Applications and Experiments with eBlocks – Electronic Blocks for Basic Sensor-Based Systems

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

Abstract

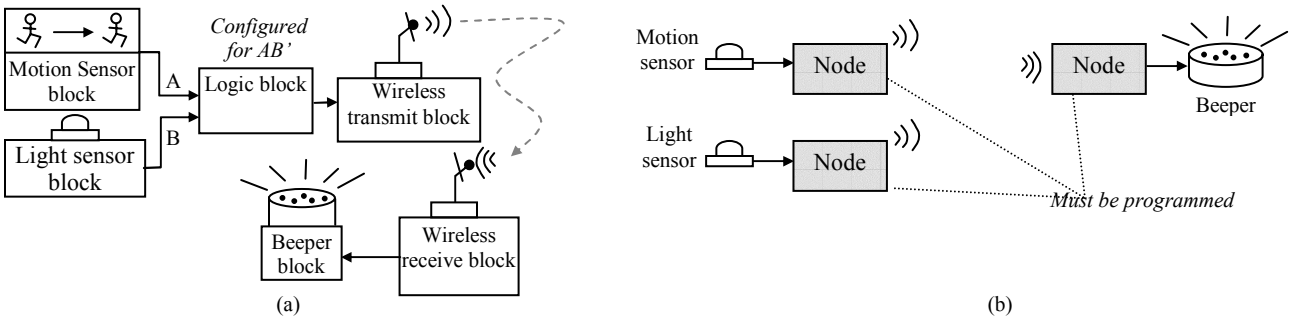
Building a sensor-based system typically requires some programming and electronics expertise. However, some applications require only basic logic transformations and/or state maintenance of sensor information. This paper describes a set of electronic blocks, called eBlocks, that enable non-experts to build basic small-scale sensor-based systems. Each block performs a particular sensing, logic/state, or output function. A user builds a system by connecting blocks together. Each block contains a hidden microprocessor executing a pre-determined low-power compute and communication protocol. A difference between eBlocks and widely known sensor-network nodes is that each eBlock has a specific easy-to-understand function, and thus does not require programming. Further, eBlocks are designed to be connected in particular configurations to create an end application, while traditional nodes form a wireless network that must be programmed to form an application. Our physical prototypes can last for several years or more on a 9-volt battery, or can receive power from wall outlets. We describe the domain of applications for which eBlocks are suitable, including being used to build complete systems or to interface with existing sensor-network compute nodes, and we summarize the eBlock compute/communication protocol. We describe experiments, involving hundreds of users of varying levels of expertise, that demonstrate how systems that otherwise would have taken weeks or more to build can be built by non-experts in just a few minutes using eBlocks.

1. Introduction

In our efforts related to embedded system design and sensor-based systems, we found that a large number of applications need a basic sensor-based system that transforms sensor data using basic logic and state functions. The transformed data is then fed directly to output devices, to a computer, or sensor-network compute nodes for further processing. Yet, we found that existing as well as proposed sensor-network technologies required some amount of programming and/or electronics knowledge to build even a basic sensor-based system. An application we encountered is a sleepwalker detector for use in a nursing home setting (but also useful in a private home), which uses motion sensors and light sensors distributed through rooms and hallways to detect motion in the dark, causing a buzzer to sound at a nursing station (or bedside in a home). This seemingly simple system today requires programming and electronics expertise to interface motion sensors, light sensors, logic integrated circuits (ICs), and microcontrollers. Contracting the design of such a system might cost well over $1,500 (excluding part costs), while off-the-shelf systems are hard to find, costly, and cannot be tuned to one's unique situation. Another application we encountered is that of photographing nocturnal animals when they feed at a particular location, to detect the presence of endangered species in a region (a project we are involved with in Riverside County, California). Again, this system needs a handful of motion sensors, a light sensor, and basic Boolean logic, to activate a camera. However, this system must be battery operated and should last for many months, further adding to the technical design demands and hence costs. The scientists on this project had to contract engineering help to design the system (costing several thousand dollars).

We have encountered several dozen similar applications and can easily think of hundreds of potential applications. Proposed sensor-network nodes, such as Motes [5], provide a framework for solutions, but still require programming. As illustrated in Figure 1, the key difference between eBlocks and sensor-network nodes is that each eBlock has a fixed and particular functionality, whereas sensor-network nodes are typically general-purpose compute/communicate nodes used to build a wireless infrastructure. In other words, connecting eBlocks results in a complete working application, while instantiating sensor-network nodes creates an infrastructure. In fact, the wireless aspect of eBlocks is optional – in many cases eBlocks will be hardwired together. We point out that eBlocks are *not* a replacement for sensor-network nodes. For some basic sensor-based applications, eBlocks can be used exclusively. However, eBlocks can also be used in conjunction with sensor-network nodes as a means to combine several sensor inputs or add basic state to sensor inputs before sensor data enters a sensor network, as illustrated in Figure 2.

