Usability of State Based Boolean eBlocks

*Susan Cotterell and Frank Vahid**

Department of Computer Science and Engineering
University of California, Riverside
{susanc, vahid}@cs.ucr.edu
* Also with the Center for Embedded Computing Systems at UC Irvine

Abstract

The advent of sensor networks has enabled a vast number of sensor-based applications such as monitoring forest fires, ensuring structural integrity of buildings or bridges, monitoring homes or offices, and others. However, sensor network application developers are often scientists, office workers, doctors, or homeowners, and *not computer programmers*, often not even engineers. Thus, setting up a sensor network can be cumbersome or costly. In this paper, we discuss experiments utilizing a set of electronic blocks, called eBlocks, that developers can use to build small-scale monitor-control systems. eBlocks are intended for non-experts and do not require any programming or electronics experience. To enable the design of useful systems, eBlocks must support some logic function and some state functions. We previously developed an intuitive logic block and demonstrated its successful use by non-experts. In this paper, we introduce our basic set of state function blocks – prolong, toggle, trip, and pulse – and we demonstrate their successful use by nearly two-hundred non-expert users. Given a description of a desired system's behavior, more than 50%, and in some cases 75% or 85%, of non-experts could correctly choose, connect and configure each of those state blocks in a short 10-minute or less time period.

1 Introduction

Sensor networks are an emerging computing domain, consisting of applications like military surveillance, inventory control, home automation, and environmental monitoring. Some advanced sensor network applications consists of thousands of small sense and compute nodes communicating wireless (Akyildiz, Su, Sankarasubramanian & Cayirci, 2002)(Hill & Culler, 2002)(National Research Council, 2001)(Warneke, Last, Liebowitz & Pister, 2001). Deploying these advanced applications require programming expertise (Horton, 2004). However, there also exist a large number of *basic* applications, consisting of perhaps just dozens of nodes, with the designers of those applications being scientists, teachers, soldiers, engineers, homeowners, etc. – people who are *not* necessarily programmers. In some cases, off-the-shelf products may exist for such basic applications, but the limited configurability of those products prevents their use in most basic applications. Yet, the cost of hiring a sensor network programmer is prohibitive for many basic applications too. The result is that basic sensor networks cannot be deployed by non-experts today.

We developed eBlocks to enable non-experts to build basic sensor-based systems (Cotterell, Downey, & Vahid, 2004)(Cotterell, Vahid, Najjar, & Hsieh, 2003). eBlocks are designed specifically for non-experts and do not require any programming or electronics experience. A difference between eBlocks and widely known sensor-network nodes is that each eBlock has a *specific pre-defined function*, thereby eliminating the need for programming knowledge. Each block is self-contained and includes the corresponding hardware as well as a PIC microcontroller programmed to execute a pre-determined low-power compute and communicate protocol, eliminating the need for electronics knowledge, as shown in Figure 1. Application designers build systems by simply *connecting blocks together*, utilizing standard plugs, and configuring dials and switches found on some blocks. The connectivity of the resulting network defines the functionality of the end application, in contrast to traditional nodes forming a wireless network that an application designer must then program to form an end application.

In general, there are two types of eBlock systems: Boolean systems that operate on yes/no values (motion detected is yes or no, light detected is yes or no, etc.) and Integer systems that operate on integer values (temperature sensed in Fahrenheit, sound sensed in decibels, etc). We currently focus on Boolean systems, while future extensions will consider integer based systems.

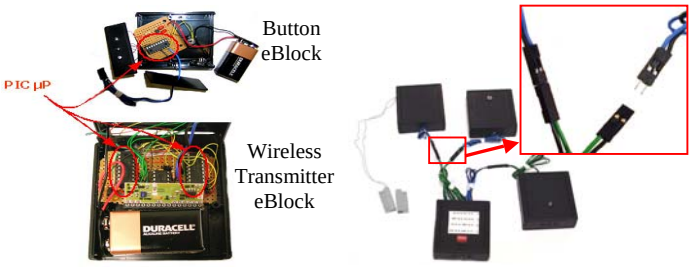

Figure 1: Each eBlock includes (a) PIC microcontroller(s) within each eBlock to simplify communication between nodes and (b) standard connecters to plug blocks together.

