iStuff: A Scalable Architecture for Lightweight, Wireless Devices for Ubicomp User Interfaces

Meredith Ringel, Joshua Tyler, Maureen Stone, Rafael Ballagas, Jan Borchers
Stanford University Department of Computer Science
{merrie, jtyler}@cs.stanford.edu, stone@stonesc.com,{ballagas, borchers}@stanford.edu

ABSTRACT
iStuff ("interactive stuff") is an architecture for a toolkit of lightweight, wireless, platform-independent, physical user interface components. We present examples of several iStuff devices we have developed, and discuss the software infrastructure that supports them and allows for easy configurability and extensibility.

Keywords
User interface, tangible interface, physical interface, tuple space, interactive workspaces, wireless devices.

INTRODUCTION
iStuff is a toolbox of wireless, platform-independent, physical user interface components designed to leverage the tuple-space-based software infrastructure of Stanford's iRoom, a technology-augmented room used as a testbed for ubiquitous computing and user interface research[4]. Application users can easily and dynamically configure iStuff physical interface components to flexibly set up sensors and actuators in a ubicomp environment without having to solder components, run wires, or write device drivers. As a prototyping toolkit, iStuff aims to facilitate research at the crossroads of ubiquitous computing and HCI.

In the past, Ishii's Tangible Bits project [3] introduced the notion of bridging the world between "bits and atoms" in user interfaces. More recently, Greenberg's Phidgets [2] provide physical widgets, designed for rapid development of physical interfaces that expand Desktop GUIs. In contrast, our approach assumes an entire interactive room as its environment. Consequently, our devices must be dynamically retargetable to different applications and platforms. Additionally, devices are lightweight because they can leverage the existing interactive room infrastructure.

iSTUFF ARCHITECTURE
Several criteria guided our design of the iStuff architecture. We wanted our devices to have completely autonomous packaging and battery-powered operation, the ability to communicate wirelessly with existing applications in the iRoom, and simple, affordable circuitry. We found that we were able to use several different hardware technologies to create iStuff devices that fulfilled these criteria; so far we have working iStuff made from custom-built RFID components, from X10 components, and from standard FM radios and transmitters. From these platforms we have built several different types of devices (see Figure 1) including buttons, sliders, LED's, buzzers, and speakers. Schematics for some of these devices can be found at http://www.stanford.edu/~borchers/istuff/. Additionally, we have integrated commercial products as iStuff devices, such as MIDI controllers for 2D input, and portable microphones with voice recognition software for speech input.

Ultimately, the specific communications medium (RFID, X10, Bluetooth, 802.11b, etc.) employed is irrelevant, because of the iRoom's underlying software infrastructure. This infrastructure is based on the Event Heap—an event-based communication mechanism that connects all of the platforms, applications, and devices in the room. To add a new physical hardware iStuff platform (such as Bluetooth), one merely needs to write a glue layer that translates hardware input into an event or, conversely, translates events to hardware output. Once the device driver is written for a platform, each individual device can be uniquely mapped and configured without modifying the driver.

Input devices (e.g., buttons, sliders) transmit device specific data (e.g., device ID, position) via their wireless protocol to a receiver which is connected to a proxy computer via a parallel, serial, or USB port. This data is then packaged into an event and placed on the Event Heap using a hardware glue layer. Applications or devices can be controlled by iStuff by adding a listener notification method for specific events, which can be done with a few lines of Java code. Output devices (e.g., LEDs, buzzers, speakers) work in a complementary manner, with software listeners converting events with device-specific data to hardware output.

Device events can be translated to different application events via a patch panel application that has user configurable, dynamic mappings. This allows the user to pick up an iStuff device, go to a configuration webpage and change the mapping of the device to control a running application on any machine in the room in less than 30 seconds.

We are currently expanding the iStuff framework by including new types of discrete and continuous input and output devices, and by experimenting with new technologies for wireless transmission such as Bluetooth.

DISCUSSION
By embodying most of the "smarts" of iStuff in a computer proxy which sends and receives IDs and handles the posting and retrieving of event tuples, we were able to make the physical devices themselves very lightweight (Figure 2). They merely need the capability to send or receive their device-specific data.

