors use environmental factors to change the sensor's resistance, such as thermistors (temperature effects resistance), photoresistors (light effects resistance), or force-sensitive resistors (FSRs). To create an analog voltage signal from these devices, we create a voltage divider with the sensor and an ordinary resistor, as shown below.



Lets analyze this circuit. The total resistance from 5V to Ground is the resistance of the sensor (R_s) plus the resistance of the resistor (R)⁶, in our case we will use a 30-50 k resistor. The current flowing through both of these resistors is $I = 5V / (R_s + R)$. Now, since current cannot flow into the PIC (it has very high resistance, or *impedance*), the current flowing through the sensor is equal to the current flowing through the resistor (where else would it go?). We use this fact to see that:

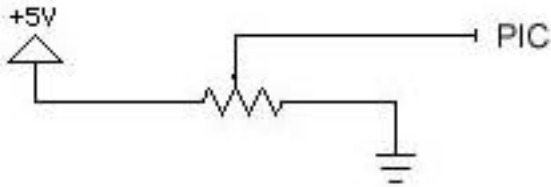
⁶ How do you pick the value of this resistor? You should match this resistor to the change of resistance of your sensor in the conditions you are most interested in detecting. Just be certain that the total resistance at any time is bigger than a kiloOhm or more so that the current flow is small.

$$(5V - V_{PIC}) = I * R_z$$

Solving this for V_{PIC} gives us $V_{PIC} = 5V - 5V * R_s / (R_s + R)$

Note that the voltage at the PIC (V_{PIC}) is proportional to the resistance of the photoresistor (R_s). Just what we wanted! We have our analog voltage signal for the PIC to measure!

There are devices known as variable resistors that have a knob or slider that is used to vary the resistance of the device. Many of these devices are set up in a voltage divider configuration for easy AD conversion. These devices are known as a potentiometer. We use one below with the same AD functions.



The center pin of a potentiometer is usually the *wiper*, or the point of variable resistance. This pin will be connected to the PIC, and the outer two pins to 5V and ground.

Exercise Six: Serial I/O

(Note: this exercise requires the installation of the Proce55ing program included on the CD)

Serial communication is a method of communicating between electronic devices. It refers to the passing of information bit-by-bit in time, in contrast to parallel communication, in which many bits are passed simultaneously. Many different protocols exist for this serial exchange. One very useful and ubiquitous protocol is known as RS-232 and it is used, amongst other things, for communication between computers and their peripherals (although this use is being rapidly replaced by USB protocol). The nice feature of RS-232 is that most of the work is done for us, either in the PIC itself or in the C compiler. We will be using RC4 and RC5 to send and receive.⁷



(Backside of the female D connector)

Setup the photoresistor circuit shown in Exercise Five. Connect pin two of the 9-pin D connector to RC4, pin three to RC5, and pin five to ground. Run the following program on the Cricket.

```
to send-data
adc_setup 0
  loop [
    send_serial (adc_read)          ;send the result of the
A/D conversion up the serial line
  ]
end
```

Once this is running, run Sketchbook/Standard/SimpleSerialDemo.pde within the Proce55ing environment and connect the serial cable to the computer. Proce55ing is an environment and libraries made specifically for graphics programming in Java. Once you have opened SimpleSerialDemo, click on the right arrow button to start. A small square should appear and change from black to white as the light level on the photoresistor changes. Serial data from the PIC might also be used for controlling graphics in Java,

⁷ The PIC 876 has a 'hardware' serial port built in, called a USART (universal synchronous/asynchronous receiver/transmitter built in. However, this feature was meant to work with a *level converter*, which converts the 0-5V signal to -12 to 12V (most computers these days don't need these levels...0-5 works fine). These level converters also invert the signal, which means that the signal coming from the hardware USART is inverted to what the computer wants to see...thus we are forced to not use the hardware USART, unless we want to wire up a level converter. However, this only means slightly larger and slower code on the PIC, and nothing more.

video in Director, or sound in Max. Much information regarding this can be found on the Internet.

