Prototyping Smart Objects for Ubiquitous Computing

Matthias Kranz Albrecht Schmidt
{matthias, albrecht}@hcilab.org

Research Group Embedded Interaction
University of Munich
Amalienstrasse 17, 80333 Munich, Germany

Abstract. Technology enabling us to augment physical objects with electronics and thereby making them 'smart' is readily available. What could keep us from building smart objects? This paper keeps in perspective current development and technologies with a focus on ubiquitous computing appliances. The primary contribution of this work is the identification of issues related to prototyping hard- and software as well as to finally building these devices. Guidelines for development are presented and an outlook for future research is given.

1 Introduction

Never it has been easier to build and deploy 'smart' artifacts or 'smart' appliances out of the labs to our everyday environment. The required technology is readily available in pieces and in the necessary size and at low costs. But has it really become that easy to build smart objects? This paper keeps this in perspective and presents current issues related with the development of smart objects regarding hard- and software.

The paper is structured as follows. Section 2 discusses current trends enabling advances in ubiquitous computing relevant to the development of smart objects. Issues related to prototyping appliances are identified and discussed. In section 3 current platforms for building smart object systems are presented. Common issues related to development of embedded systems are presented. In section 4 an outlook on future development is given.

2 Challenges with Prototyping

2.1 Enabling Technologies

Computers have become ubiquitous in our life and the urge of modern people for pervasive interaction has enabled new industrial sectors like mobile communication. There are several trends in computer science and electronics that foster or even enforce ubiquitous computing. These are namely:

Computing Power: Computing power still increases according to Moore's Law[1], although a potential end is in sight. More and more components are integrated in less space. Energy consumption is further reduced, allowing longer operation with less power and heat, e.g. with nanoWatt microcontrollers.

Storage and Memory: Size and reliability of storage solutions have dramatically increased – accompanied with a dramatic decrease in price.

Connectivity: Miscellaneous devices can be easily (?) connected to each other or to a larger network by various networking technologies, e.g. WLAN, ZigBee, Bluetooth, IrDA, GPRS, UMTS, ... Connectivity is available nearly everywhere at low and decreasing costs.

Sensors: Sensors enable systems and devices to perceive their environment and allow, by processing the gathered information, the (hopefully intelligent) reaction to the perceived facts [2]. Also, prices decrease and accuracy and reliability increase.

Actuators: Reacting to changes in the environment or the context is essential for giving meaningful and appropriate feedback in place. The variety of available actors – starting with simple LEDs and ranging to robotic devices – is gigantic. The challenge is to select an appropriate actor available for the desired output.

Display Technology: Displays evolved from monochrome, small-size, much energy-consuming devices to large, bright and light plasma or TFT screens. Large public displays based on projection or RGB-LED technology inhabit our environment. But also small and flexible screens e.g. for e-books or wearable computing appliances are available.

Upcoming devices like PDAs (personal digital assistant), MDAs (mobile digital assistant) or UMTS smart phones provide pervasive computing power together with the required connectivity and a rich set of high-bandwidth interface technologies (WLAN, Bluetooth, ZigBee, IrDA, ...). '*By removing the restriction of place, people world-wide have found new and rewarding ways of connecting with others – both privately and for business*' [3], page V.

2.2 Starting development

When starting to develop appliances, we in first place speak of interactive devices. The probably most obvious way is to start with existing (digital) devices. These devices are already (hopefully) well-designed and have their well-known functionality and affordance [4]. The researcher can concentrate on the additional functionality that is to be integrated and does not have to design the device as a whole from scratch. The prototyping of completely new ubicomp devices and smart objects is a hard business, as stated in [5]. The problem is '*the sheer difficulty of developing and combining physical devices and interfacing them to conventional programming languages'*. We discovered the following problems with the development of new smart objects:

- difficulty of wiring electronics and electrical components together to a working whole thing
- problem of accessing functionality of commercially available devices (e.g. unpublished APIs)
- wrong level of abstraction of readily available devices (too high-level oriented, low-level functionality is shielded and not available to the programmer)
- cost of commercial/scientific devices make it impossible to access these devices in early stages of work, prohibiting to test developed software with real devices
- parallel developing and debugging hardware, software and protocols which makes it harder to locate and fix problems

So, '*building systems with physical sensors is no easy task. You need a soldering iron, plenty of experience in electronics, and even more patience*' ([6], page 96). Especially patience.

