LESS SCREENS, MORE OBJECTS!

Prototyping Networked Objects for the Internet of Things

By 2020, fifty billion digital devices will be interconnected, and form the so-called "Internet of Things". The digital experience that once shifted from the computer to the mobile device, shifts again into digital objects that sense and act upon the physical space. Smart thermostats, locks, light bulbs and moisture sensors that are already on the shelves of electronic stores give us a glimpse into such a future. Yet we are just at the very beginning of an ongoing development.

This project explores the Internet of Things with the aim of making it more accessible for designers. A conceptual framework, a design method, and a tool that facilitates the rapid prototyping of networked objects lay the foundation that allows designers to investigate the interconnected future through prototyping networked experiences.

LESS SCREENS,
MORE OBJECTS!

LESS SCREENS, MORE OBJECTS!

Prototyping Networked Objects for the Internet of Things

Master's thesis by Joël Gähwiler © 2015

Interaction Design
Zurich University of the Arts

Mentors
Prof. Dr. Karmen Franinovic
Björn Franke

By 2020, fifty billion digital devices will be interconnected, and form the so-called "Internet of Things". The digital experience that once shifted from the computer to the mobile device, shifts again into digital objects that sense and act upon the physical space. Smart thermostats, locks, light bulbs and moisture sensors that are already on the shelves of electronic stores give us a glimpse into such a future. Yet we are just at the very beginning of an ongoing development.

This project explores the Internet of Things with the aim of making it more accessible for designers. A conceptual framework, a design method, and a tool that facilitates the rapid prototyping of networked objects lay the foundation that allows designers to investigate the interconnected future through prototyping networked experiences.

Thank goes to all that supported me during this project especially Adina who always had my back with her endless curiousness in what I did, which resulted in having a great partner to test my ideas and concepts. Thank goes also to my mentors Karmen and Björn who have been very supportive and helped me to get the project going and opened many doors during the project and for future opportunities. Lastly, I would like to thank my colleagues Fabian and Yves who have been willing to go through this together by simultaneously enrolling in the program and having a great time pursuing each other's goals and dreams.

Contents

Introduction

Life today cannot possibly be imagined without the changes that have been brought about by the digital information age. Many things have become easier, more efficient and available at any time and location. For the sake of all these amenities, we let the computer replace social interactions like going out with a friend or talking to a bank employee. The screen has become an integral component of our everyday life and is the main terminal to the global data stream – the Internet. Because services like everyday shopping or taking out an insurance tend to be only available in the digital space nowadays, the two worlds of the digital and physical are even more separated than before and the rise of smartphones, mobile apps and the app stores continues. The continuing separation forces the user to connect the worlds back together by transcribing, translating and transferring data between them. However, dealing with these information systems, applications, operating systems and computers in general, the user comes up against his limits in regards to the complexity of those systems. This in turn leads to errors and an even stronger sense of disconnection between the physical and the digital. The human being is originally made to interact with objects in the physical space rather than concepts in a space that is abstract and intangible.

Today, we still have the chance to distance ourselves from that screen-centered model and turn to a more object-centered model. In contrast to the traditional Internet where applications interface with the world through screens and keyboards, the future user will interact with the world using interconnected, physical objects. Wherever we used computers and smartphones to access the digital world in the past, there will be objects that connect the two worlds together and allow us to interact with them directly. These "networked objects" will enclose the digital world by providing natural and physical interfaces. Complex computer systems will be hidden and

encapsulated in objects that are limited in functionality, specifically located and only accessible in the physical world. Thus, the digital world will be embedded in the physical world, and the combined interface will be more powerful than everything available before.

This is not a new idea and such future scenarios, visions and dreams have been described for example by the academic Mark Weiser as "Ubiquitous Computing" a long time ago. Already in 1991, he saw the need for such a transition, although they had literally just invented the personal computer. Recently, the idea of an interconnected future has resurfaced under the term "Internet of Things". It has become a buzzword of the century and has caught the attention of many, mostly however of business people that are looking for new ventures to invest millions in and push digital technologies even faster and further. We can already observe this change in my school's own electronic store where they have dedicated a shelve *(Fig. 1.1)* to such "IoT devices". Among other things they sell the "iGrill" *(Fig. 1.2)* which helps you find the right grill temperature for your steak by using a sensor that communicates with your smartphone.

