MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices

Johnny C. Lee, Daniel Avrahami, Scott E. Hudson, Jodi Forlizzi, Paul H. Dietz, Darren Leigh

TR2004-076 January 08, 2004

Abstract

Toolkits and other tools have dramatically reduced the time and technical expertise needed to design and implement graphical user interfaces (GUIs) allowing high-quality, interative, user-centered design to become a common practice. Unfortunately the generation of functioning prototypes for physical interactive devices as not had similar-support - it still requires substantial time and effort by individuals with highly specialized skills and tools. This creates a divide between a designers' ability to explore form and interactivity of product designs and the ability to iterate on the basis of high fidelity interactive experiences with a functioning prototype. To help overcome this difficulty we have developed the Calder hardware toolkit. Calder is a development environment for radidly exploring and prototyping functional physical inreractive devices. Calder provides a set of reusable small input and output components, and integration into existing interface prototyping environments. These components communicate with a computer using wired and wireless connections. Calder is a tool targeted toward product and interaction designers to aid them in their early design process. In this paper we describe the process of gaining an understanding of the needs and workflow habits of our target users to generate a collection of requirements for such a toolkit. We describe technical challenges imposed by these needs, and the specifics of design and implementation of the toolkit to meet these challenges.

ACM Conference on Designing Interactive Systems (DIS)

The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices

Johnny C. Lee[1], Daniel Avrahami[1], Scott E. Hudson[1], Jodi Forlizzi[1,2]
[1]Human-Computer Interaction Institute, [2]School of Design
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{ johnny, nx6, scott.hudson, forlizzi }@cs.cmu.edu

Paul H. Dietz, Darren Leigh
Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA 02139, USA
{ dietz, leigh }@merl.com

ABSTRACT
Toolkits and other tools have dramatically reduced the time and technical expertise needed to design and implement graphical user interfaces (GUIs) allowing high-quality, iterative, user-centered design to become a common practice. Unfortunately the generation of functioning prototypes for physical interactive devices as not had similar support – it still requires substantial time and effort by individuals with highly specialized skills and tools. This creates a divide between a designers' ability to explore form and interactivity of product designs and the ability to iterate on the basis of high fidelity interactive experiences with a functioning prototype. To help overcome this difficulty we have developed the Calder hardware toolkit. Calder is a development environment for rapidly exploring and prototyping functional physical interactive devices. Calder provides a set of reusable small input and output components, and integration into existing interface prototyping environments. These components communicate with a computer using wired and wireless connections. Calder is a tool targeted toward product and interaction designers to aid them in their early design process. In this paper we describe the process of gaining an understanding of the needs and workflow habits of our target users to generate a collection of requirements for such a toolkit. We describe technical challenges imposed by these needs, and the specifics of design and implementation of the toolkit to meet these challenges.

ACM Classification:
H5.2. Information interfaces and presentation: User Interfaces; Input Devices and Strategies; Prototyping.
Keywords: Toolkits, physical user interfaces, rapid prototyping, interaction and product design.

1. INTRODUCTION

Over the last two decades tools for creating graphical user interfaces have progressed substantially, and many of the early goals of the user interface tools community (such as those described in [4, 15, 14]) have been met. In particular, creation of graphical interfaces has progressed from an activity requiring substantial effort by highly skilled experts, to one that can often be undertaken quickly by people whose primary skill set is not software development (e.g., domain experts and interaction designers). This has been a strong enabler in the widespread adoption of high quality iterative approaches to GUI design and implementation.

As interactive computing begins to take advantage of modern small low-cost processors, the design space for interactive products becomes increasingly larger and increasingly attractive. However, we find that development of such products suffer from many of the same difficulties common in the early days of developing graphical interfaces. Turning an interactive product concept into a functional prototype requires a substantial amount of effort by technically skilled individuals. This makes iterative design with high-fidelity functioning prototypes prohibitively expensive for most.

The *Calder* system is a hardware toolkit designed to extend the benefits gained in the development of tools for creating GUIs into the domain of physical interfaces. In analogy to GUI toolkits, the Calder development environment provides a set of reusable input and output components, an infrastructure for connecting these components (in both wired and wireless fashion), and integration into existing interface prototyping environments.

In this paper we describe the first two steps in a three step process: study of our intended users (product and interaction designers) and design and development of technical solutions to meet their needs. The next section of the paper describes the background study of the intended users followed by descriptions of a set of sample devices built with the toolkit to illustrate its range and versatility. The next four sections then describe the design and implementation of our technical solution. Finally, we present brief conclusions and discussion of the third step in the process – deployment, testing, and iteration – and lay out plans for future work.

Figure 1. Designers often use blocks of foam material to generate sketches of form.

2. SUPPORTING DESIGNERS

In previous work, we conducted a series of interviews with and observations of product and interaction designers that led us down initial paths toward developing lightweight functional components to support the processing of design physical products [1]. This earlier work gave us insight into the needs and demands that would be placed on a fully developed toolkit.