Within the Boolean eBlock set, we defined four categories of Boolean blocks, as shown in Figure 2 – sensor, compute, output, and communicate blocks:

- Sensor blocks - monitor the environment, including motion sensors, light sensors, buttons, contact switches, and so on. Blocks output either "yes" or "no", yes indicating the presence of the event being monitored, such as motion, light, or a button press.
- Compute blocks - perform basic logic transformations (e.g. AND, OR, NOT) or basic state functions (e.g. prolong, toggle, trip, pulse).
- Output blocks - provide stimuli, and include light-emitting diodes (LEDs), beepers, electric relays, etc.
- Communication blocks - provide wireless point-to-point communication or replicate a signal to send to multiple blocks.

The eBlock user sees an abstraction in which blocks continually transmit either yes or no. Physically, though, a block's microprocessor maintains an internal setting (yes, no, or error), and only sends yes or no packets if there is a change on the internal setting, or if the block reaches a timeout (presently 3 seconds) – thus implementing the continuous abstraction using discrete packets. The remainder of the time, the microprocessor puts the system to sleep, around 99% of the time, to save power. The blocks communicate with one another serially using packets, representing yes, no, or error. (Cotterell, Vahid, Najjar, Hsieh, 2003) provides more detail on eBlock compute and communicate protocols.

Figure 3 shows how an application designer can create custom monitor/control systems by connecting a variety of eBlocks together. Figure 3(a) demonstrates a front desk notifier, in which a customer presses the button to alert a receptionist that the customer is waiting. The beeper sounds until the receptionist presses the button again. Figure 3(b) illustrates an 8-second doorbell that activates the beeper once for 8 seconds when the button is pressed. Application developers can also create a blinking doorbell system by simply swapping a single state component, from a yes prolonger to a pulse generator, as shown in Figure 3(c). Figure 3(d) illustrates a package delivery alert system that is triggered by a delivery person on the back porch. The green/red LED inside notifies the homeowners of the presence of a package by turning green. When the homeowner retrieves the package, the homeowner resets

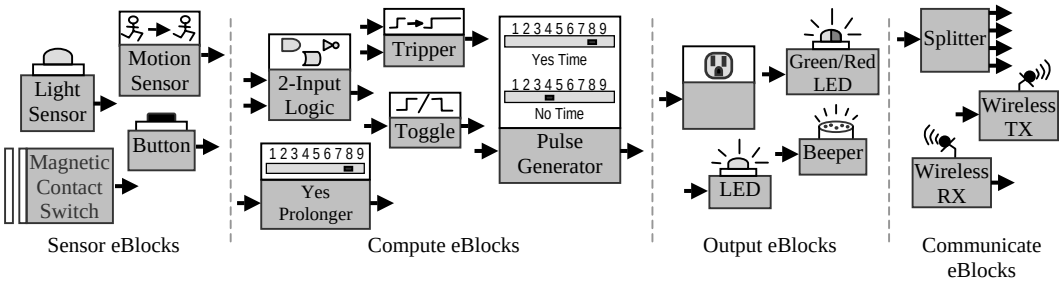

Figure 2: Sample eBlock library.

the system by simply pressing a button. Application developers can reuse most of the components in the package delivery system and create a visual doorbell as shown in Figure 3(e). When a visitor approaches the home, the visitor presses the button located at the front door, which then turns the led green. The led remains green until the

homeowner resets the visual doorbell system by pressing the reset button located within the home. A visual doorbell can be utilized to notify a hard-of-hearing person of visitors, or can be utilized by parents with sleeping children.