"The world's most advanced grill thermometer" it says in capitals on the front of the packaging. On the back, they promote the product with the following sentence: "Now you can get back to your guests, go inside and watch the game or work on other dishes. Keep a close watch on the temperature progression of your meat right from your smart device and receive an alert once your food is ready to enjoy." As I was studying this product I wasn't sure what to make of it, yet it was clear to me that it would not be part of my future in this form. I could argue that grilling is a social interaction and that the device ruins that experience through an interfering technological abstraction. Although the question "How much technology is right?" is justified in this situation, this was not what troubled me. I think


I WANT / JE VEUX
hue
PERSONAL WIRELESS LIGHTING
App Store
PHILIPS
Color your
world with light
hue lux

Fig. 1.1 – Smart Devices Shelve at the ITZ Shop
Fig. 1.2 – The iGrill mini

that such a product definitely has a right to exist. Maybe it could be more of a supportive device that prevents me from burning my steaks. Maybe it even helps me to properly prepare the coal to have a good foundation for my grilling with friends. Possibly, the device could also support interactions with multiple users and not just the smartphone owner. Or, the notification mechanism could be subtle and integrated into another physical object. As I was turning all these ideas around in my head I realized that designers, especially interaction designers, would already have a proper mindset to think about such matters but are, as of yet, completely left out of this development process. In this moment it was clear to me that this has to change.

From early research into this topic I learned that designers are already aware of these future concepts and also would be up to investigating them. However, the technological barrier is still to high and especially when it comes to building objects that are interconnected many designers lack the skills to do so. Building prototypes has become an important part of the design process and ideas stand and fall with the ability to transport the concept into working physical objects. Recognizing this lack of proper tools and methods to apply in such situations, I dedicated my master thesis to diving into this world, learning to understand it, finding applicable technologies and methods, abstracting them and making them useful for designers. The core idea of this undertaking was to come up with a framework that can be used by any designer without having to make the same journey as I did. Instead, he or she should be able to focus on thinking about and designing for our future world.

In the following chapters of my Master's thesis I will present my exploration based on the research question: "How can the prototyping of interconnected objects for the Internet of Things become more accessible for designers?". The following second chapter introduces the terms "Internet of Things" and "Ubiquitous Computing"

and elaborates on these future scenarios. The third chapter will look at "Smart Objects" and related concepts and theories. Based on this foundation I will introduce my concept of "Networked Objects" and of how they will embed the digital in the physical world in the fourth chapter. The different design aspects of such networked objects will be dealt with in chapter five. In the sixth chapter, I will look at how prototyping can be used to support the exploration of such technological spaces. As communication is key to networked objects, I will present the tool and platform "shiftr.io" in the seventh chapter. The preliminary findings will be combined into a design method that can be used by designers to easily build prototypes of such networked objects. The method will be explained in chapter eight. The ninth and last chapter is a conclusion of the conducted research as well as an outlook on future opportunities and possibilities.

An Interconnected World

> Sal awakens: she smells coffee. A few minutes ago her alarm clock, alerted by her restless rolling before waking, had quietly asked "coffee?", and she had mumbled "yes." "Yes" and "no" are the only words it knows. […] Glancing at the windows to her kids' rooms she can see that they got up 15 and 20 minutes ago and are already in the kitchen. Noticing that she is up, they start making more noise. At breakfast Sal reads the news. She still prefers the paper form, as do most people. She spots an interesting quote from a columnist in the business section. She wipes her pen over the newspaper's name, date, section, and page number and then circles the quote. The pen sends a message to the paper, which transmits the quote to her office. […] *(Weiser, 1991: p. 7)*

A digital networked world, its social and economical impact, are topics that have been discussed many times in the past decades. Researchers have looked at possible future scenarios and tried to grasp, how our world would look and feel before, during and after such a development. In 1991, Mark Weiser wrote his popular article "The Computer for the 21st Century" and founded the research field "Ubiquitous Computing" (Ubicomp). Imagining a future in which computers are everywhere and networked, seemed somewhat magical at that time, as the researchers at Xerox PARC had only just invented the personal computer. In the end however, it was his vision that became feasible not only to his colleagues but also to the masses.

Apart from influencing decades of research in computer science, the research at Xerox PARC also paved the way for the exploration of new methods that help in understanding such future scenarios. Paul Dourish and Genevieve Bell, who reflect on the Ubicomp research field in their book "Divining a Digital Future", remarked that Weiser and his colleagues had their own investigation strategies

for such matters. “One of the laboratory’s strategies was to develop and widely deploy its technologies within its own environment and to live with and use them daily.” *(Dourish et. al, 2011: p. 12)* This approach was a very practical one and headed by PARC’s own researcher Alan Kay with the statement: “The best way to predict the future is to invent it.” In fact, the laboratory took this very seriously and developed for example the laser printer, the graphical user interface, the computer mouse and Ethernet, all things that influenced the world of today and its technology-driven society.