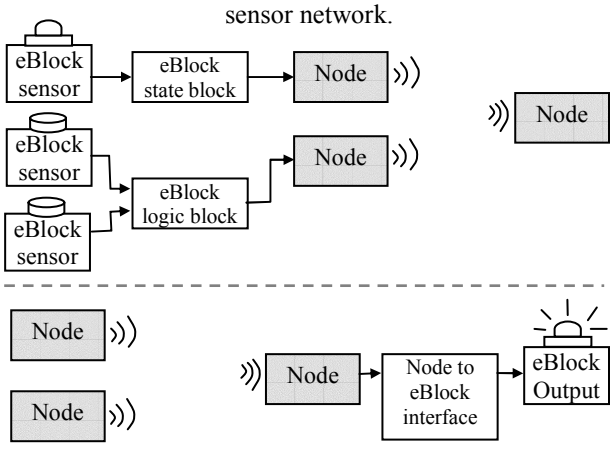
Figure 1: Connecting eBlocks: (a) forms an end-application (sleepwalk detector) without any programming but perhaps slight configuration, (b) while traditional sensor-network nodes form a framework that must still be programmed.



Other solutions, such as Phidgets [14], require a connection to a personal computer (often not possible in sensor applications) and may still require some programming. Off-the-shelf solutions for particular applications, like sleepwalk detection, sometimes exist, but can be hard to find, are typically costly due to small sales volumes, and often do not match the desired application exactly and cannot be extended easily. We discuss related work more extensively in Section 6.

We therefore sought to build a set of electronic blocks, which we call eBlocks, that would enable people with little or no programming or electronics expertise to create basic low-power sensor-based systems simply by connecting together perhaps a few dozen blocks. Some blocks would be sensors, such as a motion sensor, light sensor, button, contact switch, etc. Other blocks would perform basic logic functions (e.g., AND, OR, NOT) or basic state functions (e.g., prolong, toggle, or trip). Additional blocks would provide output (e.g., turn on a light-emitting diode (LED), sound a beeper, control an electronic relay, or interface to an electronic device). Blocks should be able to run on a battery if necessary and last for years. Our basic idea involved adding an inexpensive low-power microcontroller to each block. Thus, previously "dumb" components like a button or an LED would now actually become a tiny compute node, and connecting those components would create a small computer network communicating using packets and obeying a known protocol. By carefully defining the compute and communication protocols, we were able to create physical prototypes that last for several years on a 9-volt battery. We estimate current high-volume production materials costs of $1.50 to $3.50 per block (depending on the type of block), leading to off-the-shelf prices ranging from $6 to $14. However, trends continue to reduce those costs.

Figure 2: eBlocks as front/back-ends to a more advanced sensor network.



Using the physical prototypes, as well as an applet-based web simulator, we have over the past 12 months conducted a variety of experiments with hundreds of users of varying skill levels, to determine whether such users could build basic but useful sensor-based systems using eBlocks. Our results show that several systems, which previously took several weeks to build by skilled designers, could be built in less than 10 minutes by people with little or no programming or electronics experience.

In this paper, we will discuss the domain of applications for which eBlocks may prove useful and will describe the current state of design for those applications. We will summarize the compute and communicate protocol of eBlocks. We then describe the experiments that we have conducted demonstrating the ease with which eBlocks can be used to build basic but useful sensor-based systems. We discuss items learned and planned future work.

2. Potential Applications

Numerous sectors face everyday problems easily solved by simple sensor-based systems. However, individuals in these sectors are either unable to build these systems because they have insufficient programming or electronics experience, cannot find or customize reasonably priced off-the-shelf solutions, or cannot justify the need or expense to hire an engineer to build a custom solution. We describe four unique problems found in the residential, commercial, medical, and environmental sectors. Numerous other sectors face similar problems.

In the residential sector, one example of a useful simple sensor-based system is that of a wireless doorbell. If a homeowner is visiting their neighbor down the street or working in the backyard, the homeowner may miss visitors or delivered packages because the homeowner cannot hear the doorbell. Instead, the homeowner might

find it useful to set up a system such that when a visitor presses a button, the system would wirelessly transmit a signal to a hand-held buzzer, alerting the homeowner. Because wireless communication is utilized, the homeowner is free to go anywhere within a specified radius. Other residential applications include detecting that a garage door has been left open at night, detecting if someone is approaching the house, detecting if there is mail in a mailbox, detecting if a side gate has been left open, detecting that a carpooler has arrived, etc.

A potential commercial application is a cafeteria food alert system. A cafeteria may be laid out such that there is a food service line where workers serve the food and a separate kitchen in which workers prepare the food. When a particular item on the food service line is running low, a server must somehow notify the kitchen staff. The cafeteria food alert system would provide a solution by placing a button easily accessed by a server for each item, such that the server simply presses a button when a particular item needs replenishing. A buzzer or LED would subsequently alert the kitchen staff. The kitchen staff would also have access to another button, either so they can turn the alert off after they have been notified, or after the item has been replenished.

