Innovations in Ubicomp Products

Editor: Albrecht Schmidt ■ University of Stuttgart ■ albrecht@computer.org

A New Era for Ubicomp Development

Steve Hodges, Nicolas Villar, James Scott, and Albrecht Schmidt

EDITOR'S INTRO

Developers and inventers need tools to create functional prototypes to transform visions into tangible products. Recent advances have taken prototyping to a new level. In fact, in the "Hello World" sidebar, we show how to create a functional prototype of a digital camera in less than two hours. *—Albrecht Schmidt*

Over the past decade, a range of platforms have become available that can help researchers and hobbyists prototype new concepts in the ubiquitous computing domain. Such platforms, including Arduino and Microsoft .NET Gadgeteer, lower the barrier to entry for building custom electronic devices and facilitate plug-and-play prototyping. They also bring sophisticated software tools and abstractions—commonplace in desktop computer software development—to the embedded arena, letting users quickly and easily create devices with complex behaviors.

Combined with recent advances in rapid-physical-prototyping technologies, it's now possible to control the form factor of these prototypes and—for the first time—create devices with form and function approaching consumer expectations for electronics devices. These technologies might ultimately democratize embedded concept development, supporting a more diverse ecosystem of ubicomp products.

HARDWARE IS KEY

Many early ubicomp research projects leveraged a combination of novel hardware and software to demonstrate innovative ideas. Prominent examples include indoor location at Olivetti[1] and personal computing devices at Xerox PARC.

For many research groups with their roots in computer science, developing custom hardware was difficult and time consuming. Building a new device meant building it from scratch—designing the system and printed circuit board (PCB), programming the basic system software, and writing the application. This was a demanding endeavor because of the skills required across electronic circuit design, PCB layout, and system programming. However, the impact of systems such as the Active Badge and ParcTab underline the benefits of such an approach to exploring and validating radically new product concepts in ubicomp.

As we fast-forward from the early days of ubicomp to the present, researchers and product designers alike still see tremendous value in combining innovations in hardware and software. Examples range from projects such as Microsoft's SenseCam[2] and Intel's WISP (http://seattle.intel-research.net/wisp) to mass consumer products that push into the ubicomp domain from various directions—such as gaming controllers, satellite navigation devices, and smartphones. As the ubicomp field continues to develop, one thing seems clear: the underlying hardware isn't a given.[3]

COMPONENT-BASED ARCHITECTURES

For several decades now, software developers have been exploiting component-based architectures to speed the development of increasingly complex software systems. To reduce the effort of hardware development in an analogous way, the Smart-Its project, part of the European Disappearing Computer Initiative, explored different options for creating a modular system for prototyping smart artifacts and prototypical ubicomp systems.[4] The hardware was accompanied by detailed explanations of how it was built, and the software was open source, so others could easily replicate and extend the original designs.

The basic idea proposed in Smart-Its was a common processing and communications circuit board with a variety of ways to connect sensors and actuators (see Figure 1). Like the hardware, the software was also based on a modular approach; a core program provided basic communication and control, and additional libraries were associated with extra components.

In 2003, Hernando Barragán started to develop Wiring, a single-board microcontroller coupled with an accessible programming language and integrated development environment (IDE). The system targeted artists and designers and aimed to make it easier to create functional electronic prototypes.