Today, we are facing another revolution in computer science that assembles under the term “Internet of Things” (IoT). In contrast to Ubicomp, the term “IoT” made it into society as a buzzword, a hype and a reason to invest money in many projects conducted by scientists and the industry. What started with smart cities and an efficient distribution of resources *(Evans, 2011: p. 2)*, has by now turned into a race between many different companies that are trying to launch the next digital connected product. We already see a lot of those products in our stores: smart light bulbs, thermostats, weather stations, door locks, a variety of health and movement trackers as well as power counters, and many more. Clearly this is just the very beginning of a family of products, that will soon interoperate and add intelligence to everyday life.

Interconnected Objects

In “The Computer for the 21st Century”, Weiser describes a morning in which the protagonist Sal wakes up and starts her day by speaking to the alarm clock that is able to track her sleep but only can reply “yes” or “no”. After that she gets notified by the window in the door to her kids room, that they are already up since 20 minutes and currently in the kitchen. In the last example she reads the news that is still in paper form but interactive, so that she is able to

mark a quote with a special pen, and send it to her office for further reading.

Weiser's description paints a picture of an environment that aids people in daily life through the wide application of invisible computers. Looking at it today, one can easily see how Weiser's objects are intelligent and full of features similar to the ones in today's smartphones. However, when we look more closely at each object we find several indicators that show us well chosen, functional limits. For example, the alarm clock is able to track Sals sleep to ask the question "coffee?" in the right moment, but will only accept a "yes" or "no". The functionality of the alarm clock is obviously limited to its main task: detect the awakening and ask the pre-programed question. Another limited object is the window on the door to the children's room, which shows when they woke up and where they are now. The window is physically connected to the room and thus the provided information is limited to the content of the specific room. Lastly, Weiser anticipated that some things, such as the newspaper, will not be replaced by technology, but are anyway linked to the digital world. The pen, that Sal uses to send the quote to her office, offers a very specific functionality again, which, compared to today, is provided by our smartphones.

The theoretical design of these objects by Weiser was heavily influenced by the upcoming debate about privacy at that time. However, the strict limitation in functionality and the physical link to the world make his designs an inspiration for new definitions of the qualities ubiquitous objects should possess.

> I live and work in a world animated by invisible spirits. Or at least, that's certainly how it seems. My house, my automobile, and my office are all constantly aware of my needs and movements. My fridge knows when I am running out of milk (and it orders more). My car knows what the weather is like and begins to de-ice itself as soon as I fill the dedicated

> car-beverage container with hot coffee. My office knows when (and where) I have parked my car, and alerts my clients (and coffee-maker) accordingly. Even the clothes I am wearing are part of this web of intercommunicating support systems. My shirt monitors my heart rate, temperature, and mood, and talks to the room and car when things look dicey. *(Clark, 2009: p. 4)*

This vision formulated by Andy Clark, a cognitive scientists with an affinity to design, is very similar to Weiser's vision but obviously inspired by the massive scale of today's Internet. Clark begins with making an initial statement that in his vision all objects are constantly interconnected, have access to each other's data stream and are thus able to track all interactions with the world and with other objects. The objects are also autonomous and able to interact with other systems in the users will, as the example with the smart fridge shows. The autonomy of objects continues in the examples about the self de-icing car as well as the office that coordinates other objects and also objects that belong to other individuals. The functionality in these examples lies clearly in the interconnection, rather than the specific objects. It seems that there are no boundaries, as the office is able to communicate with his client to push notifications about his whereabouts. The scope of Clark's description is different to Weiser's, as he is also speaking about the office and the car, environments in which many of such connected objects might interact and create a dense network. With his vision, Clark draws a beautiful image of a world, where the IoT made its breakthrough and the interconnection between objects is ubiquitous.

Synthesizing Weiser's and Clark's vision we clearly can envision that the future world has lots of smart objects. They stand at the border between the physical and digital world and act as an interface between both. More importantly, these objects form a network with similar kinds of objects and others that may be near or distant. While the single object is reduced in its functionality, the network

can follow a higher task or – with the user's consent – even act autonomously. Interconnected objects link the physical world and the digital world together, and transport information between them interchangeably. The computer becomes more hidden and encapsulated in the objects and new design strategies are required to not loose functionality but enrich the environment. Traditional objects that are un-connected and isolated do not need to be redefined and replaced with new ones, but can be augmented by new objects so as to preserve their original functionality.