In the medical arena, institutions such as hospitals or nursing homes must ensure the safety of their patients. Many times, this involves nurses needing to know the location of their patients. Some patients need assistance when they are out of bed, others may sleepwalk, and some patients may not be allowed out of their beds at night. Patients typically outnumber the nurses making it difficult to be aware of the location of all patients. If nurses wanted to monitor motion in hallways at night, they could place motion sensors throughout the various hallways along with a light sensor. The combination of motion sensed and darkness would send a signal to the nurses and trigger a LED or buzzer alerting the nurse to check on a particular corridor. Depending on the situation at hand, this type of system has the potential for many variations.

Environmental science field researchers have the unique problem of monitoring nocturnal endangered species. Typically, scientists study species by first trapping individual animals, photographing the animals, and attaching a tag such that the scientists can identify the animals later. However, researchers studying an endangered species may be prohibited by law from trapping the animals. An alternative solution without trapping individual animals would be to set up a feeding station including a simple sensor-based system to detect motion at night and trigger a camera to take a photograph of the animals as they feed.

Countless other applications for each sector, and for other sectors, can be thought of.

Many of the aforementioned systems are highly specialized and therefore companies are unlikely to develop off-the-shelf systems, making it hard for users to readily purchase such systems in the market. For example, in the environmental science case, the scientists would not only have to find or build a system to detect motion at night, but would also have to properly interface the system to a camera to photograph the animals. Furthermore, even if one could find the needed products, such systems are likely to be expensive due to the systems' specialized nature and low sales volume. For example, a garage open at night system can be purchased for roughly $75 [17][19], but many consumers are not willing to pay this amount, and are unsatisfied when they cannot customize the system (to check multiple garage doors, or to provide alerts in multiple rooms, for example).

3. Design Task Today

An alternative to buying off-the-shelf products is to custom build an embedded system for each problem. While this seems like a simple task, the task is actually fairly challenging. For example, we observed an industry design situation of the cafeteria food alert system described above, requiring two weeks for a young engineer with a bachelors degree in computer engineering, just to get the basic functionality working correctly. Ideally, though, the cafeteria staff should be able to set up the desired system themselves.

Many of these systems require only a couple of sensors interfaced to an output device such as a buzzer or LED. However, upon closer examination of the problem, we see that there are a multitude of issues quickly making these systems too complex for an ordinary person with no training in programming or electronics to build.

For instance, consider building a custom wireless doorbell system. The basis of the desired system requires a button and a beeper. When the button is pressed, the beeper sounds. The first question is what kind of button should be used? A person would first start by looking at a parts catalog, such as Jameco [6]. This catalog categorizes buttons as switches. The catalog contains rocker switches, lever switches, push button switches, slide switches, toggle switches, tactile switches, key switches, and so on. Furthermore, each switch has numerous variations, involving momentary type, lead spacing, thread diameter, and contact rating. Clearly, such information is intended for engineers, as non-experts cannot easily comprehend such data. A person would face similar issues when selecting the appropriate beeper output.

Furthermore, the wireless doorbell system requires a power source. We can consider three options for a power source, a power supply, a wall outlet, or batteries. It is not likely that an ordinary user would have a specialized piece of electronics equipment, thus we eliminate the power supply option. Secondly, the end goal is a wireless system, thus we do not want to limit the user only to locations with a wall outlet. Therefore, the batteries are the most reasonable choice for a power source. If the voltage level required by system components differs from

the battery, a person will need to use a voltage regulator. The person will have to figure out which regulator to buy, read the datasheet and determine if other components are required (i.e. capacitors or resistors), and properly connect the components. The person will also have to purchase wireless components and interface the components into the system. Such interfacing requires more components, datasheets, and knowledge of electronics.

Furthermore, if the person connects the button to the beeper, the beeper may sound for too short of a time. Ideally, the system would detect the button press and sound the buzzer for several seconds or sound several short pulses. The easiest solution would be to use a microprocessor to detect the button press and control the resulting beeper action. The microprocessor could also debounce the button press, saving the user the need for more components. However, the use of a microprocessor further requires that a person knows a programming language and has the proper tools to compile and download the program to the microprocessor.

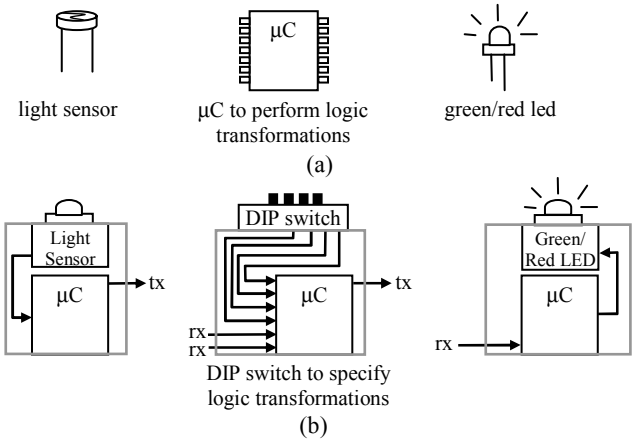
Lastly, if the person does not want to replace the battery every couple of days, he/she will need to consider the use of packets during communication, such that the microprocessor can sleep between packet transmissions, thereby reducing power consumption.