We can also receive serial on the PIC from the computer. There are two functions for this, *new_serial?*, which returns a one if there is new serial data and a zero if not, and *get_serial* which returns the new serial data and clears the *new_serial?* flag. An example might look like:

```
global [serial-data]

to serial-test
bit_clear $A1
  when[new_serial? = 1][
    setserial-data get_serial
    if (serial-data = 64)[pin_set $A1]
      if(serial-data = 65) [pin_clear $A1]
    ]
end
```

Moving to Other Microcontrollers

Take some time to build a few things with the BDTTW. Use these projects as excuses to try out some new circuits you have dug up, as well understanding further the functionality of a microcontroller. Dig for information regarding the various microcontrollers on the market, like the Basic Stamp, PIC, OOPIC, etc. Find the one that looks right for you. For those interested in using a Basic Stamp, see the web page cricket.media.mit.edu/basicstamp regarding the use of Bus Devices with the Basic Stamp. For those interested in the PIC, the BDTTW also serves as a development platform for programming and using the PICF876. The section below will take you through that process.

The BDTTW as PIC Development Platform

The first step in this direction is obtain the PICSTART Plus programmer, MPLAB and the CCS PICC compiler. The PICSTART Plus and MPLAB are available at <http://www.microchip.com/>. The CCS PICC compiler is available at <http://www.ccsinfo.com/ccscorder.html>.

Installing MPLAB and the CCS compiler and Using Activites1-6

1. After installing MPLAB on your computer and unzipping the CCS files into their own folder on your hard drive (put the unzipped folder on c:/, so that the path is simply c:/picc), open MPLAB. Choose Project/Install Language Tool. Now, choose CCS and the C-Compiler and then choose the executable PICC\CCSC.EXE (this is in the folder you unzipped, wherever you put it) You also want it Windowed and not Command Line.
2. Now we must tell the computer where to find the header files (the header files are the files specific to the type of PIC we are using that come included with the compiler). To do this (in Windows 2000), we click on the Start Menu and choose Settings/Control Panel. In Control Panel, choose System/Advanced/Environment Variables. Edit the System Variables section labeled Path and add the line c:/picc/EXAMPLES (yes, case matters). Do not erase or change any of the other variables. Now, restart your computer.
3. After copying the folder picc_examples (available on the BDTTW website) to you hard drive, select Project, New. Now find the picc_example folder, and create a new project with the name led.pjt in that folder (the project name should be the same as the source file, in this case 'led' (the source file is 'led.c'). Hit OK. The Edit Project window should pop up. The Target File name should be led.hex. You can leave the other boxes within the Project heading blank. If the Development Mode box does not read Editor Only16F876, click on the Change button. Click the check box labled None (Editor

Only), and find the PIC16F876 in the Processor field. Hit OK. Now, change the Language Tool Suite to read CCS. A box will pop up to warn you that you will lose your 'target command line options'. That's quite fine, so hit OK. Now, highlight the temp [.hex] file in the Project File box. The Node Properties button should now be available. Click on this. Click on the box labeled PCM. It should now be checked. Hit OK. Now, one last step. Click Add Node. Find the file led.c and add it. Hit OK, then back in the Edit Project dialog box, hit OK.

4. Now, we need to enable the PICSTART programming board. Click on the Menu item PICSTART Plus. With the programming board plugged into your serial port, click on Enable PICSTART PLUS.

5. To compile the file led.c, we choose Project/Build All. The program has compiled successfully if the resulting window reads 'Build Completed Successfully'. If there are errors in our code, the window will contain the line number of our error (or perhaps the line before it) and some cryptic error message. The reference for the CCS compiler will explain these error messages a bit better, but not great. Also, since the compilers only good clue to what is wrong with your program is the line number, we must use a text editor that includes the line numbers. I use the EditPlus text editor, available as shareware. Note: After we change anything to the source file, led.c, we must save it and compile it again.

6. After placing the PIC to program in the programmer (depressed dot on the PIC should be at the top of the programmer), we hit Program. The yellow light on the programmer will light up, and a dialog box should read 'success' after the programming is finished. You are ready to rock and roll. Place the PIC back in the BDDTW and power it up (the Cricket powers the BDDTW, so just turn on the Cricket)

7. Now, wire up the LED just as in Activity One. You should see the familiar blink. Note: The Cricket is just powering the BDDTW. We could easily make our own cable to power the BDDTW without the Cricket, but the Cricket is handy so what the heck. Note that in Activities 1-6, the Cricket was running the show, telling the BDDTW what to do. Now the tables have turned, and the PIC is the star.

