4 The Toolkit Approach for End-user Participation in the Internet of Things

Irena Pletikosa Cvijikj, Florian Michahelles

Information Management, Department of Management, Technology and Economics, ETH Zürich, Switzerland

Abstract Today, there are many end-user programming tools available, but in the Internet of Things domain, this concept is relatively new. Some pioneer examples include solutions, such as d.tools and Pachube, but also Web2.0, Mash-ups, Twitter and Facebook are suitable backplanes for this kind of applications. Another level of development support is various hardware concepts and solutions, such as RFIDs, Arduino, Violet, NFC, barcodes and many more. Appropriate user programmability could transform a system, multiplying the effectiveness of programmers and users. This article discusses how end-users can be empowered with new building blocks and tools, analogous to those that were emerging during the early phases of Internet growth. Accelerators, frameworks and toolkits are introduced, which would allow everybody to participate in the Internet of Things in the same manner as in the Internet through Wikis, Blogs etc.

4.1 From Internet to Internet of Things

Back in the 1960s, a group of visionaries saw great potential in allowing computers to share information, which contributed to the creation of what is today known as Internet. The first network, ARPANET, was developed for a very special purpose: the connection of just four major computers at universities in southwestern US (Salus 1995). This early form of Internet was founded by the government and was used only by computer experts, scientists and librarians for research, education, military and government purposes. There were no personal computers and anyone who used it, had to learn how to use a very complex system. Commercial uses were prohibited unless they directly served the goals of research and education. This policy continued until the late 80's, when the major breakthrough came with the introduction of Berners-Lee's World Wide Web. The new protocol he proposed gave simple information access to the general public by embedding hypertext into the Internet. In the following years, the lowering of barriers has made it simple enough, not just to use the Internet, but also to shape it and to add and generate new services. The second generation of the Internet, called Web 2.0 or *social web*, with its key supporting technologies, Ajax, Wiki, Blog, RSS, and Atom became ubiquitous, faster, and increasingly accessible to non-technical communities, thus introducing the rapid content generation and distribution, which lead to the richness of information of today's Internet.

It is evident that from its beginning the Internet was changing very fast and nowadays it is still evolving. However, instead of just connecting computers and/or wearable devices, it grew from a network linking digital information to a network relating digital information to real world physical items. This new network is called *Internet of Things* and provides embedding of physical reality into the Internet and information into the physical reality.

The concept of the Internet of Things became popular in 1999, when the Auto-ID Center at Massachusetts Institute of Technology (MIT) designed the RFID technology. Kevin Ashton, co-founder and director of MIT, in an article published in Forbes Magazine, in 2002, said,

> "...we need an internet for things, a standardised way for computers to understand the real world..."

The name of this article contained the first documented usage of the term Internet of Things in literature. In the following years this idea became more popular (Friedemann and Flörkemeier 2009) and in 2009 it was recognised as a general transformation of the Internet by the EU Commission (European Commission 2009).

Smart objects play the central role in the Internet of Things idea. Equipped with information and communication technology, everyday items provide a new quality by featuring tiny computers. These objects can store their context, they are networked together, they are able to access Internet services and they interact among themselves and with human beings. In order to connect everyday objects and devices to large databases and networks, a simple, unobtrusive and cost-effective system of item identification is required. Radio Frequency Identification (RFID) offers this functionality.

Based on this concept, many traditional industries, such as logistics, manufacturing and retail, have increased their effectiveness of the production cycle by implementing smart devices through RFID and barcode technologies. The cost of rudimentary RFID tags promises to drop to roughly \$0.05/unit in the next several years (Sarma 2001), while tags as small as 0.4mm × 0.4mm, and thin enough to be embedded in paper are already commercially available (Takaragi et al. 2001). Such improvements in cost and size will lead to a second wave of applications including vertical market applications, ubiquitous positioning and physical world web, i.e. the real Internet of Things. However, at the moment the Internet of Things is still mostly governed by business applications.

On the other hand, new generations of smart phones, sensor networks, open-source Application Programming Interfaces (APIs) and toolkits are becoming more and more pervasive. Still, these devices are mostly not customised to meet specific user expectations. Emerging trends of user programming (Scaffidi et al. 2005) give the opportunity to non-professional end-users of making additions to products, according to their specific needs. However, to let individuals play a main role in the Internet of Things, there are still a number of problems and challenges to overcome.

4.2 Problems and Challenges

The development of tools and techniques is a key focus for the concept of *participatory design* (PD). End-user tools and techniques should promote a practice where researchers and design professionals will be able to learn about users' work, and where users will be able to take an active part in technology design. To achieve this, these tools should avoid traditional design techniques with abstract representations, and instead allow users to more easily experiment with various design possibilities in a cost effective way (Kensing and Blomberg 1998).

Gronbaek et al. (Gronbaek et al. 1997) suggest the use of cooperative prototyping, where users and designers collectively explore the functionality and form of applications as well as their relations to the work in question. This type of cooperation requires access to adequate prototyping tools, which would act as *catalysts* and *triggers* in discussions about the relations between work and technology (Mogensen 1992), (Trigg et al. 1991). Tools and techniques used in PD projects should all have the common aim of providing designers and users with a way of connecting current and future work practices with envisioned new technologies.

Despite all the challenges, the need for innovation has been recognised and supported. Since Internet of Things systems will be designed, managed and used by multiple stakeholders, driven by different business models and various interests, these systems should (European Commission 2009):

- allow new applications to be built on top of existing systems,
- allow new systems to be deployed in parallel with existing systems, and
- allow an adequate level of interoperability, so that innovative and competitive cross-domain systems and applications can be developed.

Pioneer projects in this field, some of which are presented later in this chapter, have already been developed. Still, real-case user-driven scenarios do not exist yet, leading to a situation where the uptake of the technology itself is slowed down. To overcome this situation, end-users must be empowered with new building blocks and tools that were analogously emerging during the Internet growth.

4.3 Towards a Participatory Approach

The process of designing and developing new solutions presents a big challenge for scientists and manufacturers, even when it is about simple, everyday objects. When designing a new product, solutions are usually based on observations and usability conclusions. However, once the product has left the laboratory, the situation is completely different. The problem lies in the fact that designers are often drawn into the trap of trying to find uses for the tools, and deploying the coolest new features, forgetting their primary focus should be on providing value to the end user. However, the large variety of end users, usage conditions and scenarios usually leads to confusion and dissatisfaction regarding the usability of the product in the real world environment.

The concept of personalisation offers the solution for the described situation. Still, a designer working on a task of personalisation on an existing application, or building a new personalised application, is poised to make the classic error of putting technology before the needs of the end users (Kramer et al. 2000). Based on these observations, a new idea has grown, focusing on involving end users into the development process.

4.3.1 User-centered Design

User-centered design (UCD) is a broad term, used to describe a design philosophy and a variety of methods in which the needs, wants, and limitations of end users are placed at the center of attention at each stage of the design process. UCD differs from other approaches in trying to optimise the solutions based on how people can, want or need to use them, rather than forcing the users to change their working habits in order to comply with the offered approach. UCD is based on involving the users in different stages of the design process, from gathering ideas for functional requirements and usability testing, to direct involvement into the development process itself.

The term UCD was initially used by Donald Norman from the University of California San Diego (UCSD) in the 1980s and became widely used after the publication of his co-authored book "User-Centered System Design: New Perspectives on Human-Computer Interaction" (Norman and Draper 1986). Norman's work explained that involving actual users in the development process, preferably in the environment in which the product would be used, was a natural evolution in the field of UCD. Based on these statements, users became a central part of the development process resulting in more effective, efficient and safer products (Abras et al. 2004).

4.3.1.1 User-centered Principles and Activities

Today, there is an international standard, *ISO 13407: Human-centered design process* (ISO 13407 1999), that defines a general process for including human-centered activities throughout a development life-cycle. This standard describes four principles of UCD:

- active involvement of users,
- appropriate allocation of function to system and to user,
- iteration of design solutions, and
- multi-disciplinary design;

and four UCD activities:

- requirements gathering, understanding and specifying the context of use;
- requirements specification, specifying the user and organisational requirements;
- design, producing designs and prototypes; and
- evaluation, carrying out user-based assessment of the site.

The process involves iterating through these activities until the objectives are satisfied.

As a form of UDC performed during the design activity, PD, focusing on the participation of users in the development process, has gained strong acceptance, particularly in Scandinavian countries.

4.3.1.2 Participatory Design

PD applications are diverse in their perspectives, backgrounds, and areas of concern, leading to a lack of a single definition. However, in its essence, PD represents an approach towards assessment, design and development of various systems in which people, destined to actually use these systems, play the major role in designing and in the decision-making process. In other words, users are the co-designers of the systems.