Finally, the person is ready to put all the components together, implying the use of breadboards or soldering – skills that require some expertise. If the system does not work properly the first time, a person will then require debugging skills to identify and fix problems.

It quickly becomes apparent that an ordinary person does not possess the skills necessary to build even the simplest of sensor-based systems. Even engineers who are not specifically skilled in embedded system design may find the aforementioned tasks daunting.

To illustrate the difficulty, we defined a design project similar in complexity to the sleepwalker detector, assigning the project to college juniors and seniors who already completed a class in digital design and introductory embedded systems. The students had several months of experience in programming microcontrollers, assembling basic electronic systems, implementing serial communication, and interfacing with some sensors and display devices. The students were given three-weeks. Their project involved new sensors and output devices that the students had to research and order themselves. Furthermore, students had to read electronics datasheets to figure out how to interface the various components. Of 50 students who attempted the project, only 20 were able to successfully complete the project in the three weeks. Many of the problems encountered by students related to misunderstanding data sheets, errors during interfacing, and difficulty in debugging. Furthermore, none of the students were able to design systems that could operate from batteries for more than a few days.

Figure 3: (a) "Dumb" sensors/logic/outputs and (b) sensors/logic/output with added intelligence.



Evolving sensor-network nodes will certainly help to simplify the task of building low-power sensor-based systems. However, such nodes still need to be programmed. One could consider pre-programming such nodes such that each node has a fixed function corresponding to eBlocks, and that indeed is a future direction we may consider. However, even then, sensor nodes today focus almost exclusively on wireless communication, whereas eBlocks are designed to support wired and wireless. Wired has several advantages, including making explicit to a user how blocks are connected, reducing power consumed by wireless transmission and reception, and enabling low-power communication over longer distances.

4. eBlocks: Electronic Blocks

We developed eBlocks to address the need of enabling people with little or no programming or electronics experience to build basic sensor-based systems similar to those described in the previous section. We set out to elevate the abstraction level of basic electronic components in a sensor-based system, such that each component has a standard interface, and components communicate automatically when connected.

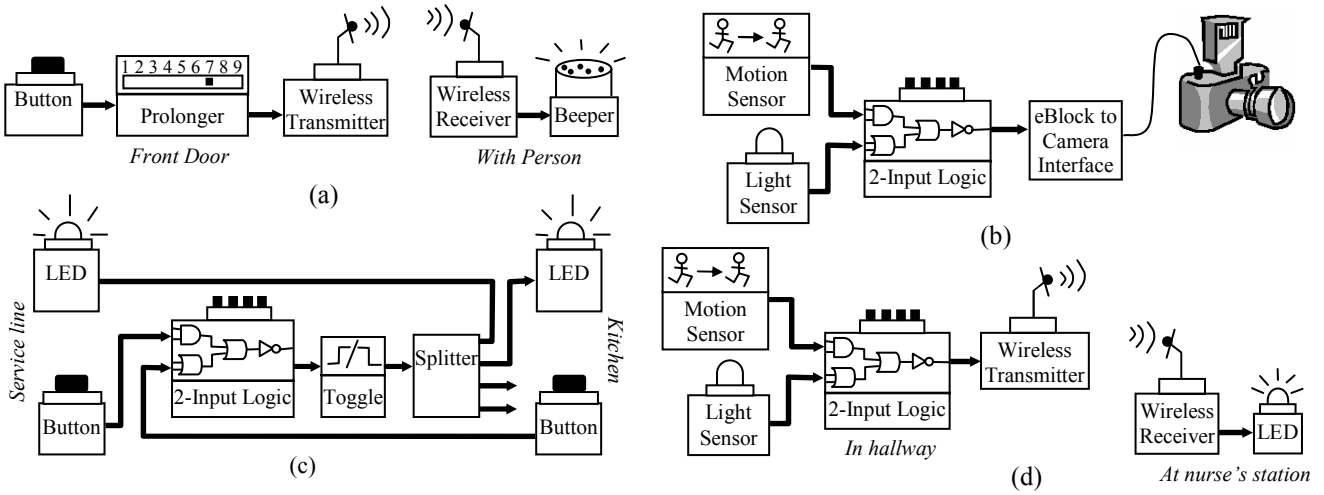
To achieve this, we took previously "dumb" components, such as buttons, light sensors, motion sensors, LEDs, buzzers, etc., and added a low-cost, low-power microprocessor (in our case, a PIC microcontroller costing about ninety cents) as illustrated in Figure 3. We preprogrammed this processor to execute the block's compute function and a communication protocol.

