A Logic Block Enabling Logic Configuration by Non-Experts in Sensor Networks

Susan Cotterell and Frank Vahid*
Department of Computer Science and Engineering
University of California, Riverside
{susanc, vahid}@cs.ucr.edu, http://www.cs.ucr.edu/{~susanc, ~vahid, eblocks}
*Also with the Center for Embedded Computer Systems at UC Irvine

ABSTRACT

Recent years have seen the evolution of networks of tiny low power computing blocks, known as sensor networks. In one class of sensor networks, a non-expert user, who has little or no experience with electronics or programming, selects, connects and/or configures one or more blocks such that the blocks compute a particular Boolean logic function of sensor values. We describe a series of experiments showing that non-expert users have much difficulty with a block based on Boolean logic truth tables, and that a logic block having a sentence-like structure with some configurable switches yields a better success rate. We also show that a particular use of color with a truth table improves results over a traditional truth table.

Categories & Subject Descriptors
H5.m. Information interfaces and presentation: Miscellaneous.

General Terms
Design; Experimentation; Human Factors

Keywords
Sensor networks; Boolean logic; embedded computing systems; truth table; eBlocks

INTRODUCTION

The continued shrinking of computer chip size and cost has led to a class of computing known as sensor networks [7,11]. A sensor network is a computing network consisting of tens to thousands (or more) of tiny compute nodes. A node may range in size from perhaps a matchbox, to something not much larger than a large piece of dust, and may cost just a few dollars to as little as just a few cents.

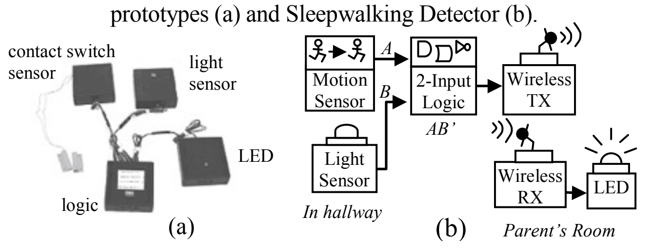
One particular evolving class of sensor networks is known as eBlocks [3,4], which we are developing at the University of California, Riverside. eBlocks are intended to enable regular people, having no electronics or programming experience, to construct basic but useful customized sensor-based systems. One such system might alert a homeowner that their garage door has been left open at night. Today, a garage-open-at-night system can be purchased off-the-shelf, but due to low sales volumes, may cost about $75. Furthermore, the off-the-shelf systems can't be customized easily. In contrast, with eBlocks, the homeowner could purchase various building blocks and as illustrated in Figure 1(a), connect the blocks, and configure the logic to detect the condition of the contact switch sensor and the light sensor each outputting false. As another example, a homeowner might want to set up a system that detects that their child is sleepwalking in the dark as illustrated in Figure 1(b). Countless similar uses of sensor blocks exist, e.g., a caregiver may want to detect certain daily activities (walking, in bed, talking on the phone) of an aging patient, an environmental scientist may wish to photograph nocturnal animals at a feeding hole, or a building manager may want to be alerted to a temperature exceeding a threshold. These systems can be built from many of same basic building blocks. Hence, makers of such blocks could sell the blocks in large volumes, thus reducing cost.

The above systems require the use of logic blocks to compute basic Boolean logic functions. For programmers and engineers, AND, OR and NOT form the basic logic blocks from which any logic function can be computed. However, we sought to create a single block that could be configured to a particular logic function, to avoid the inconvenience of users having to utilize several and varying numbers of logic blocks depending on the desired function. In this paper, we describe several designs for logic blocks, and provide data from several experiments demonstrating that a sentence-based block yields best results.

TABLE-BASED LOGIC BLOCK AND INITIAL INFORMAL EXPERIMENTS

A key design criterion for all eBlocks is that the blocks should be self-explanatory. Our experiments have confirmed to us, and are consistent with other studies [5,13], that users prefer exploratory learning and dislike reading even the shortest instructions. We built several dozen physical eBlock prototypes. Each block was about the size of a deck of cards, contained a PIC microcontroller (http://www.microchip.com), and could be connected to each other using simple two-prong plugs. We used "yes" and "no" to represent logic values, and included short descriptive phrases on each block. For example, the motion

Figure 1: Garage Open At Night System built using eBlock prototypes (a) and Sleepwalking Detector (b).



sensor block says: "Outputs 'yes' when motion is detected, 'no' otherwise." We designed a logic block with a 2-input 1-output truth table on the front of the block. We believed that non-experts would not have trouble understanding a truth table, because a truth table simply lists each input condition, and thus users would merely need to select the desired output (yes or no) for each input condition by moving a switch. We favored the truth table design because a truth table can represent all two input Boolean functions.

