Blades & Tiles: an extensible hardware architecture for ubiquitous interaction devices

Rajesh Sankaran[1,2], Brygg Ullmer[1,3], Srikanth Jandhyala[1,3], Karun Kallakuri[1], Shining Sun[1], and Chris Laan[1]

Louisiana State University
Center for Computation and Technology[1],
Electrical Engineering[2], Computer Science[3]
Baton Rouge, Louisiana 70803, USA
{rajesh, ullmer, srikanth}@cct.lsu.edu

Abstract. The development of electronic hardware for ubiquitous computing applications has been simplified and made more accessible by the emergence of hardware toolkits. Building on this prior work and our previous experiences, we are developing a new approach for hardware design suitable for middle to late stages of implementation and deployment. We believe our "blades & tiles" approach can potentially lead to accelerated iterations of design and production; greater independence of modular subcomponents; and improved stability and debugging. We describe our motivation, design principles, implementation, and initial successes with building systems based on blades & tiles.

Keywords: Hardware design principles and toolkits; extensibility; scalability

1 Introduction

Ubiquitous computing often requires strong collaborations between hardware developers, software designers and architects, and product and interaction designers. These researchers and practitioners often speak different technical languages; some are likely to have limited hardware development experience. Developing a successful interaction device for a given ubiquitous environment often involves much design iteration in all the involved fields. For example, changes in the structural and aesthetic design of an interaction device toward improving functionally, usability, visual appeal, and environmental integration often require interface hardware and software redesign. In parallel, hardware and software design decisions and constraints can deeply influence the scope and associated interface design of a product.

Substantial time is also generally required for prototyping different interaction designs and testing them for usability and performance. Building each interaction device "from scratch" (i.e., from discrete components) often requires significant development and debugging time. Current hardware toolkits have substantially simplified this process [1-3,5]. However, we believe the state of the art is still relatively young, and alternate toolkit architectures can reshape the process of interaction device design.

While software design can be highly complex, the malleability of software provides a level of flexibility and potential for rapid iteration that is rarely enjoyed in hardware design. For software, this speed is particularly true in scripting languages accompanied by rich library support. We seek to realize analogies of this scripting power in hardware. While some level of modularity is common in many hardware design toolkits, we believe architectural decisions regarding *where* and *how* modularity is expressed can have major implications for reconfigurability, reuse, extensibility, and scalability; and for the level of coupling between hardware, mechware, firmware, and software.

2 Design rationale and principles

Our effort have had several motivations. First, we needed to make several dozen copies of 3-5 different interaction devices, to be sited at a dozen locations. These devices had demanding communication, sensing, display, and actuation requirements. We knew that a number of hardware, mechware, firmware, and software design iterations would be necessary, likely spanning a period of several years, long after initial deployment began. We sought to avoid lock-in and obsoletion whenever possible. Specifically, we wished to maximize the level of hardware and firmware reuse, and minimize costs and waste as the system evolves.

Second, we sought to prepare for future generations of hardware which might be highly specialized in form and function, perhaps involving hundreds of sensors or displays in physically or logically descriptive designs. We believe interaction devices of this kind should be possible to build, ideally by a small team with limited resources. However, for several reasons, we anticipate that systems of this scale may be difficult to realize with prior hardware toolkits.

We felt an appropriate toolkit should be composed of a series of functional modules, each providing interaction modalities that can be rapidly interchanged to build various interfaces. The toolkit would ideally have a hot swappable communications infrastructure, and be scalable and fault tolerant, with support for locating faults and maintaining operability in the presence of faulty subsystems. Both the toolkit and derivative systems should be structured such that they are understandable by hardware, software, and interaction designers alike. For these different audiences, we imagined the following expectations:

I. hardware designers: ability to add interaction capabilities (i.e., sensing, display, and actuation), without dependency on kind or number of modalities in use.

II. software designers: ability to access and support hardware capabilities through clearly modularized and abstracted libraries that represent and interact with hardware, while maintaining tolerance for varying network bandwidth, latency, and reliability.

III. interaction designers: ability to add, remove, and migrate physical interaction elements (both at the level of physical interactors and supporting electronics) with confidence that the software, firmware, and electronics will continue to function.

Following this philosophy, we have developed an implementation of *blades and tiles*.

3 Implementation

We have worked to reduce our design philosophy to practice through a system of blades and tiles, and several interaction devices built upon them. Our system's architecture is structured in several hierarchical levels. From the "bottom to top" of hardware hierarchy, these include blades, tiles, and interaction devices.

Of these, interaction devices are well known. For us, these are most often network-linked tangible interfaces, but can also include the full spectrum of ubicomp hardware technologies. At the lowest level are several classes and instances of blades. There are three major blade classes: core, function, and resource blades.