We defined four types of blocks: sensors, logic/state, communication, and output.

Sensor blocks output either "yes" or "no" (yes indicating the presence of the event being monitored, such as motion, light, or a button press). Types of sensor blocks include motion sensors, light sensors, buttons, contact switches, manual switches, noise sensors, etc.

Logic/state blocks take one or more yes/no inputs from other blocks and generate one or more outputs, each

Figure 4: Various applications built with eBlocks, (a) Wireless Doorbell, (b) Animal-Monitoring System, (c) Cafeteria Food Alert, (d) Sleepwalker Detector.



block implementing a basic combinational or sequential function. Logic/state blocks include a 2-input logic block whose DIP switch can be configured to any logic function of 2 inputs; a toggle block that changes state between yes and no for each unique yes input; a tripper block that changes to a yes state when the block's main input is a yes, and that stays in that state until a reset input is a yes; and a pulse generator block that outputs yes and no at rates specified by the user using two dials.

Communicate blocks include a splitter, and wireless transmit and receive blocks that can replace any wire by a wireless point-to-point link. We intentionally use wired connections as the default connection among blocks, as wired connections make the interconnection structure explicit to the user, consume orders of magnitude less power, and can enable communication over much longer distances – we performed a physical experiment in which eBlocks communicated over a stretched out standalone wire over two miles long. Nevertheless, we included wireless transmit/receive blocks for situations requiring wireless – a pair essentially replaces a wire, forming a point-to-point link.

Output blocks include an LED, a buzzer, an electric relay that can control power to any electric appliance, and a general interface that can be used to control electronic devices (like a digital camera) or to interface to a personal computer.

While blocks conceptually always transmit or receive yes's or no's, the microprocessor in fact puts the system to sleep 99% of the time, sending yes's or no's only if there is a change on the input, or if a timeout has been reached (presently 3 seconds), in order to save power. We defined other types of timeouts to ensure comfortable use when users are connecting and disconnecting eBlocks. The blocks communication serially using a 4-bit packet, representing yes, no, or error. For further details on our compute/communication protocols, see [2].

We have designed physical eBlock prototypes, which presently are the size of a deck of cards, and we estimate could be shrunk to matchbox-sized physical components.

Although eBlocks eliminate the need for programming or electronics experience, users still need to configure the logic block to specify its functionality by setting positions on a DIP switch.

Figure 4 presents how one can build the examples discussed in Section 2 using eBlocks. Users choose components and connect these components to build the desired system. Because the user designs and builds the system, the system can be as specialized as the user desires. Furthermore, the interfacing and communication aspects are already taken care of, thus, users do not need any programming or electronics expertise. Individual eBlock components can be utilized for a wide range of systems and would be inexpensive to produce in high volumes, providing inexpensive, flexible solutions.

5. Experiments

We conducted experiments to determine whether people of varying skill levels could build basic sensor-based systems effectively using eBlocks. The experiments differed in the skill levels of the users and the complexity of the systems being built.

We categorized the users, most of whom were university students, into three skill levels: *beginner*, *intermediate*, or *advanced*. A *beginner* was a 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, etc.). An *intermediate* student was a student who had taken anywhere between 7 weeks to 25 weeks of introductory programming, but with no electronics experience. Our access to these students was through our lower-division introductory programming

Figure 5: Garage Door Open At Night System built using eBlock prototypes.

courses. An *advanced* student had both programming and electronics experience. Our access to these students was through our upper-division embedded systems courses.

We categorized the systems, from simplest to most complex, as *sensor-to-output*, *sensors-with-logic*, *sensors-with-state*, and *sensors-with-logic-and-state*.

Sensor-to-output systems required the user to select the appropriate sensor and output blocks from about a dozen possible components and to connect those blocks to implement a particular application. Such systems included a doorbell system, a system that sounds a beeper when someone is in the backyard, a system that turns on a fan when there is light in a room, and a system that turns on an LED when someone is standing at a copy machine. These systems were similar in complexity to the "Wireless Doorbell" system described in Section 2.

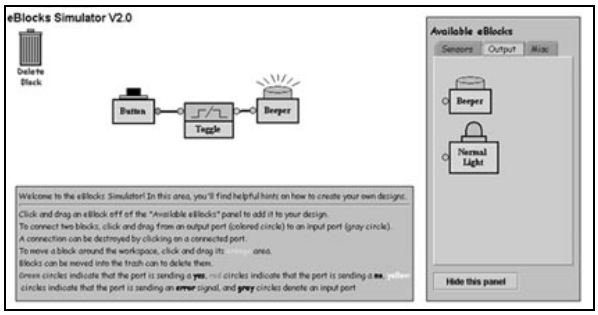
Sensors-with-logic systems required the user to use at least two sensors and to feed sensors' outputs through a logic block before connecting to an output. Examples include a daytime doorbell system, a garage door open at night system, and a motion at night detector system. These systems were similar in complexity to the "Animal Monitoring" system, and the detect and transmit portion of the "Sleepwalker Detector" system, described in Section 2.