The PD approach emerged in the mid 1970s in Scandinavia, under the name *cooperative design*. It was born out of the labor union's push for workers to have more democratic control in their work environment (Ehn 1989). However, when presented to the US community, due to the strong separation between managers and workers, *participatory* was the more appropriate description of the process, where managers and workers did not sit and work together, but rather work separately on the same problems, thus participating in the solution finding process. Pioneer projects included:

- Norwegian Iron and Metal Workers Union (NJMF) project, that took a first move from traditional research to working with people (Ehn and Kyng 1987),

- Utopia project (Bodker et al. 1987), (Ehn 1988), whose major achievements were the experience-based design methods, developed through the focus on hands-on experiences, emphasising the need for technical and organisational alternatives, and
- Florence project (Bjerknes and Bratteteig 1995) which has started a long line of Scandinavian research projects in the health sector, giving voice to nurses during the development of work and IT in hospitals.

Since then, PD projects have varied with respect to how and why workers have participated, from being limited to providing designers with access to workers' skills and experiences, where workers have little or no control over the design process or its outcome, to fully participating in the process. In all cases, worker participation is considered central to the value and, therefore, the success of the project (Kensing and Blomberg 1998).

Caused by the differences that can arise between users and designers, sometimes the users are unable to understand the language of the designers. Therefore, the development of innovative tools and techniques is a key focus for PD projects and their extension into the specific context of particular projects has become part of PD researchers' repertoire for action. PD techniques involve informal presentation of relations between technology and work, including visualisations (Brun-Cottan and Wall 1995), toolkits, prototypes and mockups. These tools and techniques promote a practice where both, the technology and the work organisation are in focus, and where users are able to take an active part in technology design.

At the dawn of the 21st century, technology is an inseparable part of everyday life, used at work, at home, in school, and on the move. This poses a new challenge to PD to embrace the fact that much technology development no longer happens as a design of isolated systems in well-defined working environments (Beck 2002), but instead community based development is becoming an emerging trend. This produces new techniques in development processes, such as open-source development, end-user programming, crowdsourcing and others, discussed below.

4.3.2 Open-source Development

Open source (OS) is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of OS is software or products of better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in (Open Source Initiative).

The concept of free software is an old one and can be traced back to the 1950s. First computers served only as research tools at universities, software was exchanged freely and programmers were paid for the programming as a process, and not for the software itself. When computers reached the business world, software became commercialised and developers started charging for each installation copy. In 1984, Richard Stallmann, a researcher at MIT, founded the Free Software Foundation (FSF) and the GNU project (GNU manifesto), thus providing the foundations for today's OS movement. Where proprietary commercial software vendors saw an economic opportunity that must be protected, Stallman saw scientific knowledge that must be shared and distributed (Dibona et al. 1999). The OS software movement has received enormous attention in the last several years. It is often characterised as a fundamentally new way to develop software (Raymond 1999) that poses a serious challenge (Dibona et al. 1999) to the commercial software businesses, which dominate most software markets today (Mockus et al. 2002). According to SourceForge.net, as of August, 2010, more than 240,000 software projects have been registered to use their services by more than 2.6 million registered users.

The OS development model fundamentally differs from the approaches and economics of traditional software development. First, the usual goal of an OS project is to create a system that is useful or interesting to those who are working on it (Godfrey and Tu 2000). Many successful OS software products have been and are being developed, distributed, and supported by an internet-based community of programmers, i.e. users themselves (Lakhani and von Hippel 2003). Developers are often unpaid volunteers who contribute towards the project as a hobby and there is no direct compensation for their work (Hars and Ou 2002). The question that this poses is the motivation for OS development? Eric Raymond reports (Raymond 1999) that there are at least three basic motives for writing or contributing to the OS projects: user's direct need for the software and software improvements, enjoyment of the work itself and the enhanced reputation.

The most fascinating development in the OS movement today is not necessarily the success of companies like Red Hat or Sendmail Inc. Instead, seeing major corporations like IBM and Oracle, turning their attention to OS as a business opportunity is intriguing. There is only one explanation for this behavior: innovation (Dibona et al. 1999). The new concept based on this, also known as *open innovation*, is using the OS as the most natural network for innovations (von Hippel 2002).

4.3.3 End-user Programming

A further type of community-based development is end-user programming (EUP). One way to define *programming* is as the process of transforming a mental plan of desired actions for a computer into a representation that can be understood by the computer (Hoc and Nguyen-Xuan 1990). Back in the 1940s, at the beginning of the era of computers, programming was done only by a small number of scientists, i.e. professional software developers. Since that time, software industry has been growing rapidly, and computer programming has become a technical skill of millions. In parallel, a second, powerful trend has begun to take shape, the so-called *end-user programming* (Myers and Ko 2009).

To understand the concept of end-user programming, it is important to explain the difference between professional and end-user programmers. While professional programmers develop software as a part of their jobs, end-user programmers are people who also write programs, but not as their primary job function. They are not formally trained in programming, yet need to program in order to accomplish their daily tasks. Spreadsheets are considered the major success story in end-user programming (Erwig 2009); however, many end-user programmers also use other special-purpose languages, or are even faced with the requirement of learning professional programming languages to achieve their goals. Despite the differences in priorities between professionals and end-user programmers, they both face the same software engineering challenges.

End-user programming has become so ubiquitous, that today there are more end-user programmers than there are professional programmers. According to the expert estimations (Scaffidi et al. 2005), in 2012, 90 million Americans will use computers on workplaces, significantly exceeding the 3 million anticipated professional programmers. Over 13 million workers will "do programming" in a self-reporting sense; and more than 55 million people will use spreadsheets and databases.

Due to the variety of end-user programmer profiles and backgrounds, there is not a single method for end-user programming. Instead, several techniques are being used, including programming by demonstration, visual and natural programming and many domain-specific languages and formalisms (End-User Programming).

What are the benefits of this approach? The obvious and most important one is: users know their problems best. Therefore, software products could become simpler, and at the same time more reliable. Only the general features will be supported by the issuing company, while details will be developed by end-user programmers, thus providing a richness of features that would otherwise be difficult to explain to a programmer. Allowing users to add their programs would give them freedom and responsibility at the same time. Therefore, it is beneficial for both users and product developers, to use these techniques and provide end-users with the possibility to shape products according to their needs.

4.3.4 Crowdsourcing

"There is an incredible story to be told about human ingenuity! The first step to its unfolding is to reject the binary notion of client/designer. ... The old-fashioned notion of an individual with a dream of perfection is being replaced by distributed problem solving and team-based multi-disciplinary practice. The reality for advanced design today is dominated by three ideas: distributed, plural, collaborative. … Problems are taken up everywhere, solutions are developed and tested and contributed to the global commons, and those ideas are tested against other solutions. The effect of this is to imagine a future for design that is both more modest and more ambitious." (Mau 2004)

The term *crowdsourcing* was initially used by Jeff Howe and Mark Robinson in 2006 for describing a new web-based, distributed problem-solving and production model that has emerged in recent years. Howe offers the following definition:

"Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call." (Crowdsourcing)

In other words, crowdsourcing is a process engaging the following major steps, (1) company posts a problem online, (2) a vast number of individuals offer solutions, (3) winning ideas are selected and awarded (Wikipedia). The difference between crowdsourcing and ordinary outsourcing is that a task or problem is outsourced to an undefined public rather than a specific other body. On the other hand, crowdsourcing differs from the OS practice. Problems solved and products designed by the crowd become the property of companies, who turn large profits off from this crowd labor. Technological advances are breaking down the cost barriers that once separated amateurs from professionals. New markets for the efforts of non-professionals open, as smart companies discover ways to tap the latent talent of the crowd. The labor isn't always free, but it costs a lot less than paying traditional employees (Howe 2006).

Crowdsourcing platforms appear to be a good place for young professionals or amateur affirmation. Learning, resource exchanges, comparison, recognition by peers or by companies, capitalisation of proposals and successes remain the main stimulation for creation, even if participants have a low chance of earning financial rewards (Trompette et al. 2008). Based on the principle of opening the design/development process to the crowd, initiated either by the company or the community, crowdsourcing provides the way to access external knowledge for the purposes of innovation, thus complying with the principles of open innovations.

4.3.5 Living Labs

Although the majority considers that the concept of Living Lab originates from William Mitchell, professor at MIT, Boston, it seems that the term Living Laboratory was initially used in literature by Abowd and his colleagues at Georgia Institute of Technology to refer to real-world contexts, in which users were given the opportunity to European state-of-the art technology (Abowd 1999). The concept was based on the idea of smart/future homes, serving as "real home" settings where real people were observed in their usage of emerging technologies. Markopoulos and Rauterberg gave broader definition by envisioning the living lab as

"... a planned research infrastructure that is pivotal for user-system interaction research in the next decade" (Markopoulos and Rauterberg 2000).