The two stories by Weiser and Clark are, on the one hand, technological dreams that seem unrealistic and exaggerated. On the other hand, they are clear, conceptual designs of objects and spaces of which these futures could be made up of. Today, technology has progressed and the technological dream is becoming more realistic everyday. This also makes the design less conceptual and the ideas described in the articles more feasible. First products have been launched and strive to bring the computer ubiquitously into every thing around us. However, the immaturity of these objects calls for more dreams and conceptual designs that together lead towards a future where these ideas gain ground and are accepted by society. Today, the dreamers have to be designers who are willing to think the ideas through to the end and build the future they envision. And just as Weiser and Clark envisioned a whole new world and defined the qualities of the objects within, designers are required to define their own vision and qualities of such a future.

A World Made of Objects

The "Hue" light bulb *(Fig. 3.1)* by Philips was one of the very first products that was available in stores on the smart object shelf. The concept was rather simple: A strong multi-colored LED and a microcontroller are paired and built into a light bulb that can be used with traditional sockets. In addition to the classic switch, the user now has the ability to change the color of the light bulb using a mobile application. Furthermore, the light bulb can autonomously adapt to the environment and pick up colors of TVs and other devices to create a uniform light atmosphere. Philips offers various different light bulbs as well as configurable controllers, which will replace traditional light switches. Combined with the mobile app, the user can arrange, customize and program his individual lighting setup.

The product series by Philips and those of other manufactures are turning users into configurators and programmers of their products' systems. This shift is not radical in itself: In the web application market users have become configurators a long time ago and are nowadays combining multiple applications to, for example, build their own web shop. However, with the IoT, this change is now taking place in the physical space. Manufacturers that used to offer all-in-one systems for home automation will now begin to focus more on individual products that integrate neatly with other products.

Another example of a smart object currently on the market, is the "Nest" thermostat *(Fig. 3.2)* by Google that controls your heating and air conditioning system at home. The product claims that it will learn and understand your usage patterns and use that information to intelligently control your system in order to always maintain the right temperature and even save money while you are not at home. Of course, the thermostat does not calculate this data by itself, but

will upload all interactions and the current conditions into an information system operated by Google that will then process the data and send back commands to the object. With the IoT, this relationship between the object and the cloud will become even stronger as products are increasingly more connected to services that extend the product and give them a smart, intelligent character.

Amazon recently unveiled a new product called "Dash" *(Fig. 3.3)* which is an Internet connected barcode scanner that allows you to order products by just scanning their barcodes. Products you do not have at hand are added to the order by using the built-in speech recognition module. The Dash is a good example for a category of products that extend an already existing service (in this case, home delivery) by providing an alternative interface (the scanner) to the already available web or smartphone app (the web shop). Another example for this kind of products is the "ROCKI" *(Fig. 3.4)*, a music stick that can be attached to any set of speakers to stream music from online services like Spotify. The radically simple design and the ability to use the device in many scenarios, lets anyone upgrade their old speakers to work seamlessly with today's generation of streaming services.

Another category of smart objects consists of the hubs that coordinate the network and give the user a centralized control panel where general changes can be made to the system. For example, when you leave your home you most likely want to turn off all objects that are not needed while you are away. This kind of functionality would be provided by the hubs and controllers. Taking that concept a little further means that upcoming smart objects also extend existing smart objects and can combine several of those to a new kind of experience. Amazons Echo home assistant *(Fig. 3.5)*, is just a Box that integrates a speech recognition module which offers "Siri-like" functionality from any corner of the room. Apart from the privacy issues that have been raised concerning the constant upload

Fig. 3.1 - Philips Hue
Fig. 3.2 - Google Nest

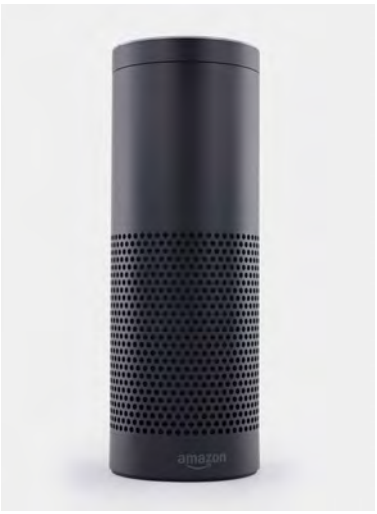


Fig. 3.3 - Amazon Dash
Fig. 3.5 - Amazon Echo
Fig. 3.4 - ROCKI

of sound recordings, the box is an interesting approach to a smart object that acts as an overseer for multiple smart objects.