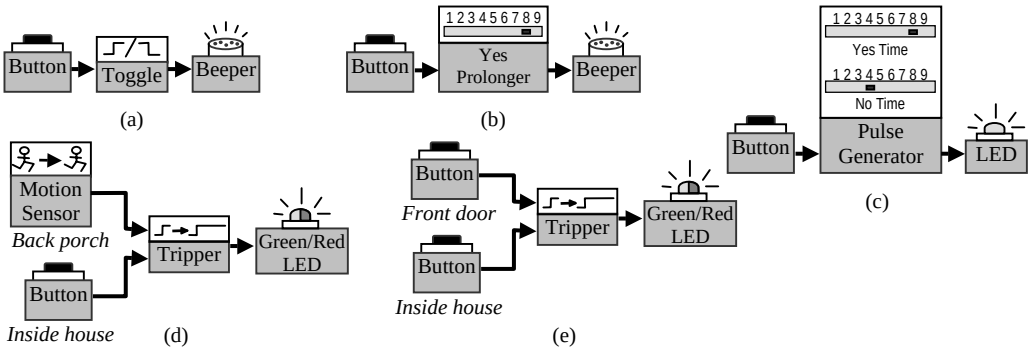

Figure 3: Various applications built with eBlocks, (a) Front Desk Notifier, (b) 8-Second Doorbell, (c) Blinking Doorbell, (d) Package Detector, and (e) Visual Doorbell.

2 Previous Work

While we attempted to design eBlocks to eliminate any need of programming, some systems still require concepts typically found in a programming environment, specifically logic and state transformations. We previously found that it is difficult for non-experts to define basic Boolean logic transformations. The difficulty of expressing Boolean equations is not limited to sensor networks. A survey for searching online public access catalogs found composing Boolean expression for searches to be difficult (Hidreth, 1989). Users were unable to compose Boolean expressions and required the aid of an expert. Web search engines also posed problems for users (Tanaka, 1999). Often times, users are familiar with AND and OR because they are found in natural language (i.e. English) but these words take on a different meaning when used in a Boolean search. The errors occur because users use the informal English meaning of AND and OR, and thus sometimes substitute the AND logical operator for the OR logical operator (Pane & Myers, 2000). Information retrieval system also requires a user to specify Boolean equations to form queries. There is work which aims to aid users in the construction of Boolean equations. Tiles, cards, or water metaphors are utilized to allow the user to view how the query is interpreted and manipulate the query accordingly (Anick, et al., 1990)(Pane & Myers, 2000)(Young & Shneiderman, 1993).

While many alternatives for specifying Boolean equations can be found, they do not translate well to a logic block interface. eBlocks are limited by power, cost, and physical dimensions, so a large graphical interface is not feasible. Further, we do not want to require a computer to configure the various blocks. We ran experiments, involving hundreds of non-expert users that evaluated various logic configuration interfaces ranging from truth tables to sentence structures. We found using color in truth tables improved success rates, as well as using a sentence-like structure. Furthermore, renaming the block from “logic block” to “combine block” enabled users to better recognize the block’s functionality, coinciding with the intuitive notion of needing to combine two or more blocks’ outputs in some manner. Students with no programming or electronics experience were more successful in configuring a sentence-like logic block than a truth table based block, with average success rates of 42% and 9% respectively. Further, only 42% of students with some programming experience were able to successfully configure a table-based logic block, while 92% were able to configure a sentence-like structure block. A more detailed description of the logic block interfaces considered, the testing methodology, and results can be found in (Cotterell & Vahid, 2004)(Cotterell & Vahid, 2005).

In testing the usability of logic block interfaces as well as generalized building of various eBlock systems, we found that many participants perform exploratory learning. eBlocks must be self-explanatory, as our experiments confirm that users often do not read or understand instructions. This is consistent with other studies - teenagers involved in computing courses preferred exploratory learning and disliked reading even the shortest manuals (Sikorski, 1998),

and visitors at museum exhibits ignored any instructions more than 20 words and instead tried to work things out for themselves if possible (Gammon, 1999).

Figure 4: Physical eBlock prototypes.