8. Go through all of the example PIC code provided (Activities 1-6). These programs should function similarly to the CricketLogo versions in the original activities. All of the wiring and circuitry is the same as the original activities. Pay close attention to the *adc.c* file, as in this example we run the Display Bus Device right from the PIC!

There are many web pages and discussion groups devoted to PIC's that you should utilize as you progress. Again, pick a project to focus your efforts and to make all of the frustration worthwhile. Good luck!

Building a Cricket Bus Device

For those interested in building their own bus device, open the *lib.txt* file in the Support folder of the BDTTW Edition of Jackal. Look at the various functions that the BDTTW uses, like *pin_set* or *output_a*. You will notice the function *bSEND*. This function sends a number over the bus line. The first number it sends is \$199⁸. This is the address of the BDTTW. Upon receiving this number, the BDTTW knows the Cricket is talking to it and then listens for more numbers. The next numbers it receives are instructions to do something else, like make a pin an input or set a pin high. This number-passing scheme is how all bus-devices work. Now open up the file BDTTW.c with a text editor such as EditPlus. Find the functions with names similar to the CricketLogo functions (they near the very bottom). These functions are what get executed with the similarly named CricketLogo function is run. Notice that there are a few extra steps in some cases, but are very similar none-the-less. Use this file as a guide to how to structure your bus device code.

⁸ \$199 is bigger than one byte! Ok, there is some slight of hand going on here. The Cricket actually sends 9 bits, not eight, with the extra bit signifying whether the other eight bits should be treated as instructions or data. In this case, the Cricket sends \$99 as an instruction. If we had written *bSEND* \$99, it would send a \$99 as data.

Appendix:

Binary and Hexadecimal numbers:

We use the decimal representation of numbers in ordinary math. However, computers operate with a binary representation. At some point, people realized that a hexadecimal representation was a good compromise between the two, allowing humans to better visualize numbers as the computer might see them, but not make them stare at ones and zeros either. We will briefly cover all three and how to convert between them.

In the decimal system, we count from 0 to 9, then add a place holder in the next space over to represent 10. We then continue counting from 0 to 9 again. In binary, we count from 0 to 1, placing a 1 in the next space over when we hit one in the first. In hexadecimal, we count from 0 to 15 before marking the next space. However, the number 15 takes two spaces! To solve this, we start counting in with letters after nine..ie, count from zero to F, going through 9 to A, then B, and so on.

Decimal: 0,1,2,3,4,5,6,7,8,9

Binary: 0,1

Hexadecimal: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Binary is denoted by a lower-case b, hexadecimal by a '\$' or '0x' prefix or a 'h' postfix.

To convert, its best to use calculator! In time, you will be able to convert back and forth in your head. For now, try out Marin Steen's Hex Calculator at <http://www.martin-steen.de/hexcalc.html> or use a scientific calculator.

Pinout for 9-pin D connector

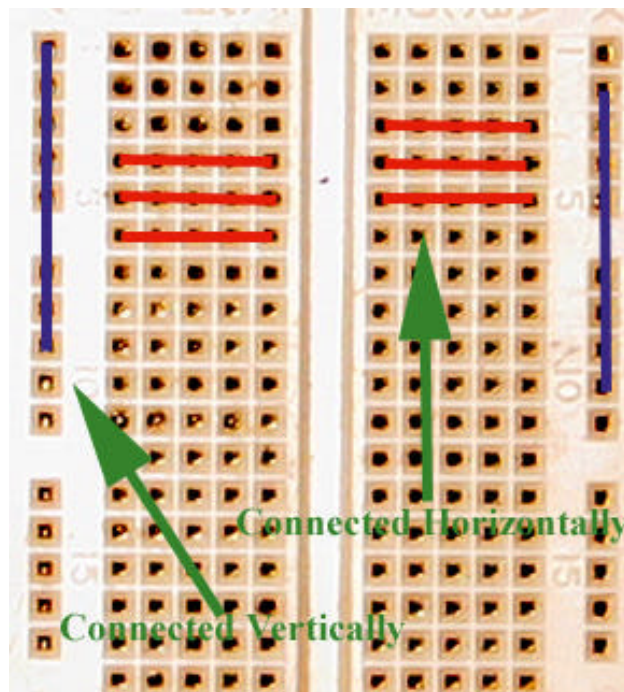
Pin No.	Function	Pin No.	Function
1	DCD (Data Carrier Detect)	6	DSR (Data Set Ready)
2	RX (Receive Data)	7	RTS (Request To Send)
3	TX (Transmit Data)	8	CTS (Clear To Send)
4	DTR (Data Terminal Ready)	9	RI (Ring Indicator)
5	GND (Signal Ground)		