With these prototypes, we conducted a series of informal experiments to see what aspects of eBlocks people readily understood, and what aspects were challenging. We gave users a set of blocks, asked users to build specific systems, and observed them for 30 minutes. Participants included neighbors, kids, friends, and one high-school class. We observed most of the participants could *not* successfully use the logic block without training. Much confusion existed as to how to configure the switches and how to interpret the truth table. These early informal experiments led to us to focus on creating a better logic block.

Truth Table Experiments Using Written Quizzes

We gave a written quiz to determine if a truth table based block was intuitive to non-experts. Typical majors of participants were psychology, business, etc. We randomly distributed three different versions of the quiz among the students. All versions introduced the idea of a motion sensor, a light sensor, and a logic block, using a short description and figure. The quiz asked students to configure the logic block for various sensor conditions.

One quiz version allowed the students to answer by checking boxes in a truth table similar to the original logic block. Table 1 summarizes results in the *Truth table with variables* column. Only 36% of the students correctly filled in the table outputs for a "detect motion at night" condition, while none of the students correctly configured the table's outputs for more difficult conditions.

One might assume that the reason for the low scores was due to the use of variable names (A, B) in the quiz, and the use of the complement symbol (A'). To test this assumption, we used a second version of the quiz that used a truth table version that wrote out each input combination in English. As can be seen in Table 1 in the *Truth table with English* column, students performed better but still poorly.

We also checked whether people were comfortable writing Boolean equations. We used a third quiz version that showed an example of a Boolean equation of A and B, and that asked students to write the correct Boolean equation of input variables. As seen in the Table 1 in the *Boolean equations* column, success rates were low.

Table 1: Results of written quizzes for truth tables and Boolean equations.

Question	Truth table with variables (11 students)	Truth table with English (9 students)	Boolean equations (9 students)
Motion at night	36%	22%	11%
Motion	0%	56%	0%
Motion at night or no motion in day	0%	22%	0%
Motion or night	0%	11%	0%

Just to be sure that the notion of truth tables was the item causing difficulties, we gave the quiz version using a truth table with variables to students in a digital design course who had previous experience with truth tables. Of the six students, 100% answered the motion at night question correctly, and the average success on the remaining three questions was 90%. We found that non-expert students simply had no understanding of what the rows and columns of the truth table signified, and we concluded that a truth table is not a known concept for non-experts. We thus began our search for a single-block logic design that was more readily understandable by users.

TRUTH TABLE AND SENTENCE EXPERIMENTS USING A SIMULATOR

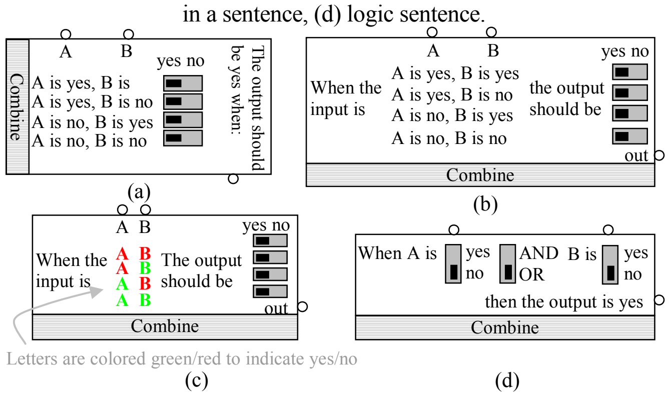
We proceeded to define three "improved" versions of table-based logic blocks, and a sentence-based logic block, shown in Figure 2. To support the next series of experiments, we developed an eBlock simulator. The simulator enabled us to experiment with larger numbers of users than possible with our limited number of physical prototypes. Furthermore, the simulator enabled users to examine the behavior of their configured systems – a feature that exists when eBlocks are used in practice, but that is absent in written quizzes.

Experiments – Four Logic Block Versions

We created a modified version of the simulator that displayed a pre-designed eBlock system with the logic block in the center, so that the user only had to configure the logic block to complete the system – the user did not have to instantiate or connect blocks. We created pre-designed systems for two different problems, with instructions briefly describing the problem and asking the user to configure the switches on the "combine" block to correctly solve the problem.