Because non-experts had such difficulty with logic configuration, we also wanted to see if similar problems arise from the selection and configuration of state based eBlocks. In programming languages, loop constructs are analogous to state based constructs. Much research has focussed on evaluating the difficulty of novices learning to program, and loop constructs have been determined to be difficult for novices (Pane & Meyers, 1996)(McIver & Conway, 1996)(Patil, Maetzel, & Neuhold, 2001). Further, studies suggest that these programming constructs need to "cognitively fit" with the user to be utilized effectively (Soloway, Bonary, & Ehrlich, 1983). Thus, in this paper we investigate the usability of state-based eBlocks by non-experts.

3 Experiments

To determine the usability of various state based eBlocks, we conducted experiments involving participants of varying skill levels. We categorized the users into three skill levels: beginner, intermediate, or advanced. A *beginner* was a university student with no programming or electronics experience and who was not in an engineering or science major – most students were in majors such as business, psychology, history, dance, etc. Our access to beginners was through a campus-wide course on computer applications (word processing, web usage, etc.). An *intermediate* student was a student who had taken anywhere between 7 weeks to 25 weeks of introductory programming, but having no electronics experience. Our access to these students was through our lower-division introductory programming courses. An *advanced* student had both programming and electronics (in particular, digital design) experience. Our access to these students was through our upper-division embedded systems courses.

We utilized three types of experiments: written, prototype, and simulator. In the *written* experiments, we provided students with a handout containing a brief introduction, a sample system built using eBlocks, and an eBlock catalog containing descriptions and the corresponding interfaces. We gave students a written quiz that asked students to build a variety of desired systems, similar to the systems in Figure 3. The quizzes described the desired system behavior, but did not directly suggest which blocks to utilize. A limitation of the written experiments was that students were not able to verify the behavior of their constructed systems, due to the lack of a simulation capability.

In the *prototype* experiments, we provided students with a box of physical eBlock prototypes, some of which are shown in Figure 4. We also included a four-step one-page written tutorial describing very basic usage. We again gave students a written quiz describing various desired systems. The physical prototypes allowed students to test the behavior of their constructed systems, which in turn allowed the students to modify the system if the students observed incorrect behavior, before the students wrote down their answers. However, fewer students were able to participate in these experiments, because we could only produce a few kits of physical eBlocks due to the time and cost of building physical block prototypes. We also found that certain limitations of our original physical prototypes, unrelated to the state features we sought to test, sometimes causes non-success. For example, students were not always familiar with the DIP (dual-inline package) switches that we incorporated in some blocks to enable block configuration. Other students forgot to power on some blocks, so that when their systems did not respond in the desired manner, they assumed their choice of blocks or their choice of interconnection was faulty. To avoid corrupting the experiments, we provided no guidance to the students beyond the one-page written tutorial. These physical prototype issues detracted from the usability testing of the state based blocks and had to be considered in parallel with the interface design.

Based on the limitations we observed with the written and prototype experiments, we developed an applet-based simulator for use in simulator experiments. In the *simulator* experiments, we directed students to a web-based applet, illustrated in Figure 5. The applet provided an interactive 2-minute tutorial containing a short introduction to

eBlocks and illustration of how to use the simulator itself (how to add, connect, and delete blocks, and provide sensor input). A pallet containing available eBlock types was located on the right of the applet's workspace. Students added blocks by clicking or dragging the various icons. The pallet consisted of three tabs – sensors, output, and compute/communicate. The sensor tab contained a button, motion sensor, and light sensor eBlock. The output tab contained a beeper and LED (light-emitting diode) eBlock. The compute/communicate tab contained a pulse generator, yes prolonger, tripper, toggle, and 2-input combine eBlock. A description of the desired system behavior was located at the top of the workspace, as shown in Figure 5. Students were able to add various eBlocks to the workspace, and simulate various inputs. They would then write the resulting systems on paper. The simulator allowed a large number of students to participate. We could also rapidly refine the block interfaces based on our observations.