They have described the Living Labs as platforms for collaborative research that will serve as a basis for development and testing of novel technologies.

Today, the literature distinguishes between two different interpretations of the Living Labs concept (Folstad 2008):

1. Contextualised co-creation: Living Labs supporting context research and co-creation with users, and
2. Testbed associations: Living Labs as extensions to testbeds, where testbed applications are accessed in contexts familiar to the users.

The tendency to view Living Labs as environments for user-driven innovation is extremely interesting, since it meets both the industry needs for involvement of end-users in the early phases of ICT innovation and the end-users' needs for obtaining personalised solutions complying with their specific interests.

According to Folstad (2008), the main goals of a great majority of Living Labs identified in the literature are:

- Evaluate or validate new ICT solutions with users;
- Gain insight in unexpected ICT uses and new service opportunities;
- Experience and experiment with ICT solutions in contexts familiar to the users;
- Medium- or long-term studies with users.

It is interesting to note that all of these goals support the idea presented in this chapter: end-user participation in the Internet of Things domain of ITC development.

According to Eriksson et al. (Eriksson et al.2005) Living Labs differ from the other known user participation approaches. While the previously described approaches mostly involve the user in the development of the end-products, the Living Labs concept refers to an R&D methodology where innovations are created and validated in collaborative multi-contextual real-world environments with the individual in focus.

The concept of Living Labs is still emerging. The continuous growth of the number of organisations and initiatives whose intentions are to promote the concept and to provide support for collaboration between existing Living Labs is evidence to this. Examples of these organisations include Living Labs Global[52], European Network of Living Labs (EnoLL)[53] and Community-Based Living Labs to Enhance SMEs Innovation in Europe (CO-LLABS)[54]. Certain activities have also been undertaken by the European Commission[55].

[52] http://www.livinglabs-europe.com/

[53] http://www.openlivinglabs.eu/

[54] http://www.ami-communities.eu/wiki/CO-LLABS

[55] http://ec.europa.eu/information_society/activities/livinglabs/index_en.htm

Until now, Living Labs have been used in R&D environments for variety of purposes, e.g., ubiquitous computing, mobile ICT, retail innovations, online communities and collaborative work-support systems. However, we also see that the Living Lab approach seem to be suitable to meet the evolving needs for user participation in the field of Internet of Things.

4.4 Innovations to Users via Toolkits

A general trend toward heterogeneous consumer needs makes product development increasingly difficult (von Hippel 2001). Product developers face the problem that users hold an essential, but rather "sticky" portion of information required for product development. Von Hippel (Von Hippel 1994) defines *sticky information* as:

"…the information that is costly to acquire, transfer, and use in a new location".

The degree of stickiness is defined as the incremental expenditure required for transferring a certain unit of information to a specified locus in a form that is useable to the information seeker. When this cost is low, information stickiness is low; when it is high, stickiness is high. The traditional development approach typically engages companies in costly market research, because they assume homogeneity of needs within a market segment and thus can amortise over many consumers. This results in creation of somewhat different products for each segment, each intended to address the average customer needs in the selected segment (Jeppesen 2005). However, research on the issue concludes that a large share (about 50%) of the total variation in consumer needs will typically remain unaddressed in within-segment variation (von Hippel and Katz 2002).

In the manufacturing domain, user toolkits for innovation were recently proposed as a means to eliminate the (costly) exchange of need-related information between users and manufactures in the product development process. Two lines of argumentation have been brought forth to explain the potential benefits of toolkits for innovation and design: (1) the heterogeneity of customer preferences; and (2) the problems associated with shifting preference information from the customer to the manufacturer (Franke and Piller 2004). This approach allows manufacturers to serve the "markets of one", and to handle large and small customers in the same way, at the same time promising opportunity for entrepreneurs.

User toolkits for innovation emerged in a primitive form in the high-tech field of custom integrated circuit (IC) design in the 1980s as a result of enormous growth of price as custom IC products grew larger and more complex. Today, they range from food industry to software toolkits, allowing consumers to design key features by themselves. Depending on the type of toolkit, the outcome might be a product (Park et al. 2000) or an innovation (Thomke and von Hippel 2002). However, regardless of the underlying research area, the supporting rationale is the same: the toolkit allows the customer to take an active part in product development.

The user toolkits method is built around the idea of relocating a segment of problem-solving tasks with sticky need-related information to the consumer setting. The intention is to eliminate the need for information transfer and iterations throughout the development process by outsourcing tasks of product development to consumers. Toolkits should divide tasks, so that consumers primarily carry out tasks related to those areas of development that involve their sticky information. Letting consumers carry out essential *design-by-trial-and-error processes* avoids costly iteration and speeds up the process.

Von Hippel (Von Hippel 2001) argues that an effective toolkit for user innovation should enable five objectives. First, toolkits should enable users to carry out complete cycles of trial-and-error learning. Second, they will offer a well defined "solution space" that encompasses the specific designs. Third requirement is that well-designed toolkits must be "user friendly" in the sense that users do not need to engage in much additional training to use them competently. Fourth, they will contain libraries of commonly used modules, thus allowing the user to focus his efforts on the truly unique elements of that design. Fifth and finally, properly-designed toolkits will ensure that custom products and services designed by users can be produced on manufacturer production equipment without requiring revisions by manufacturer-based engineers.

Although an increase in opportunities for consumer involvement seems to raise the need for supporting consumers, there is a promising solution to this problem, namely, the establishment of consumer-consumer support interaction (Jeppesen 2005). A case study of Westwood Studios, an outlier in terms of firm support to consumers, shows that consumers who use toolkits may be willing to support each other. This complies with previous experiences on the field of OS development.

The new wave of ubiquitous computing, where everyday objects are augmented with computing capabilities, poses new challenges for designing interactive technologies. In terms of Internet of Things, collaborative programming in all of its forms is a growing research field, aiming to involve end-users, i.e. individuals already in possession of the relevant need-related information, and allowing them to actively participate in the development of the next generation of Internet of Things. The challenge is to create toolkits and frameworks that would provide reusable building blocks for recurring sub-tasks pertinent to users' problems. To achieve the high level of participation and excellent quality of resulting products, an environment has to be provided where barriers to contribute are low.

4.5 Existing Toolkits

This chapter is about accelerators, frameworks and toolkits that would allow everybody to participate in the Internet of Things in the same manner as it can already be done on Internet through Wikis, Blogs, etc. Typical software with end-user programmability features should at least have an editor, an interpreter or compiler, error checking and debugging tools, documentation and version management tools, as the bare minimum requirements. Today there are many EUP tools available (End-User Programming), but in the Internet of Things domain, this concept is relatively new. Some pioneer examples include solutions, such as d.tools and Pachube, but also Web2.0, Mash-ups, Twitter and Facebook are suitable backplanes for this kind of applications. Another aspect is the emerging usage of wireless devices and sensor technologies. Today, they can be found everywhere, including the wearable devices and smart phones. This served as inspiration for various hardware concepts and solutions, such as Arduino, Violet, NFC, barcodes, RFIDs, and many more.

4.5.1 I/O Boards and HW Based Systems

4.5.1.1 Wiring

> "Wiring is an open source programming environment and electronics I/O board for exploring the electronic arts, tangible media, teaching and learning computer programming and prototyping with electronics. It illustrates the concept of programming with electronics and the physical realm of hardware control which are necessary to explore physical interaction design and tangible media aspects." (Wiring)

Wiring was designed as a part of a master thesis by Hernando Barragán, at the Interaction Design Institute Ivrea, in 2004 (Barragán 2004). Wiring is based on OS principles. Its small I/O board represents a cheap standalone computer with many connection capabilities. The board can be used to control all kinds of sensors and actuators: sensors allow the board to acquire information from the surrounding environment, while actuators allow the board to create changes in the physical world (lights, motors, heating devices, etc). Wiring can also interact with other devices, such as PC/Mac, GPS, barcode readers, etc. It can be programmed using the Processing (Processing.org) language and numerous available libraries.

Processing is an OS programming language and environment for people who want to program images, animation, and sound. It is used by students, artists, designers, architects, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Processing is developed by artists and designers as an alternative to commercial software tools in the same domain.

Wiring has been used for numerous solutions (Wiring - Exhibition Archives). It has also been used as a basis for another prototyping platform: Arduino (Arduino).