Sensor-with-state systems required the user to connect a sensor with a state block and then an output. These systems included a blinking doorbell, an 8-second doorbell, a package delivery system which detects motion and triggers a beeper until the system is reset by user, and a front desk notifier which turns on a beeper until a user presses the same button to turn the beeper off. These systems were similar in complexity to the receiving portion of the "Sleepwalker Detector" system in Section 2.

Sensors-with-logic-and-state systems required the user to connect multiple sensors through logic and state blocks before connecting to an output block. An example is the cafeteria food alert system, requiring both a toggle block and 2-input logic.

Some of our experiments utilized our physical prototypes, in which users physically plug block inputs into block outputs. Students had 15 minutes to follow a small written tutorial describing how to build eBlock systems before building their own systems. Figure 5 illustrates a typical system, specifically the Garage Door Open At Night System, built by students. Other experiments utilized an applet-based web graphical simulator, which allowed users to construct eBlock systems and view the outcome of the various systems given specified inputs. Students were given a short step-by-step tutorial illustrating the basic idea of eBlocks, how sensors interacted with one another, how to select blocks from a library, and how to draw wires to connect the various blocks within the simulator. Students then used the simulator to create their own eBlock systems. Figure 6 illustrates a typical system, specifically a Front Desk Notifier, built by students utilizing the simulator. The simulator is available at http://www.cs.ucr.edu/eblocks. Some of our earlier experiments, included in the data below, used written tests only, which we quickly determined to be non-ideal. We had to use the simulator (or written) methods rather than physical prototypes, as testing large numbers of users with physical prototypes was not possible due to limited numbers of prototypes.

Figure 6: Front Desk Notifier built using the eBlock simulator.



Because of time limitations in the classes we visited, our experiments sought to measure what percentage of users could build the system in a short period of time (about 8 minutes). Another type of testing, which we have not yet done, would measure how much time users take to

Table 1: Percentage of users who correctly built sensor-to-output systems *in less than 10 minutes*, for users with varying experience levels.

	Percentage	Number of Students
Beginner	100%	4
Intermediate	40%	63
Advanced	92%	26
Overall	**56%**	**91**

Table 2: Percentage of users who correctly built sensor-with-logic systems *in less than 10 minutes*, for users with varying experience levels.

	Percentage	Number of Students
Beginner	35%	86
Intermediate	47%	113
Advanced	85%	82
Overall	**54%**	**281**

Table 3: Percentage of users who correctly built sensor-with-state systems *in less than 10 minutes*, for users with varying experience levels.

	Percentage	Number of Students
Beginner	100%	2
Intermediate	56%	101
Advanced	80%	65
Overall	**66%**	**168**

Table 4: Percentage of users who correctly built sensor-with-logic-and-state systems *in less than 10 minutes*, for users with varying experience levels.

	Percentage	Number of Students
Beginner	0%	2
Intermediate	0%	21
Advanced	28%	16
Overall	**12%**	**39**

build each system.

Table 1 gives results for our *sensor-to-output* experiments. Students had 8-10 minutes to select the appropriate blocks and connect the blocks correctly. The beginners used physical blocks, while the intermediate and advanced used written tests. We observed that the intermediate students did not read the written tests carefully, resulting in the low percentages. Nevertheless, we see that, on average, more than half the students were able to build the desired systems in less than 10 minutes.

Table 2 gives results for our *sensors-with-logic* experiments. The beginners used the simulator. About half of the intermediate user used written tests while the other half used the simulator. About half of the advanced user used written tests while the other half used physical prototypes. We see that a rather amazing 35% of beginners were able to build a sensor-based system with multiple sensors and including a logic block in less than 10 minutes. We also see that more advanced users had even higher success rates.

Table 3 gives results for our *sensor-with-state* experiments. The beginners used physical prototypes, while the intermediate and advanced users used the simulator. We see that more than half of users were able to build sensor-based systems with state in less than 10 minutes.

Table 4 gives results for our *sensosr-with-logic-and-state* experiments. Beginner and advanced users used physical prototypes; intermediate users used the simulator. We see that only a small percentage of users were able to complete this task in less than 10 minutes. This is not surprising. Just understanding the desired task takes several minutes, and then designing the system requires some thought and experimentation. We plan to redo this experiment with an easier-to-understand system and with more time for the users than just 10 minutes.