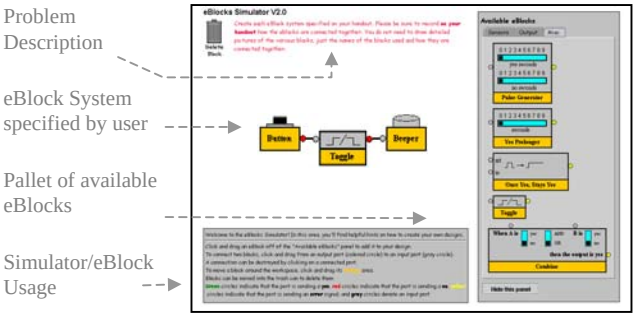

Figure 5: Graphical eBlock Simulator.

We gave students approximately 10 minutes to build desired systems. Thus, these experiments tested quick understanding of the state blocks. An alternative test would have been to allow more time, measuring the time needed by students. In all experiments, participants were first-time participants; in no case did a person participate in one experiment, and then at a later date participate in another experiment.

In all the experiments, the persons conducting the experiments gave no verbal instructions or assistance, but rather merely observed – the only instructions appeared in writing on the quizzes or in the simulator. In other experiments not reported here, the experimenter first introduced the goals of eBlocks – we found that students did better in those cases, likely due to higher motivation. However, because of the variations in student performance that could occur due to variations in the quality of the introduction, we exclude data from those experiments in this paper.

3.1 Toggle

A toggle block changes state between *yes* and *no* for each unique yes input. We provided one of the two following problem descriptions to students:

- *We want to build our own custom light switch. We want to use a button to turn a lamp plugged into the wall on and off. However, once we press a button the lamp should not turn off again until we the button is pressed a second time.*
- *Secretary Bob is frequently away from his desk. Help build Bob a system that alerts him when a customer is at the front desk. The system should keep alerting Bob until he returns to his desk and turns it off using the same button.*

Table 1: Percentage of users who correctly built systems requiring a toggle block, in less than 10 minutes, for users with varying skill levels.

Testing Method	System Description	Percentage of Success			Number of Participants
		Full Credit	Partial Credit	Total	
Written	Light Switch	24 %	0 %	**24 %**	21 - Intermediate
Written	Front Desk Notifier	20 %	0 %	**20 %**	10 - Advanced
Simulator	Front Desk Notifier	20 %	0 %	**20 %**	20 - Beginner

Table 1 provides the success rates; we gave full credit if a system implementation functioned correctly and partial credit if the correct state-based block was selected but either the connections to the block were incorrect or the configuration was incorrect. Approximately half of the students who attempted to build the light switch application did *not* select a state block. We observed many of the students attempting to build the front desk notifier used

multiple buttons, indicating that the problem description was vague (there should only be one button). We rewrote the front desk notifier problem description as follows:

- *Secretary Bob is frequently away from his desk. Help build Bob a system that alerts him when a customer is at the front desk. The customer should press the button on Bob's desk to turn the beeper on. The beeper would then keep alerting Bob until he returns to his desk and turns off the beeper using the same button as the customer.*

We also used only prototype and simulator experiments, as we observed numerous mistakes on the written quizzes that would have easily been avoided if the students could verify the behavior.

Table 2: Percentage of users who correctly built systems requiring a toggle block, in less than 10 minutes, for users with varying skill levels.

Testing Method	System Description	Percentage of Success			Number of Participants
		Full Credit	Partial Credit	Total	
Prototype	Front Desk Notifier	100 %	0 %	**100 %**	2 - Beginner 6 - Advanced
Simulator	Front Desk Notifier	85 %	0 %	**85 %**	33 - Beginner

Table 2 shows success rates for a second round of experiments (using all new students). Clarifying the problem description resulted in an 85% success rate from beginners. This success rate demonstrated that non-experts could appropriately choose and utilize a toggle block.

3.2 Yes Prolonger

Originally, the yes prolonger block was simply the prolonger block. However, we found that users incorrectly believed that the prolonger block prolonged both the yes's and no's for a user specified time. By changing the name to yes prolonger, we found that users were able to discern that only the yes input is prolonged.