2.3 Prototyping

There are several ways of (rapid) prototyping related to standard software and WWW usability. But there is up to now only little work on the prototyping of smart objects though parts of the known methodologies can be reused. The following well-known items and their descriptions have been collected at [7].

paper-prototyping a paper sketch of the user interface with enough detail to make design decisions and usability evaluations, whether through a usability inspection, a focus group, or a simple user test.

mock-up prototyping usually referring to low-fidelity prototypes, such as paper illustrations, screenshots, or simple configurations of screens with limited interaction

wizard of Oz prototyping a prototype that only works by having someone behind-the-scenes who is pulling the levers and flipping the switches. The wizard of oz technique in user testing has a user interacting with an interface without knowing that the responses are being generated by a human, not a computer. This allows testing of some difficult interface concepts before a system is fully working.

video prototyping a technique for visualizing the interactive behavior of a system using video animation. Each frame of the video is created by videotaping a brief instance in the use of a system. Each frame is typically constructed on paper with icons, pointers, and other widgets simulated by moving cut-out representations of them.

Software prototyping is simpler than hardware prototyping. There are sophisticated tools available, e.g. HyperCard and other RAD (Rapid Application Development) tools. For hardware prototyping there are experimental platforms available for rapid hardware prototyping, e.g. Smart-Its [8] and Particles [9].

Prototyping can, amongst others, be classified according to how the prototype is built. The following approaches are presented in [6]:

- throw-away prototypes: the prototype is built and tested and thrown away after evaluation
- incremental prototypes: the final system is partitioned into smaller systems which are built and integrated step by step towards the final version
- evolutionary prototypes: each prototype serves as basis for the next generation of prototypes

It is essentially important when developing new smart objects and interaction mechanisms, that there exists a graspable and tangible demonstrator of the right form factor and no substitute.

There are countless studies on what people would do if something was possible or available. But results of these 'soft' studies are not transferrable 1:1 (if at all) to reality. The danger of this approach is that this leads to 'StarTrek' devices: people tell what they may have seen or read about in science fiction and how they liked it. Only real prototypes can be subject to formal evaluation.

Evaluation methods (taken from [6]) suitable are for example:

- expert evaluation
- cognitive walk-through
- heuristic evaluation, respecting Nielsen's ten heuristics [10]

2.4 Prototypes

To overcome the lack of suitable devices, several strategies could be used:

- prototype realization using a standard PC
- prototype realization using a mobile phone/PDA
- prototype by hacking of commercial off the shelf (COTS) devices
- prototype as self contained device

Instead of using a custom device, a standard PDA or any other system available (even sometimes a desktop PC or another COTS component could be used. But there are also disadvantages that have to be considered when using 'alternative' hardware than the desired or needed one. The user experience in front of a bright, large and high-resolution CRT Monitor is completely different from a small, dark and low-resolution monitor of a mobile device. This holds also true for the strain put on the eyes of the user. If fatigue is an issue, this has to be considered. The introduced changes when substitutes are being used have to be taken into account before conclusions can be drawn.

When considering using an existing device as basis for research, there basically are three options how to get to a prototype:

restricting the use of the existing device This concept was for example used in a public exhibit at the Cybernarium. The selection of the viewpoint in the 'Aula Regia' was done with a small number of buttons. These buttons where on a layout of the floor plan on top of a box. Inside the box was a standard keyboard - not visible to the user. For the user it appeared as a normal floor plan with several buttons [11].

re-packaging the existing device For example, a Compaq IPAQ was provided with a different housing to make it look like a completely different device. The screen was restricted and only visible through a round hole in the new housing [12].

hacking of devices Adding new hardware, especially sensors and actuators to existing devices is an easy way of extending their functionality and of obtaining novel devices [13].

The decision of which option to take depends on the task. A prototype with the affordances and form factor of the envisioned device surely is favorable has to be favored to alternatives.

2.5 Developing Ubiquitous Computing Appliances

'*The traditional computer is a glass box – all you can do is press buttons and see the effect. Ubiquitous computing and augmented reality systems break this glass box by linking the real world with the electronic worlds*' [6].

We strongly argue that it is crucial to have something the users can take in their own hands, explore it themselves (guided or unguided) and try out themselves in real life. The device may be buggy and crude. The user is able to deal with these shortcomings. However, the form factor is important. The experience generated is of great value.

When having a device or artifact – however it may look like – it has certain affordances [4], affecting how people use it. The physicality of the artifact makes people think over it when really holding it, handling it and using it. This often leads to new ideas and suggestions for improvements.

The key point is the engagement of the users with the device that makes them think more about it than without the device. The developer often is not able to take a step back and get another view on his artifact. Sometimes only outsiders may give new directions to the development and prevent bad designs as e.g. listed in [14] (e.g. bad door or light switch designs). A participatory design process involving users improves the overall design significantly, but it can not substitute formal evaluation methods of which some are listed in 2.3.

To conclude, we believe that the development of ubicomp devices can only successfully be done with real prototypes and real users.

3 Platforms for Ubiquitous Computing

In section 1, fundamentals and requirements for designing and developing ubiquitous computing soft- and hardware were discussed. In this section various platforms for ubiquitous computing, especially suitable for prototyping new appliances, are presented.