The Calder toolkit is the resulting solution we developed from this earlier work. A central goal of the system is to bring fully functional prototypes into the design process much earlier while still allowing early design to be rapid and fluid. In this section we will detail how observations and feedback from designers affected our design decisions for the toolkit.

In our initial investigation of early stage design we found designers working with two and three-dimensional artifacts to support sketching and prototype activities. These activities have been documented by professional product designers [19, 5], and supported in other tools for early design [10, 12, 2]. In many cases these artifacts begin with literal sketches on paper. However, as concepts progressed and more fidelity was needed (e.g., understanding how a form would feel in the hand), sketching moved to (rapid) creation of solid forms ("3D sketching"), and to on-screen mockups of interactions. Figure 1 illustrates a common practice of roughing out of shapes in foam as a kind of 3D sketching.

We found that sketching and prototyping were used for at least three important purposes:

Design exploration: Creating enough representation of the design so that it could serve as a thinking tool to help make choices and drive idea generation,

Communication: Providing a medium for conveying a design to managers, clients, potential users, and other designers on a team,

Design testing and data gathering: providing a vehicle for understanding user models and reactions, doing early user testing, and gathering other user-centered data.

Overall, we found the sketching and prototyping practices observed to be quite effective within a certain range of activities, but also limiting in others. This led us to a dual approach of attempting to maintain as much of typical current practices as possible, but at the same time provide a means to extend the scope and fidelity of the artifacts that could be developed. Specifically, this introduced two overall driving factors for our toolkit: *introducing and extending executability*, and *preserving fluidity and flexibility*.

Introducing and extending executability: Both interaction designers and product designers often begin with non-functional paper sketches. This enables, for example, quick and easy "pencil and paper tests" [18]. Design of interactive sequences often precedes the use of executable on-screen mockups that allow for more robust communication and testing, although often in a different medium than the final product. However, those working with forms do not typically have an equivalent option, limiting them to the use of non-functional prototypes. Further, this forces exploration of interactivity to remain separate from exploration of form, making it difficult to understand the important relationships between them. A central motivation behind our toolkit work has been to extend capabilities for functional prototypes into the domain of form, and to allow form and interactivity to be treated together.

Simple programming is typically necessary to create functioning prototypes. We found many interaction designers to be quite fluent with GUI prototyping tools, such as Macromedia Director, as a means for on-screen expression of interactive concepts. In accordance with our principle of minimal disturbance of existing practice, we made it a goal to work with existing prototyping tools rather than construct new ones (we currently target support for Macromedia Director, but plugging into Java is also supported and Visual Basic support is being developed).

Preserving fluidity and flexibility: An important hallmark of sketching activities is their fluidity and flexibility – artifacts must be comparatively easy to create and modify so that the rapid iterations needed in early design are not interfered with. This is often achieved in the domain of form by using fairly soft materials such as the foam shown in Figure 1 that can be easily cut with a knife or a band-saw in a few moments. These materials offer some important capabilities that we wanted to preserve. For example, one designer described how he would directly modify a rough form by cutting parts of it away while working with potential end-users. This led us to put an emphasis on supporting work with soft materials – in particular foam.

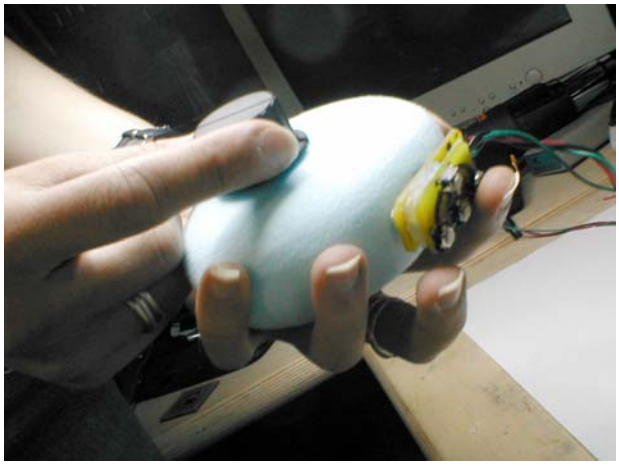

Figure 2. Wireless knob and buttons used within a foam model of a concept game controller.