The Arduino system (http://arduino.cc),[5] which, like Wiring, was conceived

INNOVATIONS IN UBICOMP PRODUCTS

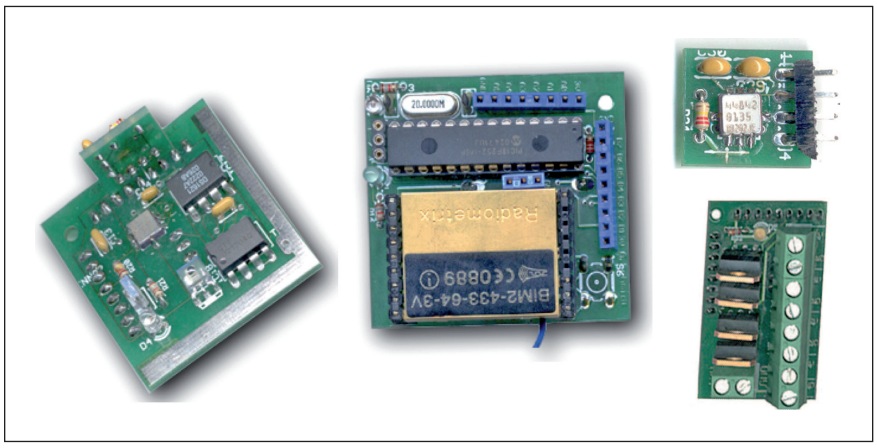
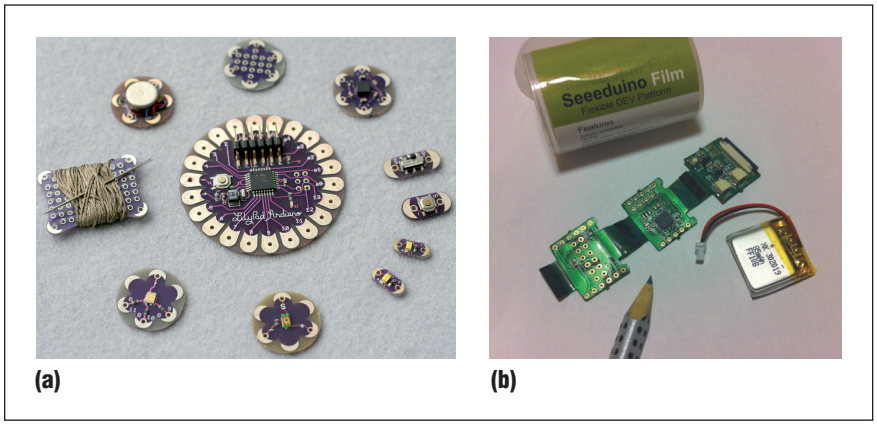


Figure 1. Smart-Its—an early prototyping platform developed in 2001.



(a) (b)

Figure 2. Arduino has revolutionized electronics development, and a very active community has evolved the hardware platform from the original standard printed circuit board (PCB) form-factor to include (a) wearable devices, such as Lilypad (photo courtesy of Leah Buechley) and (b) very small, flexible devices, such as Seeeduino film.

at the Interaction Design Institute in Ivrea, has taken the notion of low-cost and accessible components and tools much further. Arduino has arguably become *the* standard open source platform for physical prototyping. It's easy to learn and widely used in teaching,[6] in research, and by hobbyists. A lively community has emerged with many companies offering hardware, sensors, and actuators and a tremendous range of applications in the ubicomp space and beyond. Arduino hardware is available in many different shapes and sizes, such as the Lilypad[7] and the Seeeduino Film (see Figure 2). Some of the projects described on the Web include a laser-controlled harp, a "Tweeting" electricity-consumption meter, and a miniature Segway-like robot (see http://hacknmod.com/hack/top-40-arduino-projects-of-the-web).

A LOW BARRIER TO ENTRY

Platforms such as Arduino clearly have a low "barrier to entry." The integration of software and hardware is well conceived and implemented, making it relatively easy for anyone who wants to prototype new concepts that combine hardware and software to get started. The widespread availability of many simple example projects that can be modified and extended to suit new applications helps new starters benefit from the experience of the collective community. Even developers and designers with rudimentary electronics or programming skills can readily create interactive prototypes.

Taken to the extreme in terms of simplicity, a common "hello world" starting point for those picking up Arduino for the first time is a simple application in which an LED is illuminated when a user touches a sensor, such as a push-button switch. Almost any student, hobbyist, or professional developer can understand and replicate an example like this and start to alter it. Within minutes, they can be developing their own embedded electronics.

Of course, Arduino has its limitations. In particular, multimedia capabilities, such as image capture, audio processing, file system support, and advanced networking are much more involved, so the majority of systems tend to focus on relatively simple input and output. In cases where more complex functionality is required, a traditional PC is still often used, effectively turning the Arduino into an I/O device.

A HIGHER CEILING

Over the past few years, several additional platforms have been developed that complement the Arduino world in various ways. Microsoft's .NET Gadgeteer is a modular platform with a central "mainboard" containing a CPU and several sockets that can be used to connect a variety of sensors, actuators, displays, and communication and storage elements.[8] This solder-less composability of hardware components lets you construct prototypes in literally minutes and reconfigure and extend them just as quickly. From a programming viewpoint, code is written in the high-level C# language. The .NET Gadgeteer system is tightly integrated with the Visual Studio IDE, with features such as dynamic syntax checking, auto-completion prompts, and powerful debugging.