Multimeters

Multimeters are many measuring instruments packaged in one device. These instruments are usually a ohmmeter (measures resistance), a voltmeter (measures voltage), an ammeter (measures current), and sometimes a faradmeter (measures capacitance). Also, some have a continuity tester, which beeps when a connection exists between the leads. This is a handy feature. The leads of the multimeter may or may not have different plugs for the different functions. This is important to remember. Also, the ammeter will certainly have a maximum input current it can accept. Do not ignore this. Also, there might be a switch for AC/DC coupling. Use DC for all purposes in this manual. For probing a switch, use the continuity tester (if possible) or ohmmeter on lowest setting. For measuring voltage, connect the black lead to ground and the red lead to the voltage being measured. For measuring amperage, connect the device in series with the circuit, so that current is flowing in the red lead and out the black. For measuring capacitors, make sure that you discharge them by touching their leads together before inserting them into the faradmeter.

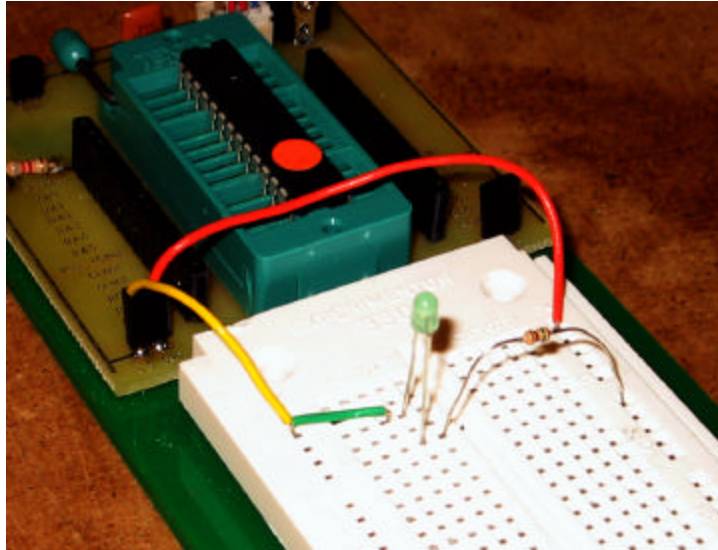
Using Breadboards and Building Circuits

A breadboard (or solderless breadboard) is a device that allows for rapid creation of electronic circuits. Notice that there are two different types of hole patterns. The holes in the center area of the board are electrically connect horizontally. The holes the run down the edges of the board are connected vertically, as shown in the picture below.

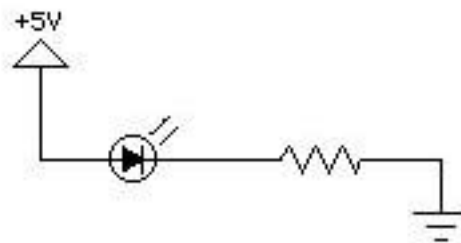


The outer edge holes (sometimes called rails) are usually used for 5V and ground (0V). By plugging in a wire to the 5V header and plugging in the other end to the left hand rail,

