# Lessons Learned in Creating Real-World Interfaces

David C. Wrighton, Dillon T. Bussert, D. Scott McCrickard

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg VA 24061-0106 USA
dwrighto@vt.edu, dbussert@vt.edu, mccricks@vt.edu

**Abstract.** Informational displays have been developed that use lighting, air flow, and physical objects external to the computer screen, but typically lacking are simple and straightforward steps for creating these displays. This paper describes our experiences in creating a real-world interface (RWI) using X10 devices and common household appliances, and outlines our framework for creating RWIs quickly, easily, and inexpensively.

Presentation preferences in order: POSTER/DEMO, 5-MINUTE PRESENTATION

## 1 Introduction

The goal of this work is to develop a framework and programming interface for creating interfaces using real-world objects. We call the resulting informational displays real-world interfaces, or RWIs. Typical user interfaces use buttons, scrollbars, sliders, and similar on-screen visual displays to convey and interact with information, whereas RWIs will augment or replace these displays with changes in the surrounding environment that will convey information in a less direct but noticeable manner. For example, changes in lighting could correspond to the upcoming forecast for temperature, changes in air flow could reflect fluctuations in the stock market, and changes in ambient music could signal an upcoming meeting. We plan to provide programmers with the ability to use real-world devices in much the same way that they would use a standard user interface toolkit.

As a first step, we have constructed a RWI that displays weather information using changes in lighting and air flow. The RWI was created using X10 devices, which provide the means to send signals from computers to household devices like lamps and fans. As X10 devices are typically not used in this way, we found several issues at the programming interface level that makes it difficult to create robust, reliable RWIs. We describe these issues and outline methods for dealing with them.

## 2 Related work

This work was inspired by several projects in the area of ubiquitous computing. Mark Weiser from Xerox PARC developed the notion of "calm technology" that seeks to encalm and inform simultaneously [5]. The work is illustrated by artist Natalie Jeremijenko in her "Dangling String", an attractive wire hanging from the ceiling in a hallway that reacts to signals from an Ethernet connection. A quiet network results in only occasional twitches, while a busy network causes the wire to whirl around noisily. Hiroshi Ishii and his Tangible Media Group at MIT have developed the concept of tangible user interfaces that use physical objects as tangible embodiments of digital information [4, 2, 3]. For example, a light pattern on the ceiling reflects the activity of the lab hamster and traffic noises indicate the level of network traffic. Scott Hudson at CMU designed an information percolator that uses bubbles passing through transparent tubes to display images and patterns, thus communicating information in a pleasing yet non-intrusive manner [1].

While these and similar projects provide interesting theories and examples of interfaces outside the desktop, few steps have been taken to enable programmers to build their own devices. While previous work has argued that lighted interfaces in the environment may be too distracting [1] or that certain elements like air flow may be too difficult to notice or too difficult to control [3], one can certainly thing of scenarios when people *want* an interface to distract them (when the interface is used as an alarm) and *would* notice air flow (when it was placed near a door or narrow hallway. We hope to empower designers with the techniques necessary to create real-world interfaces quickly, easily, and inexpensively.

## 3 Background

In our work, we are leveraging the X10 protocol and hardware. The X10 protocol defines an overlaying method for sending signals over power lines between X10 hardware devices. Using this protocol, up to 256 different channels could be used with a single in-house power grid. X10 hardware devices sends signals from transmitters to receivers. Originally, transmitters were wall-mounted switches or remote controls, though devices have emerged allowing users to control and check the status of household receivers from a computer via a device that connects to the serial port. The simplest receivers plug into any 110-volt wall outlet, toggling power to appliances plugged into them. More complex receivers include dimming and querying capabilities.

The X10 protocol has been in use for over 20 years, primarily for security systems (programming lights to turn on when away) and convenience (remote control access to lights and other devices, cameras, and motion detectors). However, we see significant possibilities in the areas in the communication and interaction with information. The lighting, sounds, and visual effects generated by typical household appliances have the ability to inform users, and the dimming capabilities with newer X10 devices provide a richer, more subtle set of cues.

We created our initial RWIs in Perl. We chose Perl because it is widely used, platform independent, and has available the interface and string manipulation capabilities that are necessary for parsing the information we plan to display using RWIs.

## 4 Approach

As a first step, we wanted to gain experience in building interfaces that use X10 technology. Our goal was to understand the unique difficulties that arise in treating an X10 receiver like a typical widget in a user interface. We acquired an X10 PowerLinc device capable of sending and receiving signals from a computer, a number of X10 two-way modules that can receive signals and respond to state queries, and several household lamps and fans. Figure 1 shows the X10 components we used in constructing our RWI.


**Fig. 1.** X10 components used in constructing a RWI. The black box is the Powerlinc interface that allows a computer to send signals via the serial port through the power lines. The white boxes are X10 receivers into which appliances can be plugged. Each receiver should have a different code, with up to 256 different codes available.

We created a RWI that uses changes in lighting and air flow to preview the weather forecast. Weather data are extracted from a popular weather Web site (www.weather.com), parsed to identify the predicted temperature and wind in the upcoming hours relative to the current measured values, and used to control the brightness of a lamp and the degree of air flow from fans all connected to X10 devices. The RWI can continually preview the weather in a set number of hours (say, three hours from now), or it can cycle through the predicted weather for a given number of hours (say, for the next 12 hours). The RWI is located in a lab near a door, with the expectation that as people leave the lab, they will have a sense of the weather outside and will be reminded to bring a coat or hat, if necessary. In addition, those in the lab can look at, listen to, or feel the effects of the RWI to get a sense of the weather outside. Figure 2 shows a user sitting at a computer next to our RWI.