In combination, the two elements—easy hardware creation and powerful software development—offer a low hurdle for not only creating relatively simple projects but also for supporting more sophisticated applications. A typical "hello world" application for .NET

HELLO WORLD: CREATING A DIGITAL CAMERA USING .NET GADGETEER

The Microsoft.NET Gadgeteer system lets you build relatively sophisticated ubiquitous computing devices quickly and easily. One of the simplest examples of this is the construction of a self-contained digital camera—something that until very recently would have required considerable hardware and software development skills.

Creating a device starts with the graphical configuration tool (see Figure A1), which is part of the Visual Studio integrated development environment (IDE). The tool lets users specify which hardware components they want to use. Here, we need an image sensor, an SD storage card, a touchscreen, and a push button, along with the mainboard and power-supply modules. Having "wired these up" graphically, it takes just a couple of minutes to construct the corresponding hardware (Figure A2). Finally, the object-oriented C# language exposes a powerful set of event-based software libraries, which present a high level of abstraction to the programmer. This allows the basic digital camera functionality—that is, when the button is pressed, the device takes a photo, displays it, and saves it to a disk—to be specified in essentially just five lines of code. The IDE automatically generates the function definitions.

Figure B shows the application in its entirety. In the first instance, the program registers two methods to serve as event handlers: the first method executes when you press the button, which triggers a request to the camera to take a picture. Once the picture-taking process is complete, it triggers the second method, which shows the picture on the display and saves it to the SD card.

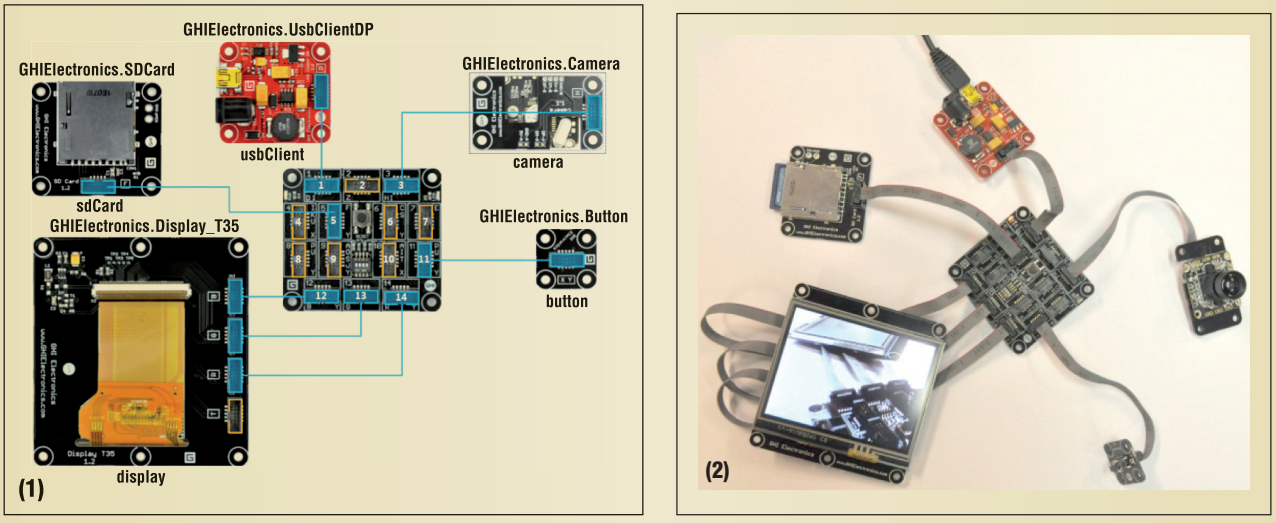


Figure A. Creating a digital camera using .NET Gadgeteer: (1) the graphical configuration tool and (2) the corresponding hardware.

```
void ProgramStarted()
{
    // Associate events with event-handling methods
    button.ButtonPressed += new Button.ButtonEventHandler(button_ButtonPressed);
    camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);
}

void button_ButtonPressed(Button sender, Button.ButtonState state)
{
    camera.TakePicture();
}

void camera_PictureCaptured(Camera sender, GT.Picture picture)
{
    // Show the picture on the display
    display.SimpleGraphics.DisplayImage(picture, 0, 0);

    // Save the picture to the SD card
    sdCard.GetStorageDevice().WriteFile("picture.bmp", picture.PictureData);
}
```

Figure B. The camera application in its entirety. The first method executes when you press the button, and the second method, which runs when the picture is captured, shows and saves the picture.