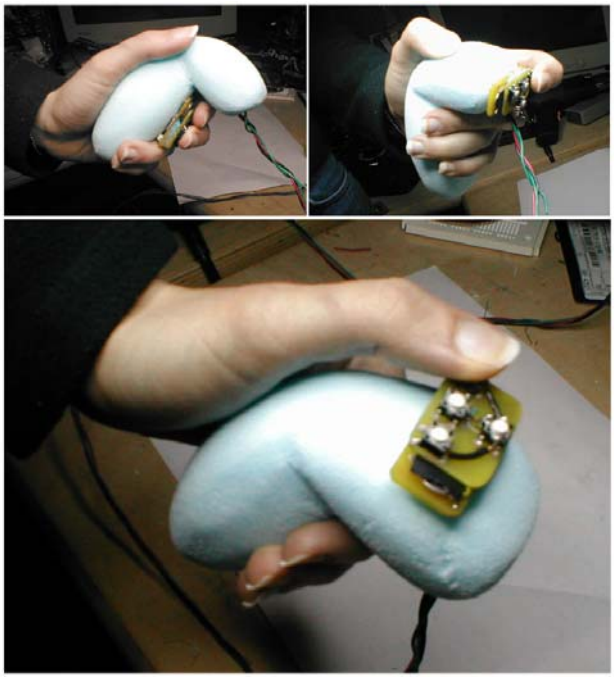

Figure 3. Navigation controller with the wireless buttons placed in a variety of locations around the form.

In order to preserve fluidity, the mechanism for attaching components to the form needed to be carefully considered. Form designers must be able to rapidly try out different placements of components (for example, control buttons), by removing them from the form and immediately placing them in another location, testing the "feel" of the placement, and then possibly moving them again.

To preserve flexibility, the size of components is also quite important. For example, if buttons or other input devices need to be placed close together in the final product, it would be helpful if this could be done in the prototype as well. This creates a technically challenging, but strong imperative for small size. This push for small components with the emphasis on the work practices of designers is perhaps the biggest differentiation between the Calder toolkit and previous efforts (most notably the recent Phidgets [7, 8] and iStuff [3] toolkits, as well as older systems such as [13]).

Related to this requirement, we also found early in our iterations that the connections between components can become a significant concern. In some cases, connectors became a size concern, and in others the routing of wires interfered with fluid placement. As a result, we felt it critical to keep connectors and wiring as small as possible, and to consider entirely wireless solutions for some components.

3. EXAMPLE DEVICES

In this section we show a number of example devices to illustrate how the Calder toolkit can be used early-on in a rapid, iterative design process.

Game controller – Figure 2, a simple game controller is prototyped by placing a wireless knob and button array in a rough foam concept model. Due to the inherent size of the knob, it is placed in a cavity which fixes its location. However, the buttons are simply pinned down into the surface and can be easily moved to the other side to experiment with both left and right-handed configurations. This allows designers to gain usage experience with the device extremely early in development. Modifications to the form and interface driven by actual interactive usage can begin immediately.

Navigation remote – Figure 3. By integrating with a tool like Macromedia Director, physical interfaces can be used to control and interact with high-fidelity interface mock-ups rather then relying on the mouse and keyboard. Here a navigation controller is being prototyped for a music kiosk. The left, right, and center buttons can be used to navigate and make selections. The buttons can be repositioned and alternative forms can be tested without having to stop the Director simulation.

Chair monitor – Figure 4. A tilt sensor has been attached to an office swivel chair and transmits data wirelessly to a nearby uplink transceiver (see Sections 4 and 5.2 below) suspended beneath the desk. The transceiver relays the state of the chair to the desktop computer and then over the network to an ambient status display. The short-range wireless connection allows monitoring the state of the chair without a wire tether, which could easily become entangled. In this case, the transmission antenna was extended with an extra wire for additional range. This example shows the diversity of potential experimental applications that could benefit from having the Calder toolkit beyond product design.

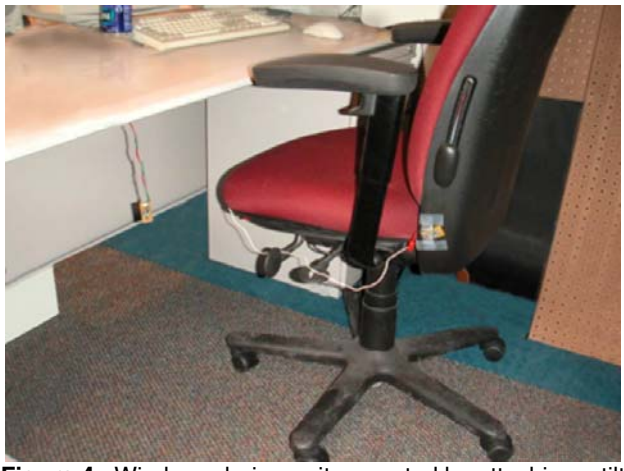

Figure 4. Wireless chair monitor created by attaching a tilt sensor to the back of an office chair. The uplink transceiver is suspended beneath the desk.