A yes prolonger changes to a yes state and remains yes for a period specified by the user using a slide switch. We gave students the following problem description:

- *Homeowner Jane built a doorbell using a button and beeper. When a visitor presses the button, the system only beeps for a short time and sometimes Jane misses the visitor. Help Jane build a system such that when a visitor presses the button, the buzzer will sound for 8 seconds.*

Table 3: Percentage of users who correctly built systems requiring a yes prolonger block, in less than 10 minutes, for users with varying skill levels.

Testing Method	System Description	Percentage of Success			Number of Participants
		Full Credit	Partial Credit	Total	
Written	8-Second Doorbell	56 %	33 %	**89 %**	9 - Advanced
Simulator	8-Second Doorbell	75 %	10 %	**85 %**	20 - Beginner

Table 3 shows that 75% of beginners could correctly choose and utilize a yes prolonger.

3.3 Tripper / Once Yes, Stays Yes

A tripper block changes to a yes state when the block's data input, *in*, receives a yes packet. The tripper block remains in the yes state until the tripper block receives a yes packet on the reset input, *rst*. We gave students one of the two problem descriptions:

- *The UPS delivery person keeps leaving packages behind Homeowner Steve's house. Sometimes, Steve does not notice the packages left behind his house for days. Build Steve a system that detects if there is motion behind his house and turns an indicator on. The system should keep the indicator on until Steve shuts it off.*
- *Homeowner Lisa wants to build a visual doorbell. She often listens to loud music and cannot hear the normal doorbell. Help Lisa build a visual doorbell so that when a visitor presses the button (button 1) located outside the house it turns the light on and keeps the light on. Lisa has a second button (button 2) located inside her room, when she presses this button it should turn the light off.*

Table 4: Percentage of users who correctly built systems requiring a tripper block, in less than 10 minutes.

Testing Method	System Description	Percentage of Success			Number of Participants
		Full Credit	Partial Credit	Total	
Written	Package Delivery	30 %	40 %	**70 %**	10 - Advanced

Table 5: Percentage of users who correctly built systems requiring a once yes, stays yes block, in less than 10 minutes.

Testing Method	System Description	Percentage of Success			Number of Participants
		Full Credit	Partial Credit	Total	
Simulator	Package Delivery	45 %	5 %	**50 %**	22 - Beginner
Simulator	Visual Doorbell	55 %	36 %	**91 %**	33 - Beginner

Table 4 provides success rates of advanced users using a tripper block. Results were weak, in part due to the experiment being the written kind, but also because we observed students were not immediately translating the term "tripper" in an understanding of the block's behaviour. We renamed the block to "once yes, stays yes." Table 5 shows simulator based results with the new block, showing 45% and 55% full success rates by beginners.

3.4 Pulse Generator

A pulse generator block outputs yes and no for time periods specified by a user using two slide switches. We provided students with the following problem description:

- *Patricia built a doorbell such that when a visitor pushes the button, the light turns on. Help Patricia build a doorbell such that when the visitor pushes the button, the light blinks instead.*

Table 6 provides success rates for students with varying levels of experience. We found that 55% of beginners could correctly choose and utilize a pulse generator block.

Table 6: Percentage of users who correctly built systems requiring a pulse generator block, in less than 10 minutes, for users with varying skill levels.

Testing Method	System Description	Percentage of Success			Number of Participants
		Full Credit	Partial Credit	Total	
Written	Blinking Lamp	10 %	43 %	**53 %**	21 - Intermediate
Written	Blinking Doorbell	90 %	0 %	**90 %**	10 - Advanced
Simulator	Blinking Doorbell	55 %	9 %	**64 %**	22 - Beginner

4 Conclusion and Future Work

eBlocks enable non-experts to build basic but useful sensor-based systems. We have defined a set of basic eBlocks from which a variety of systems can be built. In this paper, we demonstrated that the four state-based blocks were all usable by non-experts. Given a problem description, non-experts could correctly choose the appropriate state-based block from a set of five state and logic blocks, and could correctly connect and configure that block, with more than a 50% success rate on average. As our experiments imposed tight time limits on the participants, we would expect even higher success rates if participants had more time. Furthermore, motivated block users would likely perform even better. Future work includes conducting usability experiments for more complex desired systems and extending our set of blocks to handle integers as well as yes/no values. We are also developing the eBlocks paradigm as a basic programming paradigm for wireless sensor network, wherein users specify desired behavior in a simulator using eBlocks, and a tool then automatically converts the behavior to software running on sensor network nodes.