We conducted experiments in spring 2004 using the simulator. The participants were students in lab sections of our earlier-mentioned computer applications course, and our introduction to programming course for non-scientists/non-engineers. Students had 15 minutes to complete a simulator-based quiz, most finishing in less time. Table 2 shows that the logic block versions having truth tables embedded in a sentence slightly outperformed the phrased truth table. We also see that the logic sentence version seemed to outperform

Figure 2: Logic block versions: (a) phrased truth table, (b) phrased truth table embedded in a sentence, (c) colored truth table embedded in a sentence, (d) logic sentence.



the truth table versions, demonstrating 38% success for the daytime doorbell.

Experiments – Table Versus Logic Sentence

We ran simulator-based experiments again with students in a computer applications course and an introduction to programming course for scientists/engineers. We included 3 problems and tested 2 logic block versions: the colored truth table embedded in a sentence, and the logic sentence. Participants had to instantiate and connect the blocks themselves, and had 15 minutes to complete the exercises.

Table 3 shows slightly better performance by students using the logic sentence block for the slightly more difficult nighttime doorbell. Most interesting is that we see better performance by students using the logic sentence block for the motion on property problem. Although that problem seems simple, it requires a student using a truth table block to place three switches in the yes position, whereas the other two problems require only one switch in the yes position. In contrast, the sentence structure only requires moving the function switch from AND to OR.

Table 3 also shows "close to correct" responses in parentheses, defined as meaning that only one switch was in the wrong position. Notice that the logic sentence outperforms the truth table for all three problems. For the motion on property problem, note that no student failing to configure the table correctly was even close to correct.

We conducted the same experiment with "intermediate" students in a second programming course, using the daytime doorbell logic problem. Only 42% of the students successfully configured the truth table block, while 92% configured the logic sentence correctly likely because the logic sentence block looks similar to a Boolean expression in a programming language's branch or loop construct.

Experiment – Logic Sentence Block and Motivated Participants

We conducted an experiment in the summer of 2004 in which the participants were eight high-school graduates planning to attend our university as some type of engineering major, who voluntarily enrolled in a summer enrichment

Table 2: Results of simulator-based experiments for improved truth tables and logic sentence.

Question	Phrased truth table (33 students)	Phrased truth table embedded in sentence (30 students)	Colored truth table embedded in sentence 32 students)	Logic sentence (32 students)
Daytime doorbell (AB)	15%	20%	22%	38%
Garage open at night (A'B')	16%	13%	25%	19%

Table 3: Results of simulator-based experiments for logic sentence and colored truth tables. Numbers in parentheses include students whose answers were "close to correct."

Question	Colored truth table embedded in a sentence (15 students)	Logic sentence (17 students)
Daytime doorbell (AB)	47% (67%)	47% (71%)
Nighttime doorbell (AB')	33% (52%)	41% (76%)
Motion on property (A+B)	33% (33%)	65% (71%)

program. We gave them the simulator-based experiment, beginning with the daytime doorbell as the problem to solve (other problems involved state-based blocks and are beyond the scope of this paper). Students had to instantiate, connect, and configure the blocks. The only logic block available to them was the logic sentence block. Seven of the eight students, or 88%, correctly instantiated and configured the logic sentence based block.

Physical Logic Block Prototype Using a Logic Sentence

Based on the results of the above experiments, we built new physical prototypes for our logic block, using a logic sentence structure. We informally utilized these blocks with 10 additional users, and have not yet seen users encounter problems utilizing the block. Although a logic-sentence block only covers eight of the possible 16 functions of two inputs, we have found that those eight functions seem to cover most practical uses of a logic block.

OTHER EXPERIMENTS

We compared the user success rate of our single block logic approach with an approach using separate blocks for AND, OR and NOT. In the simulator-based experiments of Table 3, we included separate AND, OR, and NOT blocks as one of the versions. Users instantiated and connected the blocks. Table 4 summarizes results. We see the AND/OR/NOT blocks are competitive with the best truth table and logic sentences. However, we see that the logic sentence still has a higher success rate –we see that the intermediate students had a success rate of 69%, compared to the logic sentence of 92%. Nevertheless, AND/OR/NOT seems to be a viable option when multiple blocks for logic are feasible. We point out again though that, when using physical blocks, utilizing multiple blocks for a logic function can be inconvenient.

We also experimented with a colored logic sentence block, inspired by the colored truth table's success. The phrase "When A is yes/no" is replaced with a green or red "A," likewise for B. We introduced this colored logic sentence

Table 4: Results of using AND/OR/INV blocks. Numbers in parentheses include "close to correct" solutions.