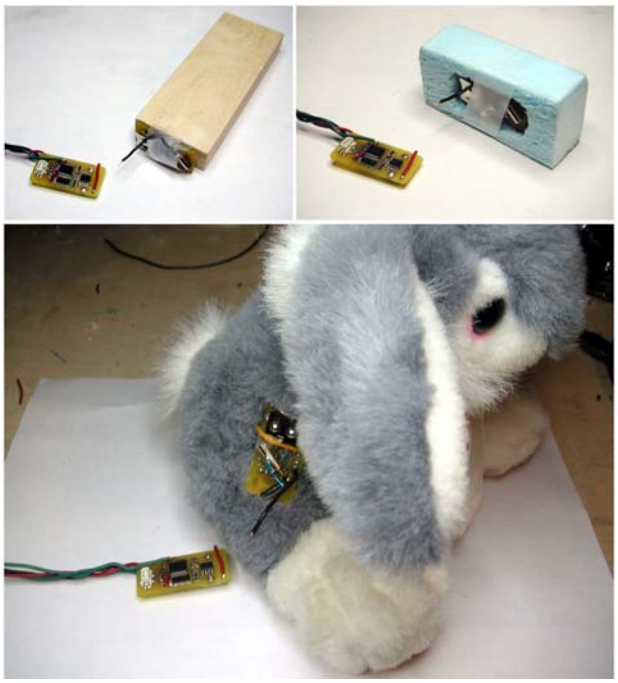

Figure 5. Quick variations on a tool stone form factor. Wooden block, foam prototype, and stuffed bunny.

Toolstone/toolbunny – Figure 5. A partial recreation of the two-handed toolstone interaction device [17] can be achieved by attaching a wireless tilt sensor onto a block of wood or foam to monitor block orientation to drive application state or modality. This allows a functional prototype to be created well before the form has been finalized. By supporting very rapid experimentation, a wide variety of forms can be tried without much investment or cause, such as using a stuffed bunny. While this example is admittedly silly, it illustrates the important point that the toolkit is fluid enough not to heavily constrain the creative expression of the designer's ideas.

In the following sections we provide detailed description of the toolkit.

4. TOOLKIT ARCHITECTURE

Each of these requirements derived from our interactions with designers have significant engineering implications and present several difficult trade-off decisions that must be made to make this toolkit realizable. This section describes some of these issues.

Figures 6 and 7 show wired and wireless components of the current version of the Calder toolkit. The relationship between these two sets of devices, the controlling PC, and development environment is illustrated in the architectural overview, Figure 8. This architectural configuration was the result of careful consideration of the user-driven factors described in Section 2, as well as the constraints imposed by technology.

From the technological point of view, two important questions became apparent very early in the development process: Where will power come from, and where will computation be performed? In considering several possible architectures and reviewing previous related systems, we found that the answers to these questions fell into three possible categories: distributed among each component, a local master, or a global master. In a distributed system, both the computational and energy resources would be distributed across each component in the toolkit. A local master is a centralized system where the computation or energy source may be pooled into a single location to achieve greater total capacity. And a global master, likely involving a tether to an external source such as a PC, would have the ability to provide substantial computational and energy resources. Each of these alternatives has implications on how well our user-driven goals could be achieved and represented potential tradeoffs. For example, component level computation is both possible and somewhat tempting, because component operation and communication will require a micro-controller anyway. It was very likely that unused computational resources would already be available in each component. However, this approach can introduce many complexities in specifying the interactive behavior of the prototype (in particular, making it a much more difficult distributed programming problem) and precludes the easy use of existing development environments. Similarly, while wireless components provide the greatest degree of fluidity and flexibility, allowing components to be attached and moved with ease, they also imply distributed power sources (i.e. batteries). This begins to conflict with one of our earlier design constraints of minimizing size since the operational life span of tiny batteries tends to be short. (Earlier work [1, 6] utilized an inductive coupling technique capable of powering small devices without the use of batteries, but we found significant functional limitations to this approach.)

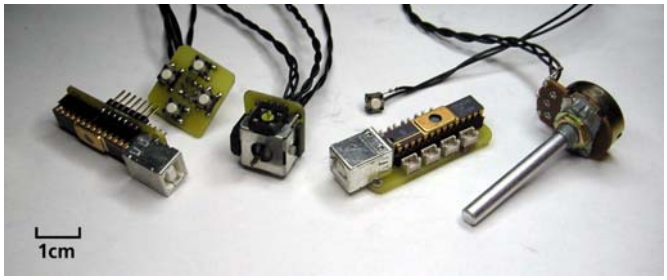

Figure 6. Wired components. (left to right) I/O Breadboard component, 4-button array, analog joystick, D/A Input Component, single button, and analog knob.

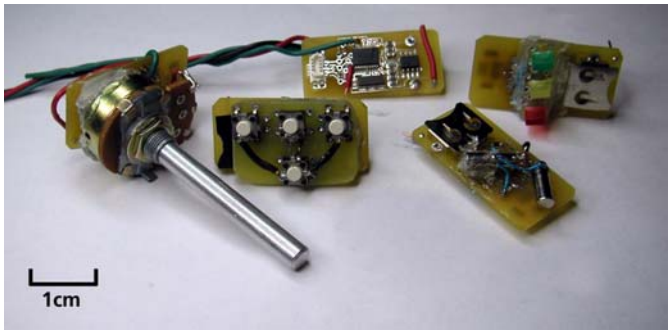

Figure 7. Wireless components: (left to right) analog knob, 4-button array, master uplink transceiver, 2-axis tilt sensor, and 3-LED array.