Furthermore, we have conducted experiments involving kids, high-school students, and adults, but the number of people involved is not yet statistically significant. Nevertheless, these interactions have helped to find shortcomings of the blocks and helped us to redefine the blocks. One redesign resulting from these experiments was the use of "yes" and "no" rather than "1" and "0" or "true" and "false," since we found the latter to be unnatural. People had a hard time converting the idea of "motion" and "no motion" to "1" and "0" or to "true" and "false," whereas "yes" there is motion and "no" there is not motion seemed more readily accepted. Further experiments resulted in our adding small LEDs directly at every block output – green meaning yes, red meaning no. This redesign was made because we found that non-engineers had a difficult time understanding the abstraction of yes/no being sent all the time; they seemed to think more in terms of just one yes being sent when motion started, rather than continuing to be sent as motion continued to be detected. The lights make explicit what is being sent. We blink those lights to reduce power consumption versus keeping them on. Yet another revision was that of eliminating a sheet describing the behavior of each block, as we found non-engineers almost never used this information – instead, we place short descriptions of each block on the block itself. We also found that most non-engineers and more than half of the engineers tested avoided reading through material more than one page, and thus we redeveloped our tutorial information to a four-step introduction on one page.

6. Previous Work

Much previous work has been done in examining new ways to implement old systems using new technology. We categorize this work into four major categories - pre-manufactured products, programmable products, board products, and block products. In this section, we will examine the work pertaining to each of these categories.

6.1 Pre-manufactured Products

Pre-manufactured products are designed for a specialized task. The benefit of pre-manufactured products is that these products are ready to use off the shelf. The drawback, however, is that each pre-manufactured system has been designed to solve only one problem and is not easily customizable to other situations. Smarthome [19] is one such company that provides over 5,000 pre-manufactured home automation products such as X10 switches and wall receptacles, motorized drape controllers, mail alert sensors, etc. Each product serves a single purpose and cannot easily be customized or re-used for other applications. For example, if a consumer purchases a wireless doorbell, they must use the included button and further cannot customize the doorbell to sound when motion is detected or only during the day.

6.2 Programmable Products

Programmable products are intended to be easily programmed by the user so that customized systems can be constructed. The user determines how the system

should be linked together and what the desired output should be. Unlike pre-manufactured products, programmable products are applicable to many different situations and must ensure that the user is able to successfully program the product, where the product's intended audience dictates the ease of programming required.

Programmable products aimed at education and toys must be easy to understand and program. Users cannot be required to read extensive datasheets to understand how a particular product works or be expected to take extensive programming courses. MIT Crickets, which evolved from the MIT Programmable Bricks project [12][13], are tiny computers powered by 9-volt batteries that receive information from two sensors and control two motors. Users program Crickets to perform a variety of functions. Users program Crickets using the Logo language [15] – a simple, graphical, highly intuitive language. Crickets provide an introduction to programming and electronics to kids and are designed for science education. Crickets also provided the foundation for the Lego Mindstorm product [21][8], which consists of numerous sensor and actuator Lego blocks used in conjunction with standard Lego blocks and connect to a central microprocessor block. Again, users program the processor using a simple graphical language. Lego Mindstorms are intended for building robotic toys.

Programmable products are often aimed at industrial applications. One such example is Phidgets [14], sensors and actuators that connect to a central board communicating with a PC. The PC is used to monitor and control the corresponding modules over a USB connection. Programming Phidgets using Visual Basic, users can quickly prototype systems. Teleo [20] is another example of a system in which a user selects sensors and actuators and connects the components to a central module. The central module is connected to a PC and can be programmed utilizing a variety of languages. However, unlike Phidgets, Teleo incorporates memory within the central module and can be disconnected from the computer.

Mica Motes [5] are miniature wireless sensing devices incorporating sensing, communication, and I/O capabilities and are intended to last years in the field utilizing only a pair of AA batteries. Each Mica node consists of processor/radio circuits that are sandwiched together with sensor circuits. A system designer would customize the Mica node to their particular application by selecting which sensors are incorporated. A collection of Mica nodes are capable of self-configuring a multi-hop network, utilizing RF communication, and support dynamic reprogramming within the network. The nodes also contain the TinyOS operating system and allow designers to customize communication protocols. The newest generation of these wireless platforms is Smart Dust [18], which are on the millimeter scale in size. These devices share many of the characteristics of the Mica nodes but utilize optical communication and have more restrictive power utilization limits. To use either the Mica Motes or Smart Dust, users must choose which sensors to utilize, program each node, and decide what communication protocols best fit the desired system. These devices are intended for people with programming and electronics experience.

6.3 Board Products

Board products consist of electronic components that must be connected on top of a specialized circuit board typically intended to provide power to the individual components. Logidules [10] were designed to help university level students studying electronics to build hardware systems. Using Logidules students snap together boxes that represent a range of components from logic gates to microprocessors. The design of Logidules eliminates the need for users to connect power to each component and users need only worry about wiring between the devices. Magic Blocks [7] were designed to teach pre-university students basic logic theory before the students begin university level computer science classes. Magic Blocks users are guided with an instruction manual that explores various logic ideas by having students build various projects. Logidules and Magic Blocks are aimed more at an educational setting. Using Logidules and Magic Blocks in real life would be challenging if not impossible due the physical setup of these systems as well as the low level components. The various gate level blocks would be confusing to users who have no computer science background.