4.5.1.2 Arduino

"Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments". (Arduino)

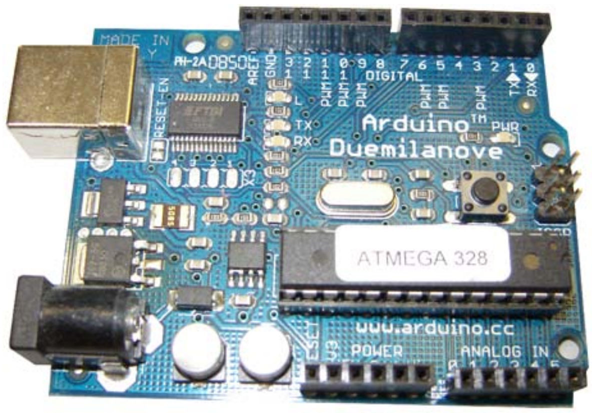

Fig. 4.1 Arduino Duemilanove Board Based on the ATmega168/ATmega328 Microcontroller

The Arduino microcontroller was originally created as an educational platform for a class project at the Interaction Design Institute Ivrea in 2005 in order to engage artistic and design-oriented minds. It was based on the previous work of the Wiring microcontroller, focusing on simplicity, a goal in pursuit of designing for a non-technical audience (Gibb 2010).

Arduino is a combined software/hardware platform. It can receive input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the hardware board can be programmed using the Arduino programming language and the Arduino Integrated Development Environment (IDE) (written in Java and based on Processing (Processing.org). Arduino supports two working modes, stand-alone or connected to a computer via USB cable.

Arduino offers everything that is needed for ubiquitous computing. That is the reason why its usage has overgrown the initial art and design solutions space. One very popular toolkit based on Arduino is LilyPad, described in detail in next section.

4.5.1.3 LilyPad

“The LilyPad Arduino is a system for experimenting with embedded computation that allows users to build their own soft wearables by sewing fabric-mounted microcontroller, sensor and actuator modules together with conductive thread...The kit was designed to engage kids (and adults) in computing and electronics and teach them fundamental skills in these areas by allowing them to creatively experiment with e-textiles in the same way that the Mindstorms kit allows people to experiment with robotics.” (Buechley et al. 2008)

LilyPad was initially designed and developed by Leah Buechley and SparkFun Electronics in 2003, providing large connecting pads, to create an interface between small electronic components and textiles, to be sewn into clothing. Various input, output, power, and sensor LilyPads are available.

A LilyPad board is based on the ATmega168V or the Atmega328V. Programming can be done using the Arduino IDE software. There are also several libraries that allow users to easily control an assortment of sensors and output devices. To program the LilyPad, a user clips it to a USB device that supplies the patch with power and facilitates computer patch communication. In order to create wearable electronic fashion items, at least the following modules are required: mainboard, power supply and a USB connection to download the software from a computer to the LilyPad mainboard.

LilyPad was used in several user studies (Buechley and Eisenberg 2008) and various end-user projects available on the Internet. Some of them are: a sonar garment for providing navigation assistance to visually impaired people by navigating the built environment, a blinking bike safety patch, an interactive passion sensing scarf, and many more.

4.5.1.4 MAKE Controller Kit

The MAKE Controller Kit (Making Things) represents a new generation of OS hardware platforms, successor to the Teleo (Making Things – Teleo) modular kits. This system was developed in collaboration with MAKE magazine and specifically embraces the “Do It Yourself” (DIY) subculture.

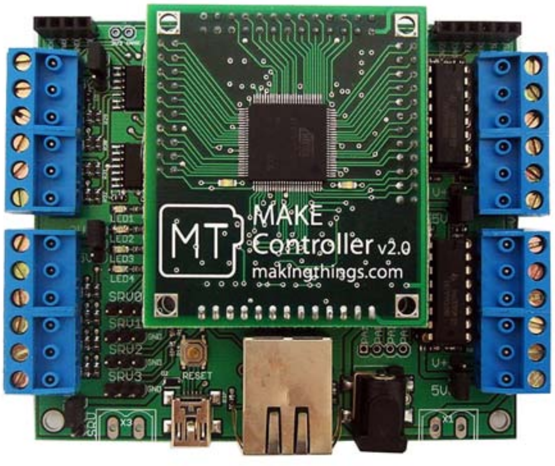

Fig. 4.2 The Make Controller Kit v2.0 Assembled with Controller and Application Boards[56]

The MAKE Controller Kit is targeted at enthusiasts and hobbyists. It is built out of two boards; a general controller board plugs into a specific application board and offers extensive features and interfaces. The controller board makes almost all signals from the chip available while the application board has application specific hardware (motor control, networking (Ethernet/USB/CAN/Serial/SPI) and a circuit protection.

This toolkit provides the possibility of designing a specific application board, including only the required features. Otherwise, the MAKE application board allows users to interface directly to the relevant devices (sensors, high current outputs for motors, etc.) and not worry that they're going to break the delicate controller board.

The MAKE controller kit comprises the software development environment. It can work as an interface to the PC when connected by Ethernet or a USB connection, or can be programmed to run standalone programs. A simplified API is available to program the board in C (freeRTOS operating system (FreeRTOS-A FreeRTOS)), so that the most difficult aspects of microcontroller programming are taken care of for less experienced users. Otherwise, full access to the chip is available for experienced coders.

The future steps in MAKE controller kit development consider providing further simplified programming environment similar to Wiring/Arduino.

[56] http://www.makingthings.com/store/make-controller/make-controller-kit.html

4.5.1.5 Phidgets

"Physical widgets, or phidgets, comprise devices and software that are almost direct analogs of graphical user interface widgets. Like widgets, phidgets abstract and package input and output devices: they hide implementation and construction details while exposing functionality through a well-defined API." (Greenberg and Fitchett 2001)

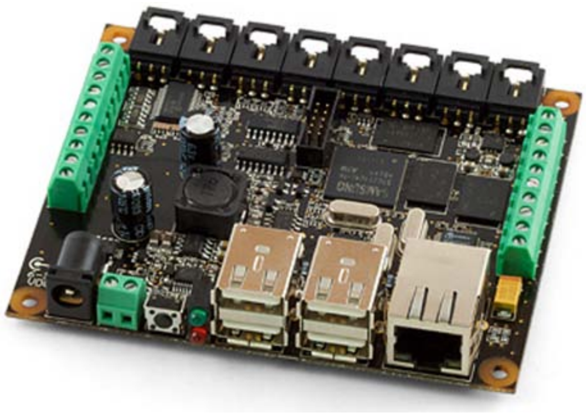

Fig. 4.3 The Phidget Single Board Computer (SBC) with an Integrated PhidgetInterfaceKit 8/8/8, 4 Full-speed USB Ports and a Network Connection[57]

In other words, Phidgets are a set of plug and play building blocks for interfacing the physical and the virtual worlds via low cost USB sensing and control from your PC. The system arose out of a research project at the University of Calgary in Canada and has later been commercialised (Phidgets, Inc.).

Phidgets includes USB-based hardware boards for input (e.g., temperature, movement, light intensity, RFID tags, switches, etc.) and output actuators (e.g., servo motors, LED indicators, LCD text displays). Its architecture and API let programmers discover, observe and control all phidgets connected to a single computer.

Phidgets are connected to a computer via USB, and are identified by the computer as a USB device. Each device knows and can transmit its phidget type, as well as an identification number that is unique for a phidget instance of that type. On the software side, all the required components are packed as an ActiveX COM Component.

Unlike widgets, each phidget component requires a corresponding visual component, providing a visual on-screen interface for interactive end-user control. Additionally, a connection manager tracks how devices appear on-line and there is a way to link a software phidget with its physical counterpart. And finally, there is a possibility for running in simulation mode, in order to allow the programmer to

[57] http://www.phidgets.com/products.php?category=0&product_id=1070

develop, debug and test a physical interface even when no physical device is present (Fitchett and Greenberg 2001).

The system has an extensive library of APIs and can be used with a large number of applications, even with other toolkits in some cases (Marquardt and Greenberg 2007). Using Phidgets enables programmers to rapidly develop physical interfaces without the need for extent knowledge in electronics design issues.