To address the design requirements within our technological constraints, we chose a hybrid solution for the toolkit. For computation, we chose a global model of an external PC so that we could employ the very large library of existing programming tools and environments that developers and designers are already familiar with. A PC provides storage and communication resources that may be necessary to simulate the functionality of many potential products. For power, we chose a split strategy to accommodate conflicting needs by designers. While wireless components provide the highest level of fluidity, the spatial overhead of the communication hardware causes them not to be the smallest in size. Additionally, wireless components cannot provide sufficient electrical power or bandwidth for certain components, particularly output devices such as motors or pixel displays. Wired components do not suffer as greatly in these respects; however, we felt that there was inherit value supporting the fluidity in design exploration that wireless components provide. Therefore, we created both wired and wireless components that employ a distributed and global source of power respectively.

To unify these forms, and simplify implementation, wireless components communicate with a special wired *uplink transceiver* component serving as a bridge from the wireless to the wired portion of the system, which in turn connects to the PC.

Parts of the architecture used in the Calder toolkit can be contrasted with prior systems in this area. The Phidget system [7, 8] uses a nearly identical approach to the wired components of our system (we considered several other alternatives, but in the end concluded that this option was indeed the best choice given the currently available technology). The iStuff toolkit [3] uses a similarly structured wireless solution with distributed power and global computation, but used technology and a communication infrastructure appropriate only for applications that permitted very large components. Finally, the Lego Mindstorms System [9], a well-known robotics prototyping toolkit, and the MetaCricket [13] system use primarily a local master model for both power and computation.

5. COMPONENT DETAILS

Each component in the Calder toolkit is equipped with a small microcontroller (μC) that is responsible for managing communications to the global master and providing sensing or control for its I/O device(s). Wired components are implemented with a Microchip PIC16C745 controller that includes a Universal Serial Bus (USB) interface engine. The wireless components are currently implemented with a PIC16LF819. These chips can be purchased for under $3 in quantity.

As illustrated in Figures 6-7, and described below, a basic though useful range of component types is currently provided by the toolkit. However, the type of sensors and actuators that can be interfaced is quite broad. As we perform evaluation studies and move into wider deployment we will expand the component set as needed, including the addition of actuators and additional output devices.

5.1 Wired Components

The current wired components include a general-purpose input "hub" component and a related runtime configurable I/O breadboard component. These components are capable of interfacing with a variety of specific input and output devices connected by a short cable.

The input "hub" component, Figure 9, has four connections that accept digital input from a switch or button, and four connections that accept analog input from a slider or knob. The analog and digital connectors are non-compatible

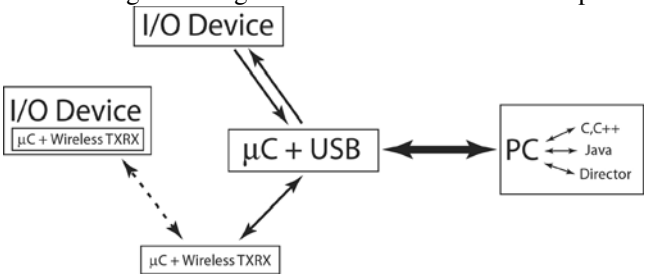

Figure 8. Architectural overview.

making it impossible to accidentally misconnect a device. Similarly, the connectors are keyed preventing polarity reversal. For testing purposes, we have created a small collection of digital and analog devices designed for use with this hub component, Figure 6. The most sophisticated of these is the 2-axis joystick with button press. This consumes 2 analog ports and 1 digital port (if the button feature is to be supported by the application). Additional hub components may be added as needed by the application to support many analog and digital input devices. The connected devices are also "hot-pluggable". This allows the designer to try reversing the axis of the joystick or replacing it with two entirely different analog devices by simply unplugging and replugging the connections even while the interface control program is still running. Thus, basic hardware reconfigurations can occur without involving the software environment.

The configurable I/O Breadboard component is designed for use as an electronic prototyping tool for interfacing to devices that are not yet in the library, Figure 10. This component pushes many of the internally configurable features of the microcontroller onto the PC for run time configuration. Pins may be switched between input and output, as well as between analog and digital as desired, using simple command on the PC. This allows the more electronics savvy developer to dynamically reconfigure the component at runtime to suit the needs of their application.