**Fig. 2.** A user at a computer with a RWI at the right. The RWI consists of a lamp and a fan.

We encountered numerous problems in designing our initial RWI, mainly because the existing X10 command set is not designed to be utilized in the same way as a typical user interface toolkit. Information can only be communicated very slowly, at a rate of approximately 4 bytes per second. When information was transmitted more quickly, X10 devices reacted unpredictably or erroneously. It is not surprising that the developers of the X10 protocol did not consider these issues in designing the toolkit as they did not expect people to use the commands in the way that we do. However, to effectively create interfaces using X10 devices, it is necessary to mask these problems. As such, we plan to create a module of functions for Perl that will provide an effective programming layer for using electrical appliances in the development of interfaces.

Below are some key concepts that we are incorporating into our module.

- **Provide a non-blocking programming interface for all commands.** Given the slow transmission time in sending commands, it is unreasonable to require programs to halt while waiting for completion and verification of the commands. As such, systems that interface with our library will not block upon calls to our library. This becomes far more critical as the systems we interface with do things other than just create their output through our library.

– **Provide built-in status/error checking, detection, and recovery.** We plan to check for the presence of a signal on the interface before it begins its transfer. There are several reasons for this, but primarily, it is poorly defined how frequently one should check for the presence of a communication on the line. To avoid failure in trying to send a command too soon after it sent the last one, despite the device reporting that it is ready to send, we plan to maintain state at the controlling computer, then use regular communication between the home X10 device and the remote appliance devices to help ensure consistency.
– **Associate special meanings with remote control access to X10 devices.** X10 users can control electrical appliances in a variety of ways: through the computer, via wall switches or motion sensors, or with a remote control. Our module will allow programmers to associate different behaviors with each control. For example, dimming a RWI using a remote control may mean that the end-user wants less noticeable alerts from the RWI: less intense lighting changes, quieter audio alerts, etc. Alternatively, it may signify that the end-user is less interested in seeing updates, so the rate at which the RWI is updated may decrease with dimming. Our module will provide the programmer with the power to associate unique meanings to external end-user inputs.

Our next step will be to create a Perl module that provides a programming interface that incorporates the previously mentioned concepts. The commands will be based on on the ones used to interface Tk, a popular graphical toolkit. With compact, simple commands using a syntax familiar to any Perl programmer, a programmer can control the power levels to an electrical device or can associate the power levels with changes to variables. The end product will be a Perl module that, with the appropriate hardware, will allow programmers to create applications for real-world interfaces.

As we develop the module, we plan to construct additional RWIs. In so doing, we expect that we will learn more about the needs of programmers in developing RWIs, knowledge that we will use in designing and improving the module. Specific areas in which we plan to develop RWIs include the following:

– **Augmenting handheld computers.** Handheld computers like the Palm have enjoyed a burst of popularity in recent years, but their small screen sizes make their output abilities somewhat limited. By grabbing desktop synchronization data, we will develop RWIs that integrate with the core handheld programs: calendars, address books, to-do lists, and notepads. For example, a RWI could cause a fan to blow progressively harder as a meeting in the calendar approaches, or the brightness of a lamp could represent the percent of completed items in the to-do list.
– **Computer management.** We intend to support awareness of computer system behavior at both the user level and the administrator level. For typical users, RWIs could be used to show the the ambient temperature inside

case, temperature of speed of computer fan, CPU usage, and battery availability. For system administrators, RWIs could reflect system logins, network usage, or server hits. One potential way to show this information would be to use an aquarium bubbler, where problem levels in the system would be reflected with a loud motor and lots of bubbles, thus providing audio and visual cues of the problem.
- **Tracking Web information.** Information on the Web changes frequently, to the point that it is impossible for a user to keep track of new information. These changes can be monitored with a script and displayed using a RWI. Stock prices, traffic data, weather data, news, sports scores, and library checkouts all exemplify this type of information.

When the development of the RWI module has been completed, we plan to use it in human-computer interaction courses. The students in the classes will use the toolkit to create RWIs similar to ones in the list above or of their own invention. Upon completion of the classes, we expect to have a toolkit and set of sample applications that is robust enough for general use.

## 5 Conclusion

This paper described our experiences in creating a RWI using X10 devices and common household appliances. We discussed the issues that arise when using X10 devices in this way, and we outlined a set of concepts that should be considered to help overcome them. We described our plans for a toolkit for building RWIs and a series of tools that could be built using this toolkit.

## References


1. Jeremy M. Heiner, Scott E. Hudson, and Kenichiro Tanaka. The information percolator: Ambient information display in a decorative object. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '99)*, pages 141–148, Asheville, NC, November 1999.
2. Hiroshi Ishii, Ali Mazalek, and Jay Lee. Bottles as a minimal interface to access digital information. In *Conference Companion of the ACM Conference on Human Factors in Computing Systems (CHI '01)*, pages 187–188, Seattle, WA, April 2001.
3. Hiroshi Ishii, Sandia Ren, and Phil Frei. Pinwheels: Visualizing information flow in an architectural space. In *Conference Companion of the ACM Conference on Human Factors in Computing Systems (CHI '01)*, pages 111–112, Seattle, WA, April 2001.
4. Hiroshi Ishii and Brygg Ulmer. Tangible bits: Towards seamless interfaces between people, bits, and atoms. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '97)*, pages 234–241, Atlanta, GA, March 1997.
5. Mark Weiser and John Seely Brown. Designing calm technology. *PowerGrid Journal*, 1.01, July 1996.