Question	Non-expert Students (16 students)	Intermediate Students (13 students)
Daytime doorbell (AB)	63% (69%)	69% (77%)
Nighttime doorbell (AB')	50% (38%)	
Motion on property (A+B)	63% (63%)	

block into the earlier-described table verse logic sentence experiments. 12 students had received the simulator version with the colored logic sentence block. The colored logic sentence block did not perform as well as the regular logic sentence block, having only a 58% success rate compared to the 92% success rate of the regular logic sentence block.

RELATED WORK

Much work strives to simplify the design and integration of sensor based components. We briefly discuss several classes of solutions; a more detailed discussion can be found in [2]. Programmable products [7,10] are composed of a centralized programmable board or block to which a user can easily add desired sensors and actuators. Applications are specified in graphical or textual programming languages, compiled, then downloaded to a centralized board or across multiple boards. Requiring a user to learn a programming language may intimidate non-expert users and conflict with one of the main goals of eBlocks. Board products consist of electronic components connected on a specialized circuit board [8] and intended for a different audience. Block products [9] are composed of electronic components that users simply connect the desired blocks together to build complete systems. While the use of separate AND, OR, and NOT is feasible, we want to minimize the number of blocks required to build various eBlocks systems for convenience, reduced power, and cost.

The difficulty of expressing Boolean equations is not limited to sensor networks. Some research [6] shows that users confuse the vague meanings of AND and OR as used in English with those operators' precise meanings in Boolean equations. Further, users are unfamiliar with the scope of the NOT operator and often ignore parentheses [12]. There is much work in information retrieval systems, which aims to aid in the construction of Boolean equations [12]. The many alternatives for specifying Boolean equations do not translate well to the logic block interface, due to power, cost, and physical constraints making a large graphical interface infeasible, and due to not wanting to require a computer to configure blocks. Home automation is another area that requires end user configuration specifying the interactions between various appliances and devices. Direct manipulation of physical blocks is used as a programming paradigm in [1] and enables users to observe effects of the manipulation. Our approach similarly uses direct interaction and exploratory learning to enable successful configuration.

CONCLUSION

We have described an emerging class of sensor networks, which requires non-expert users to specify Boolean logic functions. We presented a variety of logic block interfaces, and experiments showing that non-expert users have difficulty with truth table based blocks. We demonstrated that utilizing color in truth tables improves success, and that a logic block having a sentence-like structure with some configurable switches yields a better success rate.

ACKNOWLEDGMENTS

This work is being supported by the National Science Foundation (CCR-0311026), and by a Department of Education GAANN fellowship. We also thank Ryan Mannion for his efforts in developing an eBlock Simulator.

REFERENCES

1. Blackwell, A., R. Hague. AutoHAN: An Architecture for Programming the Home. IEEE Symposia on Human-Centric Computing Languages and Environments, 2001.
2. Cotterell, S., F. Vahid. A Logic Block Enabling Logic Configuration by Non-Experts in Sensor Networks. UC Riverside Technical Report UCR-CSE-04-09, 2004.
3. S. Cotterell, K. Downey, F. Vahid. Applications and Experiments with eBlocks - Electronic Blocks for Basic Sensor-Based Systems. SECON 2004.
4. S. Cotterell, F. Vahid, W. Najjar, H. Hsieh. First Results with eBlocks: Embedded Systems Building Blocks. CODES+ISSS Merged Conference, October 2003.
5. Gammon, B. Everything we currently know about making visitor-friendly mechanical interactives. British Interactive Group, http://www.big.uk.com, 1999.
6. Hidreth, C. R. Intelligent Interfaces and Retrieval methods for Subject Search in Bibliographic Retrieval Systems. *Advances in Library Info. Tech., 2* (1989).
7. Hill, J., D. Culler. MICA: A Wireless Platform For Deeply Embedded Networks. *IEEE Micro 22, 6* (2002).
8. Kharma, N. and L. Caro. MagicBlocks: A Game Kit for Exploring Digital Logic. *Proc. of the 2002 American Society for Eng. Education Annual Conference* (2002).
9. Logiblocs. http://www.logiblocs.com.
10. MIT Media Laboratory. Programmable Bricks. http://llk.media.mit.edu/projects/cricket/
11. National Research Council. Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers. National Academies Press (2001).
12. Pane, J. and Myers, B. Tabular and Textual Methods for Selecting Objects form a Group. *Proc. Visual Languages* (2000), 157-164.
13. Sikorski, M. Teaching Computers the Young and the Adults: Observations on Learning Style Differences. CHI (1998), pp 42-43.