4.5.1.6 I-CubeX

"I-CubeX proprietary system is based on the MIDI communication protocol and offers modular components covering a large field of applications. With more than 10 years of experience in real-time sensor data gathering, I-CubeX is renowned for its ease of use, its variety of sensors and its robustness. It is widely used as a tool for prototyping, experimentation, research and teaching". (I-CubeX Online Store - Resources)

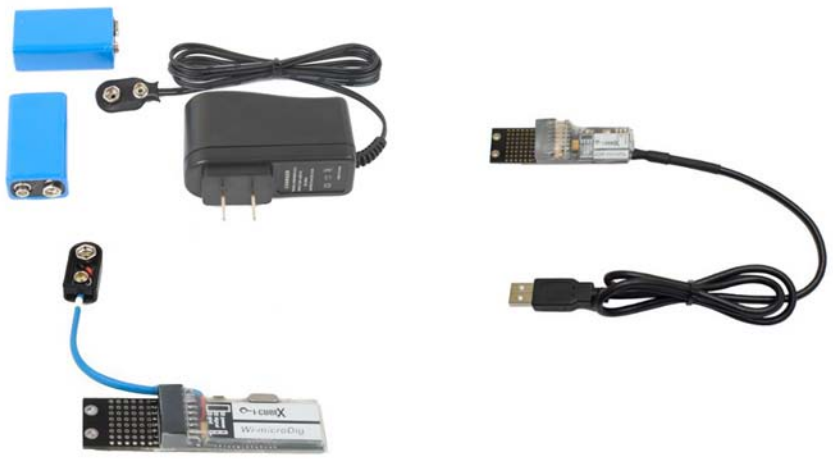

Fig. 4.4 Left: I-CubeX Wi-micro System, Including a Wi-microDig Analog to Digital Encoder with Wireless Bluetooth Transmitter, Cable, 9V Batteries and a BatteryPack-800; right: USB-micro System, Including a USB-microDig Analog Sensor Interface[58]

I-CubeX arose out of a research project in 1995 (Mulder 1995) directed by Axel Mulder at the Department of Kinesiology, Simon Fraser University, to address the need for better tools for artists to create interactive art and for musicians to more easily create or modify musical instruments. While I-CubeX helped to open up access to technology for artists interested in sensor technology; it in itself inspired others to create new technology.

[58] http://www.partly-cloudy.com/misc/

I-CubeX comprises a system of sensors, actuators and interfaces that are configured by a personal computer. Using MIDI, Bluetooth or USB as the basis for all communication, the complexity is managed by a variety of software tools, including an end-user configuration editor, Max (software) plug-ins, and a C++ API, which allows applications to be developed in Mac OS X, Linux and Windows operating systems.

Today, this system is intended for production or serious integration (I-CubeX Online Store – Demos) and the possibilities offered by this system are quite large. It can be used in science to obtain human performance data, and study animal and human responses to environments that change depending on the sensor data. In engineering, it can be used to develop prototypes of interactive products and to create innovative control surfaces enabling new ways to interact with multimedia. And in the arts, its environment of origin, it can create a responsive environment, build an alternate musical controller and develop novel interactive media pieces.

4.5.1.7 Comparison

Table 4.1 gives a comparison of the previously presented HW based toolkits.

Name	Wiring	Arduino	LilyPad	Make Controller[59]	Phidgets [60]	I-CubeX
Components	IO board, SW platform	IO board, SW platform	IO board, SW platform	2 IO boards, SW platform	IO board, sensors, motors, SW platform	digitisers, sensors, SW platform
OpenSource	yes	yes	yes	yes	no	no
Microcontrollers	ATmega128 ATmega1281 ATmega2561	ATmega8 ATmga168 ATmga328 ATmega1280	ATmega168V ATmega328	Atmel SAM7X processor, ARM7	PhidgetSBC	N/A
Memory	128K	16/32 KB (ATmega168/ATmega328)	16/32 KB (ATmega168/ATmega328)	256K	SDRAM 64MB	N/A
Programming Lan-	Wiring	Arduino	Arduino	C/C++	C/C++, Java	N/A

[59] Data corresponds to the Interface/Application Board.

[60] Data corresponds to Phidget SBC with integrated InterfaceKit 8/8/8.

guage						
Board OS	-	-	-	FreeRTOS	Linux	N/A
Tools Support	Wiring IDE	Arduino IDE	Arduino IDE	mcbuilder	Phidget IDE	Mac OS / Windows editor
PWM (analog) Outputs	6	6	6	4	N/A	N/A
Analog Inputs	8	6	6	8	8	N/A
Digital I/O pins	54	14	14	35 / 8	8I + 8O	N/A
MIDI Ports	-	-	-	-	-	IN/OUT
USB Ports	1	1	1	1	4	via MIDI-USB Adapter
Power	7-12V/USB	7-12V/USB	2.7-5.5V/USB	6-12V	6-15V	7.5V
External Interrupts	8	2	2	N/A	N/A	N/A
Hardware Serial Port	2	1	1	2.5-2/1	via Serial Adapter	N/A

Table 4.1 Comparison of HW Centered Prototyping Systems

Phidgets and I-CubeX distinguish from other toolkits since they represent complete systems with “ready-to-use” sensor and actuator/motor components, thus representing low-end solutions. Therefore, these two systems can easily be used by non-technical persons, without the need to go into detail regarding complex schematics and electronic issues. Additionally, I-CubeX differs from other systems in two more aspects. First, it provides only MIDI interfaces; other communication possibilities (USB, Bluetooth) are available via appropriate adapters. Second, I-CubeX has support only for Windows and Mac OSX, while all the other systems also have Linux support.

While Phidgets and I-CubeX are low-end solutions, high-end solutions, i.e. Wiring, Arduino, LilyPad and MAKE Controller Kit, provide more freedom for their users in the process of developing customised solutions.

4.5.2 SW Based Solutions

4.5.2.1 d.tools

d.tools is a hardware and software system that can be described as

> "…a design tool that embodies an iterative-design-centered approach to prototyping physical UIs." (Hartmann et al. 2006)

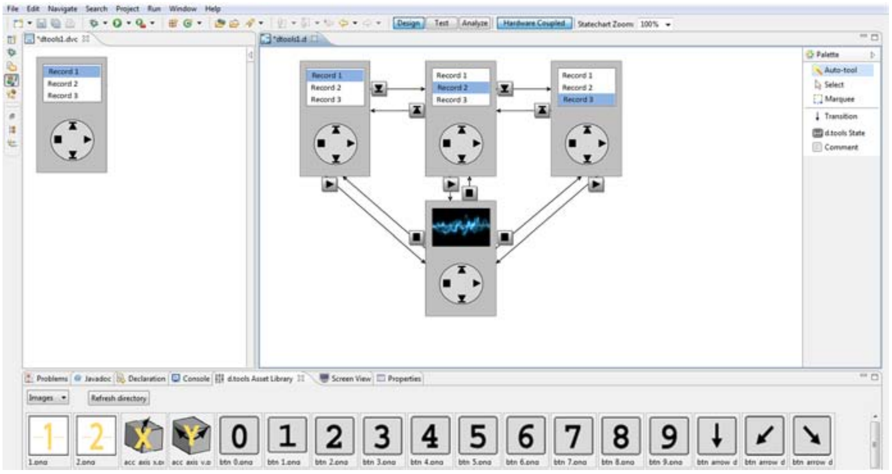
File Edit Navigate Search Project Run Window Help
Design Test Analyze Hardware Coupled Statechart Zoom: 100%
Record 1
Record 2
Record 3
Palette
Auto-tool
Select
Marquee
Transition
d.tools State
Comment
Problems Javadoc Declaration Console d.tools Asset Library Screen View Properties
Images
Refresh directory
0 1 2 3 4 5 6 7 8 9

Fig. 4.5 The d.tools Visual Authoring Environment Showing a State-chart for an iPod Shuffle Prototype[61]

d.tools was built to support design thinking rather than implementation thinking. With d.tools, designers place physical controllers (e.g., buttons, sliders), sensors (e.g., accelerometers), and output devices (e.g., LEDs, LCD screens) directly into the prototypes, and program their behavior visually in the provided software workbench. The d.tools visual authoring environment is implemented in Java J2SE 5.0 as an Eclipse IDE plug-in using its Graphical Editing Framework (GEF). The d.tools interface comprises a device designer, a state-chart designer, and associated views for specifying properties.

d.tools employs a PC as a proxy for embedded processors, so designers can focus on user experience-related tasks rather than implementation-related details. The d.tools library includes an extensible set of smart components that cover a wide range of input and output technologies. It provides plug-and-play prototyping of hardware components by adding microcontrollers to each component and networking the components on a common bus.

[61] http://hci.stanford.edu/research/dtools/gallery.html

d.tools can now be connected to other commercially available hardware platforms, such as Wiring boards, Arduino boards and Phidgets Interface Kits (d.tools).