5.2 Wireless Components

The core of the wireless system used in the toolkit is a wired uplink transceiver, which provides a communication bridge between the PC and the wireless components. Included among wireless components are a small button array, small LED array, an analog knob, and a 2-axis tilt-switch sensor, Figures 7 and 11. Although the set of devices is still limited, they demonstrate that the wireless infrastructure is capable of supporting both non-trivial input and non-trivial output. A variety of other sensors and output modules can easily be created, with the primary limitation likely to be battery life rather than the wireless technology.

To meet our goal of "foam friendly" components, we designed our wireless components with a push-pin attachment mechanism (two pins per component are necessary to prevent rotation) as shown in Figure 11. These metal pins also serve as the wireless communication antennas. This satisfies the design requirement of supporting fluid design exploration by allowing the components to be repositioned without interrupting the interface control program while also satisfying a technology requirement.

6. COMMUNICATION INFRASTRUCTURE

Wired components communicate with a global master (a PC) via a USB connection. This connection provides both communication and electrical power (up to 2.5W per device), and is standard equipment on both PCs and Macintosh computers. (It is worth noting that both the Phidgets and iStuff systems made this same design choice due to these technological benefits.)

However, a wireless communication system that fit within our initial design constraints was not commercially available. Though substantial advances in wireless technology have been made recently, size of the components and the size of the battery required to implement standard wireless solutions did not meet our

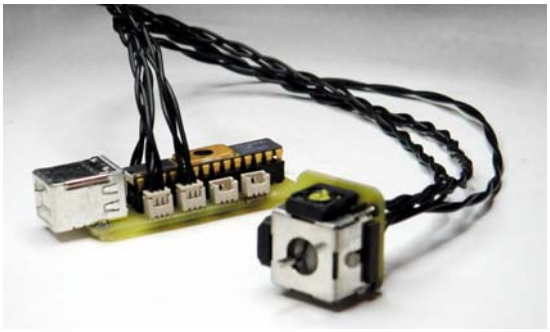

Figure 9. 2-axis joystick with button press consumes 2 analog ports and 1 digital port on the D/A Input "hub" component.

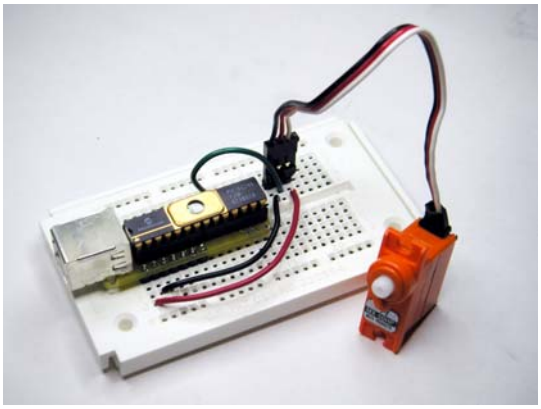

Figure 10. General-purpose runtime configurable I/O Breadboard component used for exploring new and custom devices.

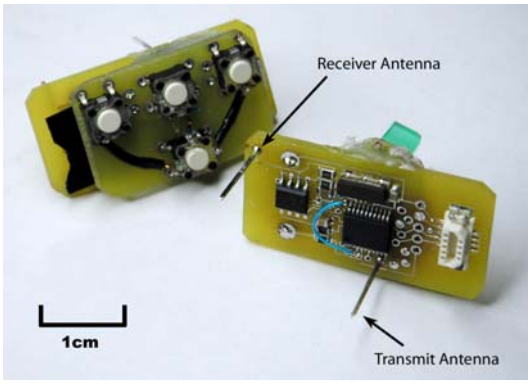

Figure 11. Close up of the wireless button and LED array. The pins on the back of the components used to attach to foam models also serve as transmit and receiver antennae

.

requirements. In order to achieve small size through a minimum of parts, we developed a simple, very low-power wireless system designed to operate only at short distances. In its typical use with foam models, we foresaw that a small transceiver component would be placed inside a cavity in the foam or on the bottom of the model (see Figures 3-5). The wireless components would then be freely positioned a few inches away on the surface of the foam.

Unlike conventional radio, the wireless technology presented here uses a capacitive coupling technique that is very low-power and does not require specific size, shape, or oriented antennas to communicate [16]. This allows each component to be operated by small coin cell batteries and short antennae.

To transmit data, all that is needed is a pulse-width modulation (PWM) capable microcontroller and a short antenna wire. Data is transmitted by controlling the presence or absence of a carrier signal on the antenna generated by the PWM module. This is often referred to as CPCA (carrier-present carrier-absent) modulation or OOK (on-off keying) [11]. Some of the limitations of this modulation scheme are mitigated by the use of a slotted transmission protocol that precludes packet collisions.

Since nothing besides an open wire is attached to the PWM output pin, a negligible amount of current is necessary for data transmission. Typically, the attached output device or input sensor will consume the majority of energy resources. Using a single 12mm lithium coin cell, we have experienced continuous transmit and receive cycles lasting over 72 hours. Since the 16LF819 draws as little as 0.2μA while in sleep mode, simple power conservation techniques can extend the operational life span of these components to several weeks without a significant change in behavior from the user's perspective.