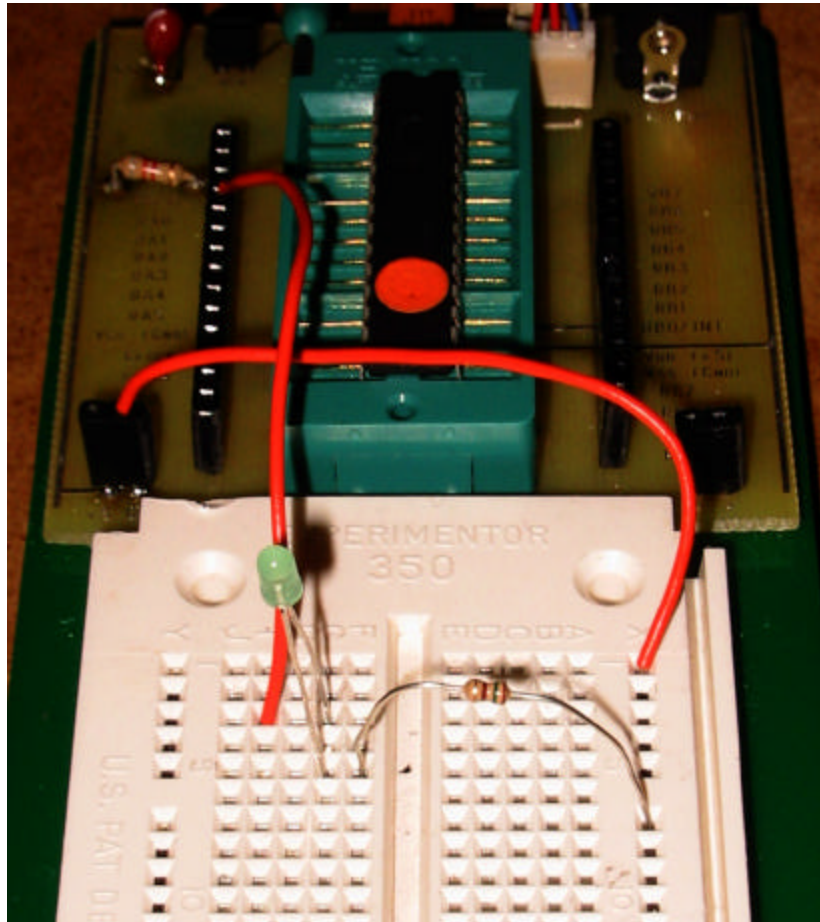
we can now connect 5V into any circuit by utilizing the fact that all of its holes are connected vertically. Do a similar thing for ground. Now, plug a wire into the left hand rail (5V), and insert the other end into an arbitrary hole in the center area. Since these holes are connected horizontally, we will now plug the long end of an LED (the anode, or positive voltage side) into any hole in that row. This lead will now be at 5V. Insert the short end of the LED into another arbitrary hole (anyone except one in the same horizontal row as the first!) Now plug one end of a 100 Ohm-1kOhm resistor into a hole in the same horizontal row as the short end of the LED. The other end of this resistor should now be plugged into the ground rail. Your circuit should look like this:



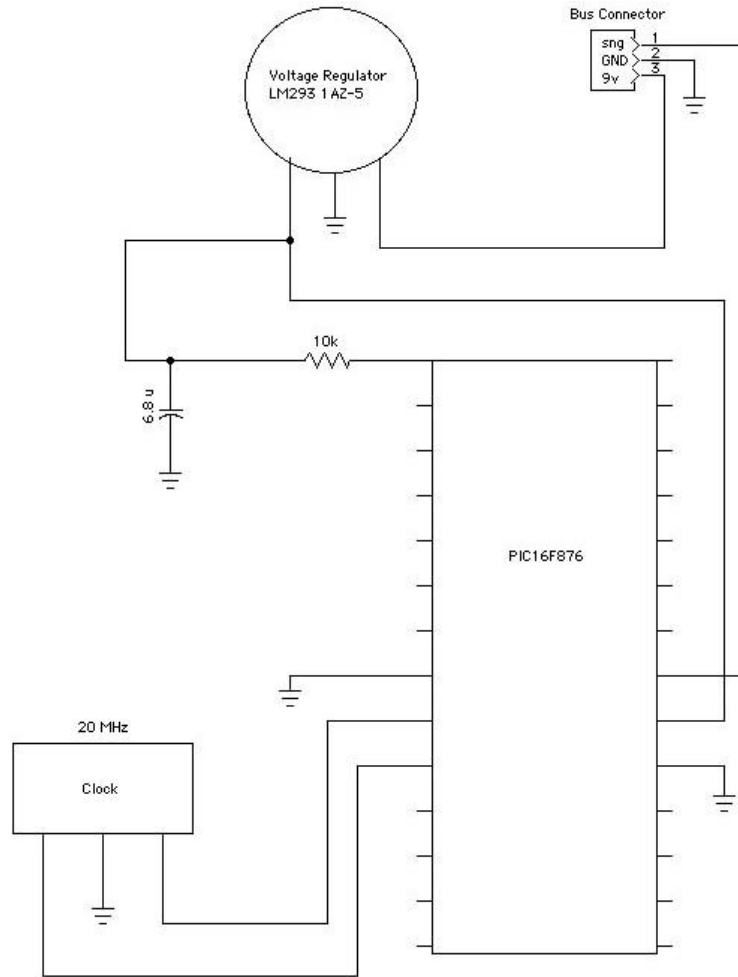
The schematic of this circuit looks like:



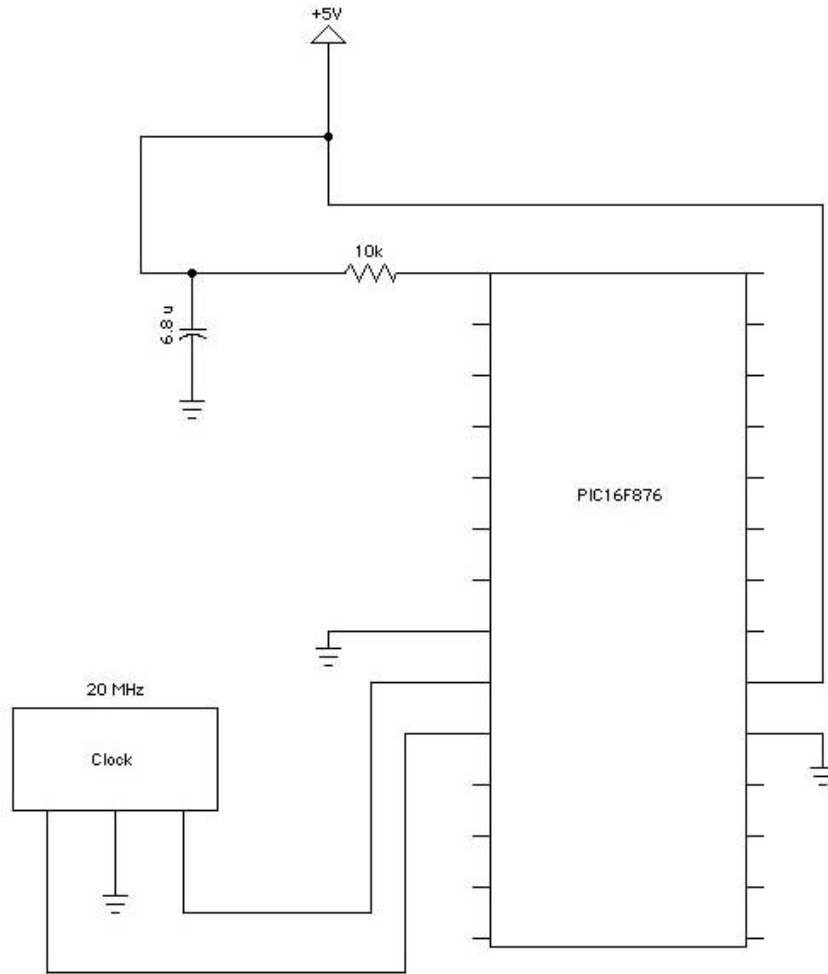
The schematic pictured in Activity One looks like this on the protoboard:



Schematic of BDTTW



Simplest Circuit Needed to Run the PIC16F876



Button Schematic