4.5.2.2 iStuff

"iStuff is a toolkit for physical devices that extends the ideas of supporting wireless devices, a loose coupling between input and application logic, and the ability to develop physical interactions that function across an entire ubiquitous computing environment." (Ballagas et al. 2003)

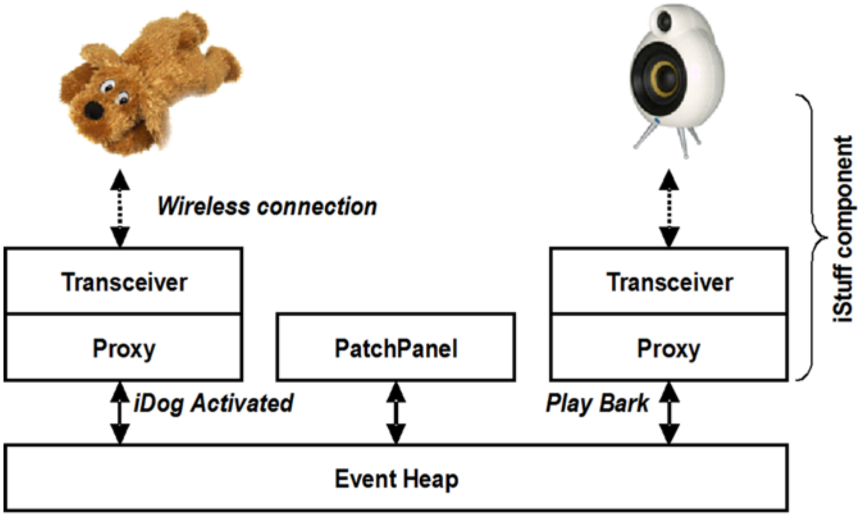

Fig. 4.6 The iStuff Components Architecture[62]

The iStuff toolkit was developed in a research conducted on a Computer Science Department at Stanford University in 2003. It was designed on top of iROS, a TCP and Java based middleware that allows multiple machines and applications to exchange information. Recently, its functionality has been extended with toolkits for mobile phone interactions (Ballagas et al. 2007).

iStuff leverages an existing interactive workspace infrastructure, making it lightweight and platform independent. The supporting software framework includes a dynamically configurable intermediary to simplify the mapping of input and output devices, such as buttons, sliders, wands, speakers, buzzers, microphones, etc., with their respective software proxies in order to create iStuff com-

[62] http://hci.rwth-aachen.de/istuff/tutorial.php

ponents. iStuff, in conjunction with the Patch Panel (Ballagas et al. 2004), enables standard UIs to be controlled by novel inputs.

4.5.2.3 Lego Mindstorms

Lego Mindstorms (LEGO.com MINDSTORMS) is a line of programmable robotics/construction toys, manufactured by the Lego Group. The hardware and software roots of the Mindstorms Robotics Invention System Kit go back to the programmable brick created at the MIT Media Lab.

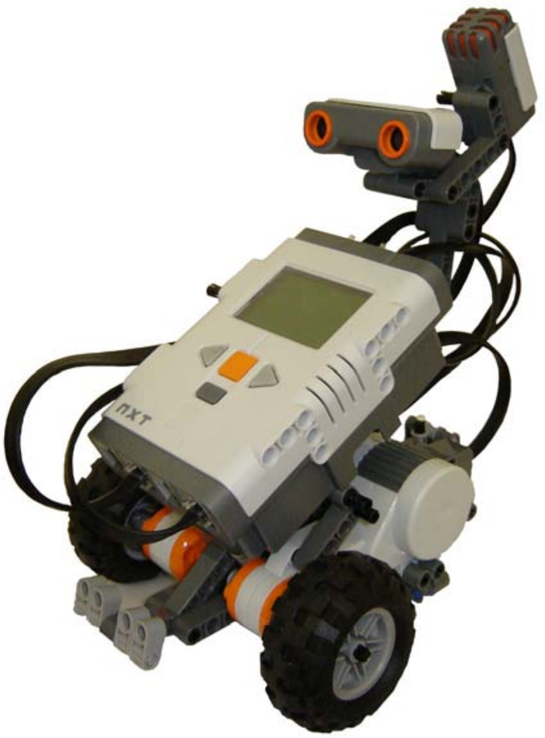

Fig. 4.7 Lego MINDSTORMS NXT in a Mobile Robot Configuration

The first Lego's visual programming environment, called LEGOsheets, was created by the University of Colorado in 1994 (Gindling et al. 1995). The initial Mindstorms Robotics Invention System Kit consisted of two motors, two touch sensors, and one light sensor. Today's NXT version has three servo motors and one sensor, each for touch, light, sound, and distance.

The programmable LEGO brick is the RCX, which transforms models into robots and controls their actions. LEGO provides two tools for programming the RCX. The first one is a development environment for programming the RCX with an interface that models programming as a process of dragging the puzzle pieces together to produce a chain (complete program). This GUI environment supports the basic programming constructs, such as loops, subroutines (though not true procedure calls), and concurrency. LEGO's second programming tool is a library for generating Visual Basic programs to control the RCX.

Lego Mindstorms may be used to build a model of an embedded system with computer-controlled electromechanical parts. Many kinds of real-life embedded systems, from elevator controllers to industrial robots, may be modeled using Mindstorms.

4.5.2.4 Pachube

"Pachube is a web service that enables storing, sharing & discovering real-time sensor, energy and environment data from objects, devices & buildings around the world. It represents a convenient, secure & scalable platform that helps in building the Internet of Things." (Pachube)

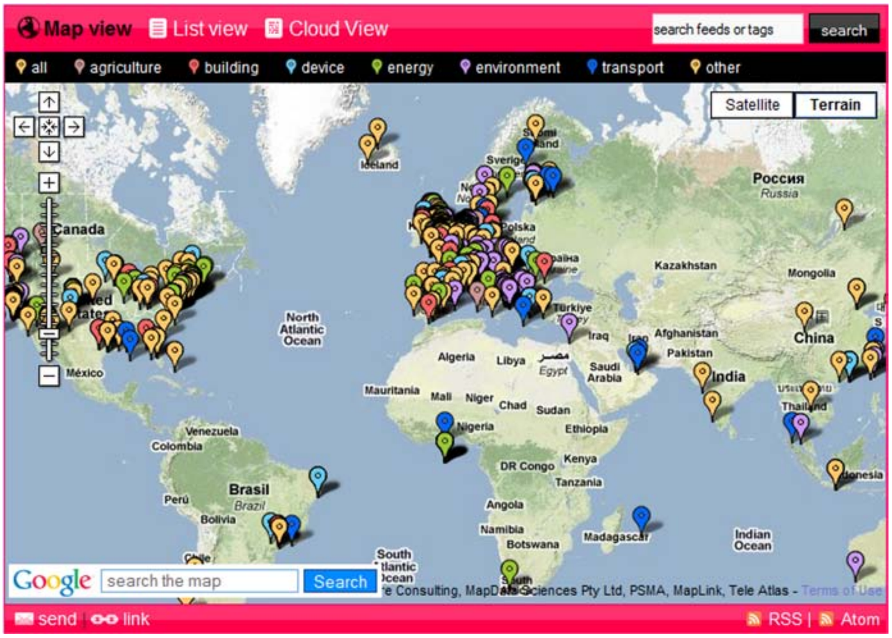

Fig. 4.8 Location of Pachube Sensors All over the World

The key idea behind the concept is to facilitate interaction between remote environments, both physical and virtual. Pachube enables a direct connection between these two environments, but it can also be used to facilitate many-to-many connections enabling any participating project to plug-in to any other project participating in real time, so that they could "talk" to each other and exchange data.

Apart from being used in physical environments, it also enables people to embed this data in web-pages and thus in effect to the blog sensor data.

Pachube uses Extended Environments Markup Language (EEML), which extends the construction industry protocol IFC. An extensive RESTful API makes it possible to both serve and request the data in all formats. Integration with other tools, such as Arduino, is also possible.

4.5.2.5 Comparison

The key issue to be discussed when comparing the SW prototyping platform is the support for the hardware platforms that they offer. Table 4.2 shows a comparison of all the relevant aspects of the previously presented SW based solutions that enable fast prototyping.

Name	d.tools	iStuff	Lego Mindstorms	Pachube
Components	Eclipse IDE plug-in	iStuff IDE	LabVIEW Graphical Programming Microsoft Robotics Studio IAR Embedded Workbench	Internet of Things platform, RESTful based API
SW Platforms	Java	Java	NBC[63]	Java, Ruby, PHP, .NET, Processing, Xquery, …
HW Platforms	Wiring, Arduino, Phidgets	Phidgets, MAKE Controller Kit, Smart-its, Powerbook Tilt Sensor, …	N/A	Zigbee, Phidgets, Arduino (+ WiShield / Danger shield), Sun SPOT, Home Automation Hub, …
OpenSource	yes	yes	no	no
Dedicated HW	yes	no	yes	no

[63] Next Byte Codes

Community	no	no	no	yes

Table 4.2 Comparison of SW Centered Prototyping Platforms.

Based on this comparison, we can conclude that there is greater variety in approaches when it comes to SW centered toolkits than HW centered ones. Lego Mindstorms is a closed platform, providing the possibility for usage only with the available LEGO HW components (sensors and servo motors).