6.4 Block Products

Block products are composed of electronic components that can be connected together to build the desired system and do not require a central module or specialized circuit board to implement the systems. Users simply need to connect the desired blocks together to build complete systems. Logiblocs [9] are small plastic blocks that users snap together to build various systems and consist of light sensors, buttons, AND, OR, NOT, speakers, bleeps, LEDs, etc. Logiblocs are intended for education and toys. Electronic Blocks [3] are blocks that consist of processors incorporated inside of LEGO Duplo Prima blocks. Users simply stack the correct combination of blocks to produce the desired output. Electronic Blocks are aimed at students between the ages of 3 to 8 years old and are limited in use for older students due to the simplicity of the blocks. Currently, most block products are aimed at younger individuals and therefore the possible systems that one can build are simplistic with no ability to create more advanced systems. RoboBrix [16] are components that users plug together to build functioning robots quickly. Each part contains a PIC microprocessor incorporating intelligence to allow components to be connected together as necessary. RoboBrix are intended

to aid in building robotic systems and are not intended for monitor/control embedded system applications.

7. Conclusions and Future Work

We have developed a set of electronic blocks, which we presently call eBlocks, that enable users with little or no programming or electronics experience to build basic but useful small-scale, low-power, sensor-based systems. Potential uses include stand-alone applications as well as front-end sensor systems to larger sensor-network systems. We asked students of varying expertise levels to build a variety of systems involving just sensors and outputs, sensors with logic and outputs, sensors with state and outputs, and sensors with logic and state and outputs. We saw that more than half of all users were able to build the first three types of systems, in less than 10 minutes. These 10 minutes can be compared to days or weeks previously required by advanced students not having access to eBlocks.

Extensive future work remains. We are presently performing studies to compare different forms of logic blocks and state-based blocks to determine relative ease of use and understanding by users; we are working with a colleague in human-computer interfacing on this topic. We are developing methods for blocks to share power with one another, so that a block with a dead battery can still operate, and so that a single block with a strong power source (wall, solar, etc.) can power all connected blocks and obviate the need for batteries. We are developing PC-based tools for more advanced users to specify and automatically synthesize and optimize eBlock-based systems, which can enable design of systems with hundreds of blocks. We are also developing a general programmable eBlock with multiple inputs and outputs, and environments for basic to advanced users to program such a block. We are interfacing eBlocks with existing sensor-based compute nodes (e.g., Motes) to enable combinations of sensors to connect with each node. We are also developing a set of eBlocks that can also operate on integers, rather than just Boolean yes/no values. We have also developed a CAD framework for designing eBlocks themselves, and exploring the implications of various compute and communication protocols.

8. Acknowledgements

This work was supported by the National Science Foundation (CCR-0311026) and a Department of Education GAANN fellowship. We would also like to thank Ryan Mannion for his contributions in developing a simulator.

9. References

[1] Beyond Black Boxes http://llk.media.mit.edu/projects/bbb/, 2004.

[2] Cotterell, S., F. Vahid, W. Najjar, H. Hsieh. First Results with eBlocks: Embedded Systems Building Blocks. CODES+ISSS, 2003.

[3] Electronic Blocks, http://www.itee.uq.edu.au/~peta/Electronic%20Blocks.htm.

[4] Home Depot, Inc., http://www.homedepot.com.

[5] Horton, Mike. et. al. MICA: The Commercialization of Microsensor Notes. Sensors Online, April 2002.

[6] Jameco Electronics, http://www.jameco.com/.

[7] Kharma, N. and L. Caro. Magic Blocks: A Game Kit for Exploring Digital Logic. American Society for Engineering Education Annual Conference, 2002.

[8] LegoMindstorms, http://mindstorms.lego.com.

[9] Logiblocks, http://www.logiblocs.com/.

[10] Logidules, http://diwww.epfl.ch/lami/teach/logidules.html, 2004.

[11] Lowe's, http://www.lowes.com.

[12] Martin, F., et. al. The MIT Programmable Brick, http://llk.media.mit.edu/projects/cricket/.

[13] Martin, F., et. al. Crickets: Tiny Computers for Big Ideas. http://lcs.www.media.mit.edu/people/fredm/projects/cricket/.

[14] Phidgets, http://www.phidgets.com/.

[15] Resnick, M., S. Ocko, and S. Papert, LEGO, Logo, and Design, Children's Environments Quarterly 5, No. 4, pg. 14-18, 1988.

[16] RoboBRiX, http://www.robobrix.com.

[17] SeniorShops, http://www.seniorshops.com.

[18] Smart Dust, http://robotics.eecs.berkeley.edu/~pister/SmartDust/, 2004.

[19] Smarthome Inc., http://www.smarthome.com.

[20] Teleo, http://www.makingthings.com/.

[21] Wallich, P. Mindstorms Not Just a Kid's toy. IEEE Spectrum, September 2001.