Gadgeteer is a digital camera—see the sidebar to understand how straightforward it can be to create such a device.

.NET Gadgeteer isn't a system to replace Arduino but rather a complementary tool for prototyping more complex digital artifacts, incorporating components such as touchscreens, cameras, SD cards, and Wi-Fi networking.

INNOVATIONS IN UBICOMP PRODUCTS

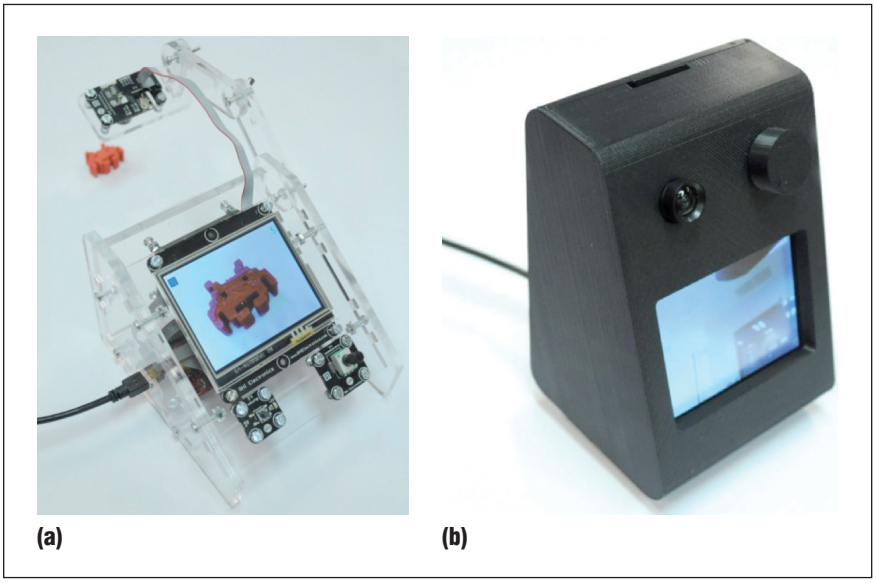


Figure 3. Two devices built with Microsoft's .NET Gadgeteer platform. Both contain similar components, but the different physical arrangements greatly influence the affordances of each device. (a) The stop-motion animation unit, which has a camera mounted on an adjustable arm, and (b) the telepresence device, which has a fixed camera pointing at the user. Rapid prototyping technologies such as laser-cutting and 3D printing are of great help here.

Each hardware module has an associated software library component that exposes its functionality in an intuitive and accessible way.

A variety of applications have already been realized using .NET Gadgeteer. These include simpler projects that resonate with hobbyists and enthusiasts, such as a stop-motion animation tool and small gaming units, and also more sophisticated research systems, deployed by academics to understand issues such as home heating control and consumer buying behavior.[9,10]

PRESENTATION IS EVERYTHING

As digital technologies become increasingly sophisticated, usability has become more important than ever. Many factors affect this, but two important elements are a device's form factor and the user interface it presents. Even subtle changes in these two regards can affect how users perceive a device and its possible uses and how easily they can operate it. In essence, while the functionality of any new device is clearly critical, the means of interaction, along with the product's shape and finish, are just as important.

In a previous issue, this column addressed how rapid-prototyping technologies, such as laser cutting and 3D printing, have given rise to a range of possibilities for instantiating new designs in a physical form.[11] These new technologies are a perfect complement to rapid-electronic-prototyping tools, such as Arduino and .NET Gadgeteer, because they enable enclosures to be created on a similar timescale to the development of the components they house. Together, they provide almost unlimited design options for the enclosure, flexibility in the selection and placement of components, and full control of the user interface via the application software. Furthermore, any of these elements can be altered, improved, or even rejected much more quickly and easily than is possible with traditional methods.

Figure 3 highlights how functionality, form, and presentation go hand-in-hand, showing two ubicomp prototypes similar to the digital camera described in the sidebar in terms of underlying components. Each device contains a digital image sensor, SD card storage, a rotary control, and a touchscreen display, along with power-supply and processor boards. However, the device's shape, the relative position of the various components, and the user interface exposed by the software all combine to suggest a different set of affordances in each case and ultimately help deliver a very different set of user experiences.