3.1 Platform Discussion

When talking about platforms, one has to distinguish between platforms dedicated especially as platforms for sensor networks. and those for appliance or smart artifact development. The following citation describes the differences well:

'*The emphasis is therefore not on routing and sensor data diffusion as investigated in large-scale wireless sensor networks, but on fast network discovery and exchange of context (as opposed to low-level sensor data)*' [8].

The characteristic of sensor nodes is that each node is (mostly) identical to the other nodes and has equal weight (in the sense of the importance of their data). Basically they all have the same properties and the same sensors. Nodes for smart appliances are at most similar, but not equal or identical at all and change from appliance to appliance often significantly. Also, from an application point of view, when using sensor nodes, one is interested in pure sensor data. When looking at appliances, one is interested in context.

There are different levels of abstraction.

high level: frameworks, methods
medium level: toolkits, IDEs
low level: pure programming languages like ASM, C, ...

When building devices with embedded systems, it is important to not have to start from scratch over and over again and thereby lose knowledge gathered during development. It sure is important to have some knowledge of the basic way of operation and maybe even the protocols used for e.g. RF data transmission, but this work has been done (several times) and there is no big prize to win (if at all) for doing it again. So the developer of ubicomp appliances shall also have a glance at what level of software is available for his preferred devices and choose the appropriate one for the task at hand.

The following Table 1 lists several platforms for prototyping smart objects. This is far from being complete, but mentions some of the most important platforms for ubiquitous computing hardware together with their technical characteristics and the programming languages available for them.

Platform Properties	BTNodes	Motes	SoapBox	Smart-Its	Particles
μController	ATMega128L	ATMega128L	8bit	PIC 16	PIC 18
Flash	128kB	128kB	unk.	128 kB	128 kB
RAM	4 kB	4 kB	unk.	4 kB	4 kB
RF Speed	76.8 kbps	250 kbps	10 kbps	125 kbps	125 kbps
RF Range	15 m	15 m	15 m	15 m	30 m
RF Band	2400	868 - 870[1]	868.3 MHz	868 MHz	868.35/315 MHz
Other Buses	SPI, I2C	RSR232, parallel	RS232	RS232,SPI, parallel, I2C	RS232, SPI,parallel, I2C
Other Comm. Capabilities	RS232, Bluetooth	Ethernet, ZigBee	-	IrDA, Bluetooth	Bluetooth, ZigBee, Ethernet (via USB/X-Bridge)
Digital I/O	unk.	unk.	yes	yes	yes
Analog I/O	yes	yes	yes	yes	yes
RTC	yes	yes	yes	yes	yes
Supply Voltage	0.5-3.3 V	2.7 - 3.3 V	1.5 - 28 V	3V	0.9 - 3.3 V
Size (widhtxlength)	60x40 mm	matchbox	matchbox	matchbox	45x18 mm
Prog. Languages	C	NESC (C dialect)	C	C	C

Table 1. Overview: Platforms for Ubiquitous Computing

BTNodes: BTNodes [15][16] are a '*hardware and software platform for rapid prototyping of augmented sensor networks systems, which may be temporarily connected to a backend infrastructure for data storage and user interaction, and which may also make use of actuators or devices with rich computing resources that perform complex signal processing tasks*' [15].

Motes: Motes were originally developed at Berkeley and are now commercialized by [17] and [18]. '*Crossbow's wireless sensor networking platform enables powerful, wireless, and automated data collection and monitoring systems*' [17]. Motes are supposed to form '*wireless sensors networks for large-scale commercial use*' [17].

SoapBox: '*It was developed as a research tool as well as to be utilized as a generic prototype for experimenting with product ideas and developing prototypes for them. ... As a configurable platform, SoapBox offers ultra low power implementation of wireless low-bit-rate devices such as small sensors, tags and actuators*' [19].

Smart-Its: Smart-Its [8] is an open embedded hardware platform with the goal of providing *'sensing, perception, computation, and communication*' [8] to everyday objects.

TecO Smart-Its Particles: Particles [9] can be seen as successors to Smart-Its. They are designed to be smaller and to consume much less power so they

[0] and/or 902 - 928, 433 - 434, 313 - 316, 2400 - 2483 MHz, depending on the basic Mote

can be deployed and remain in action for a long time (e.g. several weeks) without maintenance which becomes more important when conducting real life studies over a longer time period. The '*Particle Computer is a platform for rapid prototyping of Ubiquitous and Pervasive Computing environments, for Ad-Hoc (Sensor) Networks, Wearable Computers, Home Automation and Ambient Intelligence Environments*'[9].