Though transmission is possible with very short antennae, larger antennas can substantially increase the amount of capacitive coupling achieved between the transmitter and the receiver thereby improving range. By attaching a thin 6-inch wire to the transmission antenna, we have seen the maximum component range increase from 2-3 inches to nearly 2 feet. This allows the toolkit user to easily increase range simply by adding extra conductive material to the transmission antenna, such as twisting on a thin piece of wire. In many cases, this modification may have little to no impact on the usability of the prototype.

Though transmission can occur with nearly no parts external to the existing microcontroller (i.e., with only an antenna wire attached to one of its output pins), the same cannot be said about receiving data. A few additional parts are necessary to amplify and perform basic filtering. Shown in Figure 12, the receiving circuit consists of two MCP602 operational amplifiers (normally packaged in a single small integrated circuit) with a simple low-pass filter in between.

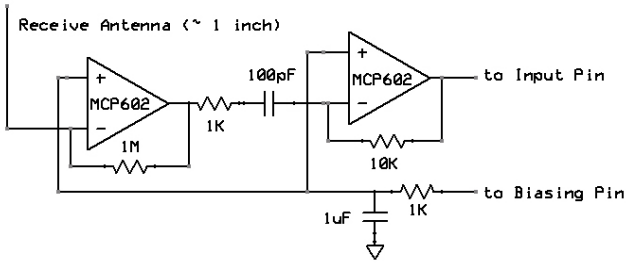

Figure 12. Amplifier circuit used to receive wireless signals. The input and biasing pins are simple digital I/O pins on a micro-controller.

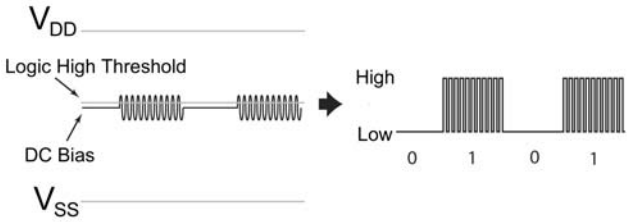

Figure 13. DC biasing the input allows small voltage variations to trigger logic high and low switches inside the micro-controller. With CPCA modulation, software can demodulate the original data.

The output of the receiver is DC biased by the microcontroller to just below logic high threshold voltage. This done with the resistor-capacitor pair connected to the biasing pin (bottom right of Figure 12). The capacitor is charged until the biasing pin reads it as logic high. The capacitor is then discharged for a controlled amount of time (in the range of 20μs) to drop the DC bias just slightly below logic high threshold. This technique allows a standard digital I/O pin to be sensitive to very small voltage variations. In the presence of the carrier signal, voltage spikes will occur at the carrier wave frequency registering as logic high, Figure 13. Since the microcontroller actively maintains the DC bias, the amount of signal necessary to trigger logic high can be dynamically adjusted in software. This gives the receiver the ability to adapt to different environments of varying noise, somewhat like automatic gain control. The input signal is then sampled and demodulated in software reconstructing the original transmitted data string. By utilizing the computational capabilities of the microcontroller, a wireless receiver can be accomplished with a very small number of external components (one eight-pin IC, four resistors, and two capacitors).

When compared to typical radio tuning technology, the frequency response of the receiver circuit is fairly wide due to the simplicity of the circuit. The result of having a wide bandwidth sensitivity is a reduction in range performance in the presence of high ambient noise. Tighter tuning around the carrier frequency is possible with additional filtering

components. This would allow signal amplification to be much more aggressive, resulting in improved range. However, our current design was chosen for its small part count and correspondingly low space requirements yet reasonable frequency selectivity at 200 KHz.

Our current implementation of this system yields a data transfer rate of 4Kbits/s (half-duplex). Improving the software demodulation or using a diode detector circuit [11] could increase the amount of available bandwidth. A data rate of 4Kbits/s is sufficient to support several concurrently operating input and output devices (but not high bandwidth components such as graphical displays and high fidelity audio).

7. PROGRAMMING INFRASTRUCTURE

The final important aspect of the Calder toolkit is the programming infrastructure. A primary goal here has been to integrate it with tools that our target users already use. Almost all interactive GUI systems are now based on a paradigm that uses objects to represent interactive components – each responsible for managing its own state, producing its own output on the screen, etc. – along with an event-based model for input. To integrate with such a system we employ the same strategy as the Phidgets toolkit – we insert *surrogate* objects inside the GUI system to represent the external Calder objects. Changes to the state of Calder components are made by changing the state of the corresponding surrogate objects (which encapsulate access to the underlying communication mechanism) and similarly changes to the Calder component state are reflected in the surrogate object. Actions on the physical component that correspond to inputs are injected into the host system as conventional events.