Where will new tools for the development of ubicomp concepts, such as electronic development platforms and rapid prototyping machines, ultimately take us? They'll inevitably continue to decrease the time required to turn new ubicomp ideas into real prototypes, allowing them to be immediately evaluated in the hands of real users and iteratively improved—all in a matter of hours and days rather than weeks and months. This acceleration of the product development cycle should help get new products to market more quickly and might also let product designers explore a wider range of concepts than is currently possible due to the time and cost involved.

But the increasing sophistication and accessibility of these tools might bring about an even more fundamental shift in ubicomp concept development by enabling people outside the traditional product development community to enter this arena. For example, there's a growing community of "Makers"—hobbyists who work individually and collaboratively to modify existing products, re-use components, and realize new ideas using a wide variety of skills and approaches. The Maker community is embracing Arduino and 3D printing and continually uncovering innovative applications and devices.

Not only do platforms like Arduino and .NET Gadgeteer empower these people to realize their product ideas, but the openness of the tools in terms of hardware designs and source code, coupled with services like Kickstarter (www.kickstarter.com), let people with little experience commercialize their ideas.

In the coming years, perhaps we'll see a wave of exciting new ubicomp products with roots in the Maker community. Some of these might be mass-produced, like many of today's consumer electronic devices. But there will also be an opportunity for individuals to produce hardware themselves and sell directly through online marketplaces analogous to mobile phone App Stores, which have become firmly established in the last few years. Such a "democratization" of ubicomp device development is an exciting prospect. P

REFERENCES

1. M. Addlesee et al., "Implementing a Sentient Computing System," *Computer*, vol. 34, no. 8, 2001, pp. 50–56.
2. S. Hodges et al., "SenseCam: A Retrospective Memory Aid," *Proc. 8th Int'l Conf. Ubiquitous Computing* (UbiComp 06), LNCS 4206, Springer, 2006, pp. 177–193.
3. S. Hodges and N. Villar, "The Hardware Is Not a Given," *Computer*, vol. 43, no. 8, 2010, pp. 106–109.
4. H. Gellersen et al., "Physical Prototyping With Smart-Its," *IEEE Pervasive Computing*, vol. 3, no. 3, 2004, pp. 74–82.
5. M. Banzi, *Getting Started with Arduino*, O'Reilly, 2008.
6. J.D. Brock, R.F. Bruce, and S.L. Reiser, "Using Arduino for Introductory Programming Courses," *J. Computing Small Colleges*, vol. 25, no. 2, 2009, pp. 129–130.
7. L. Buechley et al., "The LilyPad Arduino: Using Computational Textiles to Investigate Engagement, Aesthetics, and Diversity in Computer Science Education," *Proc. SIGCHI Conf. Human Factors in Computing Systems* (CHI 08), ACM Press, 2008, pp. 423–432.
8. N. Villar, J. Scott, and S. Hodges, "Prototyping with Microsoft .NET Gadgeteer," *Proc. Fifth Int'l Conf. Tangible, Embedded and Embodied Interaction* (TEI 11), ACM Press, 2010, pp. 377–380.
9. J. Scott et al., "PreHeat: Controlling Home Heating Using Occupancy Prediction," *Proc. 13th Int'l Conf. Ubiquitous Computing* (UbiComp 11), ACM Press, 2011, pp. 281–290.
10. V. Kalnikaitė et al., "How to Nudge in Situ: Designing Lambent Devices to Deliver Salient Information in Supermarkets," *Proc. 13th Int'l Conf. Ubiquitous Computing* (UbiComp 11), ACM Press, 2011, pp. 11–20.
11. A. Schmidt, T. Doring, and A. Sylvester, "Changing How We Make and Deliver Smart Devices: When Can I Print Out My New Phone?" *IEEE Pervasive Computing*, vol. 10, no. 4, 2011, pp. 6–9.

Steve Hodges is a principal hardware engineer at Microsoft Research, Cambridge. Contact him at shodges@microsoft.com.

Nicolas Villar is a researcher at Microsoft Research, Cambridge. Contact him at nvillar@microsoft.com.

James Scott is a researcher at Microsoft Research, Cambridge. Contact him at jws@microsoft.com.

Albrecht Schmidt is a professor of human-computer interaction at the University of Stuttgart. Contact him at albrecht.schmidt@computer.org.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.