5 Acknowledgements

This work is being supported by the National Science Foundation (CCR-0311026). We thank Ryan Mannion for his development of the applet-based eBlock simulator.

6 References

Akyildiz, I.F., Su, W., Sankarasubramanian, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38, 393-422.

Anick, P. G., Brennan, J. D., Flynn, R. A., Hanssen, D. R., Alvey, B., & Robbins, J. M. (1990). A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query. *Proc. ACM SIGIR Conf. On Research and Development in Information Retrieval, User Interfaces*, 135-150.

Cotterell, S., Downey, K. & Vahid, F. (2004). Applications and Experiments with eBlocks - Electronic Blocks for Basic Sensor-Based Systems. *Sensor and Ad Hoc Communications and Networks (SECON).*

Cotterell, S., & Vahid, F. (2004). A Logic Block Enabling Logic Configuration by Non-Experts in Sensor Networks. *UC Riverside Technical Report UCR-CSE-04-09.*

Cotterell, S., & Vahid, F. (2005). A Logic Block Enabling Logic Configuration by Non-Experts in Sensor Networks. *Conference on Human Factors in Computing Systems (CHI).*

Cotterell, S., Vahid, F., Najjar, W., & Hsieh, H. (2003). First Results with eBlocks: Embedded Systems Building Blocks. *CODES+ISSS Merged Conference.*

Gammon, B. (1999). Everything we currently know about making visitor-friendly mechanical interactives. *British Interactive Group, http://www.big.uk.com*.

Hidreth, C. R. (1989). Intelligent Interfaces and Retrieval methods for Subject Search in Bibliographic Retrieval Systems. *Advances in Library Information Technology*, 2.

Hill, J., & Culler, D. (2002) MICA: A Wireless Platform For Deeply Embedded Networks. *IEEE Micro*, 22(6), 12-24.

Horton, M. (2004). Commercial Wireless Sensor Networks: Status, Issues and Challenges. *Keynote Presentation, IEEE SECON*.

McIver, L., & Conway, D. (1996). Seven Deadly Sins of Introductory Programming Language Design. *Proc. Software Engineering: Education and Practice (SE:E&P).*

National Research Council. (2001). Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers (1st ed.). Washington: National Academies Press.

Pane, J. & Myers, B. (2000). Tabular and Textual Methods for Selecting Objects form a Group. *Proc. Visual Languages*, 157-164.

Pane, J. & Myers, B. (1996). Usability Issues in the Design of Novice Programming Systems. *Human-Computer Interaction Institute Technical Report CMU-HCII-96-101*.

Patil, B., Maetzel, K., & Neuhold, E. (2001). Native End-User Languages: A Design Framework. *Workshop of the Psychology of Programming Interest Group.*

Sikorski, M. (1998). Teaching Computers the Young and the Adults: Observations on Learning Style Differences. *Conference on Human Factors in Computing Systems (CHI)*.

Soloway, E., Bonar, J., & Ehrlich, K. (1983). Cognitive Strategies and Looping Constructs: An Empirical Study. *Communications of the ACM*, (26)11, 853-860.

Tanaka, J. (1999). The Perfect Search. *Newsweek* 134 (13), 71-72.

Warneke, B., Last, M., Liebowitz, B., & Pister, K. (2001). Smart Dust: Communicating with a Cubic-Millimeter. *Computer*. 44-51.

Young, D. & Shneiderman, B. (1993). A Graphical Filter/Flow Representation of Boolean Queries: A Prototype Implementation and Evaluation. *Journal of American Society for Information Science*, 44, 327-339.