Core blades form the "nervous system" and communications backbone of blade-based devices, and are analogous to Internet routers. *Gateway blades* link the internal bus of blade-based devices with external networks (e.g., USB or Bluetooth). *Intracomm blades* hierarchically route messages and power between global and local buses.

Function blades implement specific interaction capabilities. They include *sensor blades* – e.g., a four-sensor RFID blade, or a 10-element switch blade; *display blades* – e.g., a 10-element LED blade, or a single-element LCD blade; and *actuation blade* – e.g., a two-element stepper (motor) blade, and a four-element servo blade.

Finally, *resource blades* provide system-level resources. Envisioned examples include battery, compute, and storage blades. We have built and used dozens of working prototypes of core and function blades, but not yet of resource blades.

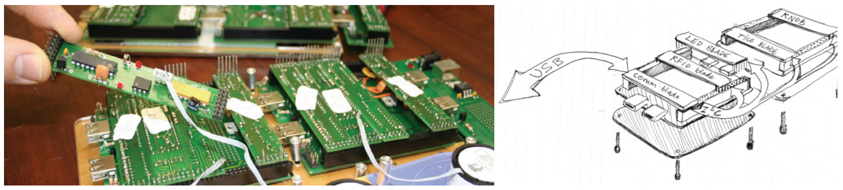

Figure 1: Blades and tiles installed within interaction devices; illustration of control flow

Partially analogous to server racks, blades are of integral size – currently 2cm wide, 10cm long, and 1cm deep. Blades may also be of integral width multiples (e.g., 4cm or 6cm). Blades mechanically fixture and electronically interlink through two interconnects. One of these is a fixed-pinout bus interconnect channeling power, identity, network, and in-circuit programming . Pinouts of the other interconnect are blade-specific. Each blade typically integrates a local microcontroller or processor.

Blades physically connect to tiles. Tile circuit boards typically have no active processors; instead, they typically host switches, LEDs, RFID interrogation coils, and other blade-mediated interactors. Some interaction devices can be implemented using a single tile. However, most devices we have implemented incorporate four or more tiles. As such, tiles provide a structural decomposition of an interaction device.

Blade & tile systems internally use an I^2C communications network. Gateway and intracomm blades execute our algorithms for blade discovery, data transport, and system control. Each tile has one intracomm blade, and one or more slave function blades. Each function blade's identity is interrogated before soft power-up, supporting hot-swapping, on-the-fly configurability, and extensibility. Our transport protocol has been modeled in part based on TCP/IP and embedded protocols like CANbus. Following this approach, we have successfully built and tested systems including up to 14 tiles and 45 intercommunicating blades. The most mature of these systems are several kinds of "core tangibles" interaction devices. These are pictured and described in [4]. These interaction devices have undergone more than a dozen iterations. Blades and tiles have been invaluable in this process, allowing us to rapidly evolve and customize our prototypes as we gain usage experience.

For example, we initially designed our hardware around one format of RFID-tagged objects. Later, we decided to double and quadruple the physical size of these objects. This required four times as many switches, LEDs, and RFID control points, along with an alternate layout. To realize this modification, less than two hours of electronics alteration and one line of firmware modification were required, all resulting in rapidly replicable, final-form circuit boards.

4 Conclusion

Building on prior hardware toolkits by others and ourselves, we have articulated a series of design principles for a new kind of extensible hardware architecture for ubicomp interaction devices. We have supported the feasibility and utility of this architecture with a system of blades & tiles, along with several interaction devices which internally rely upon them. We have tested these devices intensively in our laboratory. Blades and tiles have also been used extensively and effectively by students in two mixed undergraduate/graduate courses. Initial results have been very promising. We are working to release blades, tiles, and several embodying interaction devices open source hardware, mechware, firmware, and software, in hopes of stimulating broader adoption and a community of supporting software and devices.

5 Acknowledgements

This research is supported by NSF MRI 0521559. Special thanks to Urban Wiggins, Miriam Konkel, Frederick Konkel, Michael Carroll, and Mariel Losso.

6 References

1. Greenberg, S. and Fitchett, C. Phidgets: Easy Development of Physical Interfaces through Physical Widgets. In *Proc. of UIST'01*, pp. 209-218.

2. Hudson, S., and Mankoff, J., Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil, and masking tape. In *Proc. of UIST'06*, pp. 289-298**.**

3. Joliffe, D. Arduino Fever. In *Make*, v7, pp. 52-53, 2006.

4. Ullmer, B., Sankaran, R., et.al. Tangible Menus and Interaction Pages: tabletop tangibles for common physical/digital activities. Submitted to *Tabletop'07*, http://tangviz.cct.lsu.edu/i/tabletop07-sub.pdf.

5. Van Laerhoven, K., Villar, N., et al. Pin & Play: Bringing Power and Networking to Wall-Mounted Appliances. In *Proc. of Networked Appliances'02*, pp. 131-137.