Figure 1.Various iStuff. The iDog sends a button press event when turned over. The iPen is an augmented SMARTBoard pen, where the embedded button operates as a "right click" to the Windows OS. The X10 buttons are standard X10 hardware. All other devices work through a simple RF receiver that plugs into the USB port of the Proxy PC.

Utilizing the tuple-based event model of the Event Heap has allowed us to create configurable devices which are platform-independent. This architecture provides great flexibility and rapid prototyping of input devices [1]. It has allowed us to turn several commercially available items like X10 controllers into iStuff, in addition to our own custom-built hardware. The "patch panel" software increases the flexibility of our toolkit by allowing for dynamically configuring mappings between events and devices, thus permitting exploration of the ramifications of having input devices that are not tied to a specific machine or display.

iStuff has proven its worth in our lab by allowing us to quickly create experimental interfaces. For instance, we have integrated iButtons into meeting capture software as a way to provide customized annotations, and we have made a videogame where iSliders can substitute for mouse input (refer to our poster for more details on usage scenarios).

As technology continues to move beyond the desktop and toward ubiquitous computing environments, we predict that device architectures like iStuff will become increasingly commonplace. By leveraging the infrastructure in its environment, iStuff enables rapid prototyping and configuration of new and creative user interfaces for ubicomp settings.

ACKNOWLEDGMENTS

We would like to thank Michael Champlin, Joyce Ho, Robert Brydon, and Adam Rothschild for their contributions, and the NSF for graduate student fellowship support.

REFERENCES

1. Borchers, J., Ringel, M., Tyler, J., and Fox, A. Stanford Interactive Workspaces: A Framework for Physical and Graphical User Interface Prototyping. *IEEE Wireless Communications*. Special Issue on Smart Homes, 2002 (in press).
2. Greenberg, S. and Fitchett, C. Phidgets: Easy Development of Physical Interfaces Through Physical Widgets. *Proceedings of UIST 2001*, 209-218.
3. Ishii, H. and Ullmer, B. Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. *Proceedings of CHI 1997*, 234-241.
4. Johanson, B., Fox A., and Winograd, T. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing Magazine,* 1(2), April-June 2002.

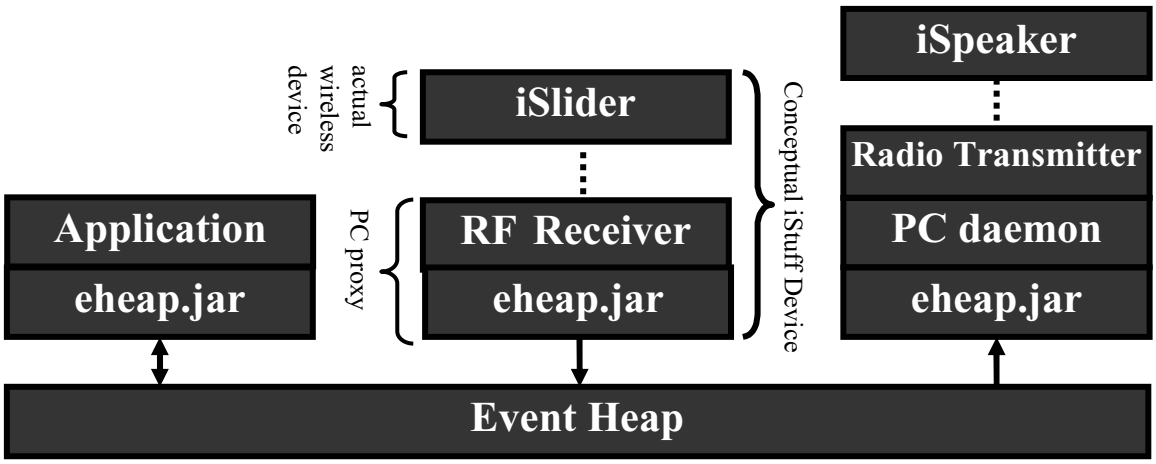

Figure 2. The iStuff architecture – much of the functionality is handled by the computer proxy, allowing the actual physical device to be quite simple and lightweight. The Event Heap mediates communication between disparate devices and applications – in the above diagram, the Application can receive events from the iSlider and send events to the iSpeaker via the Event Heap; the slider and speaker could easily be replaced by other iStuff input and output devices.