Pachube, on the other hand, is not a suitable platform for developing customised solutions, but rather a platform for sharing the data on the Web inside the provided Internet of Things community. However, it provides integration of customised solutions based on the previously described HW toolkits.

D.tools and iStuff are the closest ones to the idea of toolkit based prototyping, providing direct support for creating and programming systems on the basis of the lower level HW based toolkits described in the previous section.

4.6 Discussion

Building on the potential benefits that the Internet of Things offers, poses a number of challenges, not only due to the nature of the underlying technologies, but also to the sheer scale of their deployment. Although it is essential for the mass deployment and diffusion, technological standardisation is still in its infancy, or remains fragmented. Successful standardisation in RFID was initiated by the Auto-ID Center and is now under the governance of EPC Global. The ZigBee Alliance, among others, contributed in the standardisation of wireless sensor networks, but in the case of nanotechnology and robotics, the situation is still very fragmented (International Telecommunication Union 2005).

On the other hand, research on toolkits for user design and innovation has been going on for the last 30 years. The conceptualisation of this phenomenon and the exploration of possibilities, limitations, and the underlying theoretical patterns of these new instruments only constitutes an initial step in this area. Although there are many research articles on technical aspects of toolkits and the production environment, many questions concerning the design side of individualisation and innovation still remain unanswered.

Greatest challenges and objectives to achieve when it comes to the toolkits usage for end-user participation are:

- *Diversifying and expanding the number of toolkits*. The greater the diversity and number of existing toolkits, the greater the diversity and number of toolkit users will be, leading to mass adoption. Modes of exploiting the toolkit approach depend directly on what the toolkit allows the user to do. The comparison of some of the existing toolkits presented in the previous chapter gave an indication that there is a great variety between the existing toolkits, not just from a perspective whether they are software or hardware based solutions, but also from the perspective of the level and form of user involvement in the innovation and prototyping process. Studies (Prügl and Schreier 2006) have already shown that "one toolkit may not serve all users effectively". End-users often try to surpass the limits of the design freedom provided in firm-constructed toolkits by employing tools from related fields and by expanding the scope of existing tools or even creating their own toolkits. Thus, different types of users employ different types of tools that in turn lead to different types of innovation activities, which support the previous discussion.

- *Focusing on low-end toolkits.* Highly technically skilled individuals represent a small percentage of the users' pool. Therefore, low-end toolkits could significantly increase the number of participating end-users. High-end toolkits offer a theoretically unlimited solution space within the production capabilities of the manufacturer, thus resulting in new functions or even new products. Innovations derived from this type of toolkit could be marketed successfully to a broader target group. However, not all users have the required skills to use such toolkits. Therefore, low-end toolkits might be more valuable when it comes to proposing new, individualised solutions. Additionally, not all users need radically new solutions. In the previous chapter we gave a brief overview of some of the most popular existing toolkits in the Internet of Things domain. However, the majority belong to the category of high-end toolkits, requiring a certain level of effort to learn how to use and configure them in order to create personalised solutions. Further development, with the emphasis on low-end toolkits, is required in order to achieve their mass popularisation and usage.
- *Exploitation of existing markets through individualisation.* Another possible way to exploit toolkits in the Internet of Things domain is to allow incremental innovations, product adaptations and individualisation. Individualisation platforms can be seen as valuable contribution due to the simplicity of usage. This approach has successfully been used in some industries, e.g., designing your personal watch, sneakers, etc., and has shown to be popular. Hence, in order to target a greater number of participants, the major field of toolkit applications should be the (further) exploitation of existing markets through individualisation.
- *Living Labs as innovation platform.* Living Labs can provide the missing ideas for real-life applications in the Internet of Things domain. Since the Internet of Things is still a relatively new and growing field, there is a requirement for the generation of ideas for possible applications in the Internet of Things domain. These ideas could significantly benefit the creation of new toolkits for participation. A popular approach for generating ideas, related to the open innovation process is Living Labs approach, presented previously in this chapter. Living Labs present an appealing opportunity for both generating ideas and at the same time creating and testing new solutions, based on the observation of the everyday activities and end-user requirements.

- *Revealing the source code.* In parallel to supplying a diversified toolkit, revealing the source code of the toolkit itself to users would enable them to push a toolkit's design limits. OS development was proven to be an efficient approach in supporting the open-innovation concept. Applied to the Internet of Things, it could speed up the process of integrating the Internet of Things in everyday life.

The common challenges related with the concept of involving end-users in the process of building solutions for new technologies are not the only ones that the Internet of Things is confronting. The complexity of the Internet of Things vision itself posses great challenges upon practitioners of the Internet of Things related EUP. The Internet of Things should not be seen as an extension of today's Internet, but rather as a number of new independent systems that operate with their own infrastructures (and partly rely on existing Internet infrastructures). The Internet of Things differs from the traditional Internet in many aspects that would influence the development of prototyping tools, from the characteristics and dimensions of the used hardware, growing size and diversity of "end nodes" of the smart objects' networks, standardisation issues for object identification and services it covers. The prototyping approach could give solution to at least some of these issues. Development of plug-and-play software and hardware prototyping platforms based on high level object abstraction could solve the challenges related to heterogeneity and complexity of Internet of Things network nodes, as well as the diversity of modes of communication.

From the business point of view, applying the described approach would lead to decreasing the product development costs and achieving greater customer satisfaction. Appropriate user programmability could transform a system, multiplying the effectiveness of programmers and users. Concepts like volunteering and OS have been used successfully for years. These observations could indicate that toolkit end-user participation is the right direction to go towards the future of the Internet of Things.

4.7 Conclusion

In this chapter we tried to answer the question: how to proceed in order to achieve the vision known as Internet of Things and identified toolkits as a major component in realising the vision. We have described how the Internet grew to its current form and then into the Internet of Things and we argued about the major problems and challenges faced upon its future growth. Then we gave a brief overview of the theoretical background regarding EUP and PD methodologies. In section 4.5 we illustrated early application examples of the previously described approach and finally we discussed specific issues that should be reconsidered when trying to apply PD in the Internet of Things domain.

Based on the given discussion, our general conclusion is that creating prototyping tools and techniques may be the right approach for achieving faster growth of the Internet of Things network and corresponding applications. When realising the described PD techniques, the general ideas of individualisation, in terms of providing users with different types of toolkits, and the widely adopted open software/hardware concepts should be considered. Standardisation of the Internet of Things network itself, with its high diversity and size, should be the initial step in this direction.

References

Abowd GD (1999) Classroom 2000: An experiment with the instrumentation of a living educational environment. IBM Sys J 38:508-530

Abras C, Maloney-Krichmar D, Preece J (2004 (in press)) User-Centered Design. In: Bainbridge W (ed) Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications

Arduino. http://www.arduino.cc/. Accessed 14 June 2010

Ballagas R, Ringel M, Stone M, Borchers J (2003) iStuff: a physical user interface toolkit for ubiquitous computing environments. CHI: ACM Conference on Human Factors in Computing Systems, CHI Letters 5

Ballagas R, Szybalski A, Fox A (2004) Patch Panel: Enabling control-flow interoperability in ubicomp environments. Proceedings of PerCom Second IEEE International Conference on Pervasive Computing and Communications

Ballagas R, Memon F, Reiners R, Borchers J (2007) iStuff Mobile: Rapidly prototyping new mobile phone interfaces for ubiquitous computing. SIGCHI Conference on Human Factors in Computing Systems

Barragán H (2004) Wiring: Prototyping physical interaction design. Thesis, Interaction Design Institute, Ivrea, Italy

Beck E (2002) P for Political - Participation is not enough. SJIS 14

Bjerknes G, Bratteteig T (1995) User participation and democracy: A discussion of Scandinavian research on system development. Scand J Inf Syst 7:73–98

Bodker S, Ehn P, Kammersgaard J, Kyng M, Sundblad Y (1987) A Utopian experience. In: Bjerknes G, Ehn P,Kyng M (eds) Computers and democracy: A Scandinavian challenge. Aldershot, UK: Avebury

Brun-Cottan F, Wall P (1995) Using video to re-present the user. Commun ACM 38:61–71

Buechley L, Eisenberg M (2008) The LilyPad Arduino: Toward wearable engineering for everyone. IEEE Pervasive Comput 7:12-15

Buechley L, Eisenberg M, Catchen J, Crockett A (2008) The LilyPad Arduino: Using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. Proceedings of the SIGCHI conference on Human factors in computing systems

Crowdsourcing. http://crowdsourcing.typepad.com/. Accessed 14 June 2010

Dibona C, Ockman S, Stone M (1999) Open Sources: Voices from the open source revolution. O'Reilly, Sebastopol, California

d.tools: Enabling rapid prototyping for physical interaction design. http://hci.stanford.edu/research/dtools/. Accessed 14 June 2010