3.2 Building Appliances with Embedded Systems

The platforms described above enable the development of smart devices and smart artifacts, but there are of course also special problems associated with embedded systems:

platform restrictions limited amount of RAM and ROM as well for stack and code

debugging no highly sophisticated tools and development environments, tracing of the program execution difficult or impossible

development level of the platform missing or only unsuitable protocols available which forces to spend noteworthy amount of development time in porting existing code for other platforms and requires substantial changes

difficulty of changing the hardware sometimes a different chip set and hardware parts are desirable or necessary. This often is impossible with the highly integrated nature of embedded systems

Knowledge on the peculiarities of the platform of choice has to be gathered and the difficulties have to be overcome to successfully develop ubiquitous computing hardware with the desired form factor. Greenberg in [5] also talks about the problems with custom hardware development when they talk about '*the sheer difficulty of developing and combining physical devices and interfacing them to conventional programming languages'.*

4 Conclusion - From Smart-Its to Particles and Beyond

Is it really the single device or smart object that is in the center of interest? The focus recently has shifted from building only single devices to complete appliances. But this only can be done with real prototypes. The existing hardware platforms, though each having some problems, provide a solid basis for building very interesting appliances.

In many projects, e.g. the Smart-Its project [20], there has been much valuable 'demonstrator building' and less focus on middle-ware and APIs. So the amount of existing higher level software available is still limited. This currently is a serious issue when researchers try to build new systems that are not limited to just one prototype or device. It is a challenging task to design and develop more generic and higher level toolkits and APIs for still growing field of research. Also systems are needed that integrate more than one ubicomp platform to enable researchers to combine the benefits of each platform.

Currently, we are just at the edge of these developments and have instruments for building sophisticated prototypes and with them, smart objects.

5 Acknowledgments

The work has been conducted in the context of the research project Embedded Interaction ('Eingebettete Interaktion') and was funded by the DFG ('Deutsche Forschungsgemeinschaft').

References

1. Moore, G.E.: Cramming more components onto integrated circuits. Electronics **38** (1965) 114–117
2. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. 2nd edition edn. Prentice-Hall, Englewood Cliffs, NJ (2003)
3. Hansmann, U., Merk, L., Nicklous, M.S., Stober, T.: Pervasive Computing. Springer (2003)
4. Gibson, J.: The theory of affordances. In: Perceiving, Acting, and Knowing, Lawrence Erlbaum Associates (1977)
5. Greenberg, S., Fitchett, C.: Phidgets: easy development of physical interfaces through physical widgets. In: Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST), ACM Press (2001) 209–218
6. Dix, A., Finley, J., Abowd, G., Beale, R.: Human-computer interaction (3nd ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2004)
7. Design, D.B.: Usability first. `http://www.usabilityfirst.com/glossary/main.cgi` (2005)
8. Beigl, M., Gellersen, H.W.: Smart-Its: An Embedded Platform for Smart Objects. In: In Proc. Smart Objects Conference (SOC 2003). (2003)
9. Decker, C., Krohn, A., Beigl, M., Zimmer, T.: The particle computer system. In: IPSN Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems (SPOTS), Proceedings of the ACM/IEEE Fourth International Conference on Information Processing in Sensor Networks. (2005)
10. Nielsen, J.: Heuristic evaluation. Usability inspection methods (1994) 25–62
11. Siemens Forum (AR): Demonstration Aula Regia on the CybernariumDays. `http://www.cybernarium.de/download/CDaysMuenchenKatalog.pdf` (2004)
12. Prante, T., Röcker, C., Streitz, N., Stenzel, R., Magerkurth, C., van Alphen, D., Plewe, D.: Hello. Wall - Beyond Ambient Displays. Video and Adjunct Proceedings of UbiComp 2003 (2003)
13. Andersen, K.: Hacking wireless game-pads. Toolkit Support for Interaction in the Physical World on Pervasive 2004 (2003)
14. Norman, D.: The Design of Everyday Things. Doubleday (1990) originally published by Basic Books in 1988 as 'The Pscyhology of Everyday Things'.
15. Beutel, J., Kasten, O., Mattern, F., Römer, K., Siegemund, F., Thiele, L.: Prototyping Wireless Sensor Network Applications with BTnodes. In: 1st European Workshop on Wireless Sensor Networks (EWSN), Berlin, Germany, Springer-Verlag (2004) 323–338
16. BTnode Project: BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks. `http://www.btnode.ethz.ch` (2004)
17. Crossbow: Motes. `http://www.xbow.com/` (2005)
18. Intel: Motes. `http://www.intel.com/research/exploratory/motes.htm` (2004)
19. Tuulari, E., Ylisaukko-oja, A.: Soapbox: A platform for ubiquitous computing research and applications. In: Proceedings of the First International Conference on Pervasive Computing (PERVASIVE), London, UK, Springer-Verlag (2002) 125–138
20. Gellersen, H.W., Kortuem, G., Beigl, M., Schmidt, A.: Physical Prototyping with Smart-Its. IEEE Pervasive Computing Magazine **3** (2004) 74–82