In this way, by providing only a small, very low level communication infrastructure, and a "glue" level of surrogate objects specialized to a particular host toolkit, we can easily transparently embed Calder components in a conventional GUI environment. Above the level of surrogate objects, Calder components behave like other components in the host environment. This allows us to reuse rather than reinvent the extensive programming and debugging environments that have been developed for GUI implementation.

At a detailed level, each wired Calder component presents itself as a standard USB Human Interface Device (HID) class that loads default hardware drivers. Each component is given a unique serial number for identification. Low-level USB HID interface details are packaged into Dynamically Linked Libraries that provide simplified access routines in C. These C routines can then be accessed from within the host environment using standard native code extension mechanisms (e.g., the Java Native Interface facility, or Macromedia Director XTRAs interface).

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented the design and implementation of Calder, a hardware toolkit to support the design of interactive products. Our design and development decision were based on preliminary work of observing existing design practices and interviews. Now that we have developed a functioning prototype toolkit, the next step in our process is to perform an evaluation of our toolkit. To this end we are in the process of developing a library of components for use in a graduate product design course. From this feedback, we will iterate on our current generation of hardware to create a system that meets the needs and expectations that designers have and facilitate the creation of better interactive products in the future.

ACKNOWLEDGMENTS

This work has been funded in part by the National Science Foundation under grants IIS-0121560 and IIS-0325351.

REFERENCES

1. Avrahami, D., Hudson, S.E., "Forming Interactivity: A Tool for Rapid Prototyping of Physical Interactive Products". Proceedings of *DIS 2002*. NY: ACM Press, 2002
2. Bailey, B., Konstan, J., "Are Informal Tools Better?: Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design", in *Proceedings of the conference on Human factors in computing systems*, pp. 313-320, April 2003
3. Ballagas, R., Ringel, M., Stone, M., Borchers, J., "iStuff: a Physical User Interface Toolkit for Ubiquitous Computing Environments", in *Proceedings of the conference on Human factors in computing systems*, pp. 537-544, April 2003.
4. Buxton, W., Lamb, M., Sherman, D., Smith, K., "Towards a Comprehensive User Interface Management System", in P*roceedings of the 10th annual conference on Computer graphics and interactive techniqu*es, pp. 35-42, August 1983.
5. Cagan, J., and Vogel, C., *Creating Breakthrough Products: Innovation from Product Planning to Program Approval*, Financial Times - Prentice Hall Publishing. 2001.
6. Dietz, P.H.; Leigh, D.L.; Yerazunis, W.S., "Wireless Liquid Level Sensing for Restaurant Applications", *IEEE Sensors,* pp. 715-20, June 2002.
7. Greenburg, S., Fitchett, C., "Phidgets: Easy Development of Physical Interfaces through Physical Widgets", Proceedings of the *ACM Symposium on User Interface Software and Technology*, pp. 209-218, November 2001.
8. Greenburg, S., Boyle, M., "Interaction in the real world: Customizable physical interfaces for interacting with conventional applications", Proceedings of the

ACM Symposium on User Interface Software and Technology, pp. 31-40, October 2002.

9. Knudsen, J., *The Unofficial Guide to LEGO Mindstorms Robots*. O'Reilly Press, 1999.
10. Landay, J., Myers, B., "Interactive sketching for the early stages of user interface design", *in Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 43-50, 1985.
11. Linx Technologies, "Modulation Techniques for low-cost RF Data Links." Application Notes: AN-00130, 1997.
12. Lin, J., Newman, M.W., Hong, J.I., Landay, J.A., "DENIM: finding a tighter fit between tools and practice for website design", Proceedings of *SIGCHI: Conference on Human Factors in Computing Systems*, ACM Press. April 2000.
13. Martin, F., Mikhak, B., Silverman, B., "MetaCricket: A designer's kit for making computational devices", *IBM Systems Journal*, v39, n3-4, p.795, 2000.
14. Myers, B.A. Hudson, S.E, Pausch, R., "Past, Present and Future of User Interface Software Tools", *ACM Transactions on Computer Human Interaction*, v7, n1, pp 3-28, March 2000.
15. Pfaff, G. (ed), *User Interface Management Systems: Proceedings of the Seeheim Workshop*, Springer-Verlag, Berlin, 1985.
16. Reed, D.G. (ed). *The ARRL Handbook For Radio Amateurs*, 79th ed. Chapter 20. The American Radio Relay League, Inc. 2001
17. Rekimoto, J., Sciammarella, E., "ToolStone: effective use of physical manipulation vocabularies of input devices". Proceedings of the *ACM Symposium on User Interface Software and Technology*, pp. 109-117, November 2000.
18. Rettig, M., "Prototyping for Tiny Fingers", *Communications of the ACM*, pp. 21-27, 37(4), 1994.
19. Ulrich, K., Eppinger, S., *Product Design and Development*. Princeton, NJ: McGraw Hill Higher Education, 1999.