Ehn P (1988) Work-oriented design of computer artifacts. Falköping: Arbetslivscentrum/Almqvist & Wiksell International, Hillsdale, NJ: Lawrence Erlbaum Associates

Ehn P (1989) Work-oriented design of computer artifacts, 2nd edn. Erlbaum

Ehn P, Kyng M (1987) The collective resource approach to systems design. In: Bjerknes G, Ehn P, Kyng M (eds), Computers and Democracy - A Scandinavian Challenge. Aldershot, UK: Avebury

End-User Programming. http://www.cs.uml.edu/~hgoodell/EndUser. Accessed 14 June 2010

Eriksson M, Niitamo VP, Kulkki S (2005) State-of-the-art in utilizing Living Labs approach to user-centric ICT innovation - a European approach, Centre of Distance Spanning Technology at Luleå University of Technology, Sweden, Nokia Oy, Centre for Knowledge and Innovation Research at Helsinki School of Economics, Finland

Erwig M (2009) Software engineering for spreadsheets. IEEE Softw Arch 26:25-30

European commission (2009) Internet of Things - An action plan for Europe. http://ec.europa.eu/information_society/policy/rfid/documents/commiot2009.pdf. Accessed 14 June 2010

Fitchett C, Greenberg S (2001) The Phidget architecture: Rapid development of physical user interfaces. UbiTools'01, Workshop on Application Models and Programming Tools for Ubiquitous Computing. UBICOMP

Folstad A (2008) Living Labs for Innovation and Development of Information and Communication Technology: A Literature Review. Electron J Virtual Organ Netw 10:99-131

Franke N, Piller F (2004) Value creation by toolkits for user innovation and design: The case of the watch market. J Prod Innov Manag 21:401-415

FreeRTOS-A Free RTOS for ARM7, ARM9, Cortex-M3, MSP430, MicroBlaze, AVR, x86, PIC32, PIC24, dsPIC, H8S, HCS12 and 8051. http://www.freertos.org/. Accessed 14 June 2010

Friedemann M, Flörkemeier C (2009) Vom Internet der Computer zum Internet der Dinge. Inform-Spektrum 33:107-121

Gibb AM (2010) New media art, design, and the Arduino microcontroller: A malleable tool. Thesis, School of Art and Design, Pratt Institute

Gindling J, Ioannidou A, Loh J, Lokkebo O, Repenning A (1995) LEGOsheets: A rule-based programming, simulation and manipulation environment for the LEGO programming brick. Proceedings of IEEE Symposium on Visual Languages

GNU manifesto. http://www.gnu.org/gnu/manifesto.html. Accessed 14 June 2010

Godfrey MW, Tu Q (2000) Evolution in open source software: A case study. Proceedings of the International Conference on Software Maintenance, ICSM 2000

Greenberg S, Fitchett C (2001) Phidgets: Incorporating physical devices into the interface. Proc. UIST 2001

Gronbaek K, Kyng M, Mogensen P (1997) Toward a cooperative experimental system development approach. In: Kyng M, Mathiassen L (eds) Computers and design in context. Cambridge, MA: MIT Press

Hars A, Ou S (2002) Working for free? Motivations for participating in Open-Source projects. Int J Electron Commer 6:25–39

Hartmann B, Klemmer SR, Bernstein M, Abdulla L, Burr B, Robinson-Mosher A, Gee J (2006) Reflective physical prototyping through integrated design, test, and analysis. Proc. UIST 2006

Hoc JM, Nguyen-Xuan A (1990) Language semantics, mental models and analogy. In: Hoc JM, Green TRG, Samurçay R, Gilmore DJ (eds) Psychology of Programming Psychology of Programming. Academic Press, London

Howe J (2006) The Rise of Crowdsourcing, Wired, Issue 14.06. http://www.wired.com/wired/archive/14.06/crowds.html. Accessed 14 June 2010

I-CubeX Online Store - Resources: About I-CubeX. http://infusionsystems.com/catalog/info_pages.php?pages_id=117. Accessed 14 June 2010

I-CubeX Online Store - Demos. http://infusionsystems.com/catalog/info_pages.php/pages_id/137. Accessed 14 June 2010

International Telecommunication Union (2005) ITU Internet Reports 2005: The Internet of Things. http://www.itu.int/osg/spu/publications/internetofthings/. Accessed 27 September 2010

ISO (1999) ISO 13407: Human centered design processes for interactive systems. http://www.iso.org/iso/catalogue_detail.htm?csnumber=21197. Accessed 27 September 2010

Jeppesen LB (2005) User toolkits for innovation: Consumers support each other. J Prod Innov Manag 22:347-362

Kensing F, Blomberg J (1998) Participatory design: issues and concerns. Comp Support Coop Work 7:167-185

Kramer J, Noronha S, Vergo J (2000) A user-centered design approach to personalization. ACM Computing Surveys, 43: 44-48

Lakhani K, von Hippel E (2003) How open source software works: Free user-to-user assistance. Res Policy 32:923-943

LEGO.com MINDSTORMS. http://mindstorms.lego.com/. Accessed 14 June 2010

Making Things. http://www.makingthings.com/. Accessed 14 June 2010

Making Things - Teleo. http://www.makingthings.com/teleo/. Accessed 14 June 2010

Markopoulos P, Rauterberg GWM (2000) LivingLab: A white paper, IPO Annual Progress Report 35

Marquardt N, Greenberg S (2007) Distributed physical interfaces with shared phidgets. Proc. of TEI'2007

Mau B (2004) Massive Change. Phaidon Press Ltd., London

Mockus A, Fielding R, Herbsleb J (2002) Two case studies of open source software development: Apache and Mozilla. ACM Trans Softw Eng Methodol 11:1–38

Mogensen P (1992) Towards a prototyping approach in systems development. Scand J Inf Syst 4:31–53

Mulder A (1995) The I-Cube system: Moving towards sensor technology for artists. Proc, the 6th International Symposium on Electronic Art

Myers BA, Ko AJ (2009) The past, present and future of programming in HCI. Human-Computer Interaction Consortium

Norman DA, Draper SW (1986) User-centered system design: New perspectives on human-computer interaction. Lawrence Earlbaum Associates, Hillsdale, NJ, Editors

Pachube: connecting environments, patching the planet. http://www.pachube.com/. Accessed 14 June 2010

Park CW, Jun SY, MacInnis DJ (2000) Choosing what I want versus rejecting what I do not want: An application of decision framing to product option choice decisions. J Mark Res 37:187–202

Phidgets, Inc. http://www.phidgets.com/. Accessed 14 June 2010

Processing.org. http://www.processing.org/. Accessed 14 June 2010

Prügl R, Schreier M (2006) Learning from leading-edge customers at The Sims: opening up the innovation process using toolkits. R&D Manag 36:237-250

Raymond ES (1999) The cathedral and the bazaar. http://www.tuxedo.org/~esr/writings/cathedralbazaar/. Accessed 14 June 2010

Salus PH (1995) Casting the Net: from ARPANET to Internet and beyond. Addison-Wesley

Sarma SE (2001) Towards the five-cent tag. Technical Report MIT-AUTOID-WH-006, Auto-ID Labs, 2001

Scaffidi C, Shaw M, Myers B (2005) Estimating the numbers of end users and end user programmers. IEEE Symposium on Visual Languages and Human-Centric Computing

SourceForge.net. http://sourceforge.net/. Accessed 27 September 2010

Takaragi K, Usami M, Imura R, Itsuki R, Satoh T (2001) An ultra small individual recognition security chip. IEEE Micro 21:43–49

Thomke S, von Hippel E (2002). Customers as innovators: A new way to create value. Harv Bus Rev 80:74–81

Trigg RH, Bodker S, Gronbaek K (1991) Open-ended interaction in cooperative prototyping: A video-based analysis. Scand J Inf Syst 3:63– 86

Trompette P, Chanal V, Pelissier C (2008) Crowdsourcing as a way to access external knowledge for innovation. 24th EGOS Colloquium, Amsterdam: France

von Hippel E (1994) Sticky information and the locus of problem solving: Implications for Innovation. Manag Sci 40:429–439

von Hippel E (2001) Perspective: User toolkits for innovation. Prod Innov Manag 18:247-257

von Hippel E (2002) Open source projects as user innovation networks. MIT Sloan School of Management Working Paper 4366-02

von Hippel E, Katz R (2002) Shifting innovation to users via toolkits. Manag Sci 48:821-833

Wiring. http://wiring.org.co/. Accessed 14 June 2010

Wiring - Exhibition Archives. http://wiring.org.co/exhibition/. Accessed 14 June 2010