The previous examples show a broad range of products that all assemble under the term "smart objects". Looking at the examples more precisely, one can see certain tendencies that indicate the direction the development might take. As a summary, one can classify existing smart objects by four different characteristics: 1. Smart objects that are configurable and programmable by the user to form individual and unique systems that integrate with other objects; 2. Smart objects that are connected to information systems that process, analyze and correlate the uploaded data in order to make smart and intelligent decisions. 3. Smart objects that provide an alternative input/output interface to existing services that may have previously been covered by a smartphone app; 4. Smart objects that are networked with objects similar to itself as well as others to allow configuration and integration beyond their default usage.

> The combination of the Internet and emerging technologies such as near-field communications, real-time localization, and embedded sensors lets us transform everyday objects into smart objects that can understand and react to their environment. Such objects are building blocks for the Internet of Things and enable novel computing applications. […] *(Kortuem et al. 2010: p. 46)*

Gerd Kortuem and his colleagues presented an architectural concept for smart objects in their paper "Smart Objects as Building Blocks for the Internet of Things". Their model is very focused on the industry as their examples show smart objects that deal with problems such as road construction and petrochemicals. Without going into details, the examples are well chosen to illustrate their argument, but are not very useful for the design discipline that is more focused on the end-consumer market. Anyhow, Kortuem and his colleagues make an interesting point by stating that "[…], smart

objects' true power arise when multiple objects cooperate to link their respective capabilities" *(Kortuem et al. 2010: p. 49)*.

In their "Road-patching case study" a link is made between smart objects that collect data from the construction machines and objects that enforce safety policies by turning the device on or off. The separation of concerns does not only simplify the development of the products but also supports the comprehensibleness of the system. This principle can easily be transformed to smart objects that reside in a user's home and, for example, observe several conditions of the house's climate or security. Furthermore, smart objects can use capabilities of other networked smart objects to execute their task in a better way or reach a higher, more complex goal.

Why shouldn't smart objects be networked together and make use of their neighbor's capabilities? In fact, if they would not do that, would we not end up with products that all have similar features like a smartphone today? Herein lies the power of the network: Smart objects do not need to be considered as devices that in themselves are complete and offer a full featured interaction, moreover they leave room to be extend with other smart objects and this provide a required set of features. The configuration by the user completes the system.

Tangible and Embedded

> Imagine an iceberg, a floating mass of ice in the ocean. That is the metaphor of Tangible User Interfaces. A Tangible User Interface gives physical form to digital information and computation, salvaging the bits from the bottom of the water, setting them afloat, and making them directly manipulatable by human hands. *(Ishii 2008: p. 15)*

The theory on Tangible User Interfaces (TUI) is crucial to smart objects, as they often also hide the intangible screen-based GUIs behind a interface that is physical, tangible and accessible. Yet, most smart objects come bundled with a mobile companion app that lets the user display collected data and change configuration settings. In the future these apps might become obsolete when these objects are interoperable and even more embedded. Of course, in theory, TUIs heavily link digital data to physical objects and make them more tangible by integrating haptic feedback and providing the user with the ability to play with the data by manipulating the objects. Still, smart objects are more flexible and may just rudimentarily link digital data and provide haptic feedback. Designers should still aim for as much tangibility as possible, since it will not just make their product stronger, but will also animate people to touch and play with the object.

> Here, we present the underlying concepts of embedded interaction, the technological and conceptual phenomena of seamlessly integrating the means for interaction into everyday artifacts. Technically, this requires embedding sensing, actuation, processing, and networking into common objects. Conceptually, it requires embedding interaction into users' everyday tasks. […] *(Kranz et al. 2010: p. 46, 47)*

Embedded Interaction, coined by Matthias Kranz and his colleagues, is a concept that takes ideas from TUI and enhances them for smart objects. Among the various projects the paper gives as examples, the context-aware kitchen utilities are the most interesting ones. The basic idea of the project is that the kitchen is a social space, therefore technology should stay hidden as much as possible to leave room for natural interactions. The introduced objects minimally augment existing objects with capabilities of sensing the user's actions. Sensors in the knife and the cutting board allow the system to detect the meal being cooked and use that information to make suggestions for other ingredients and/or possible variations. The

objects are very embedded and the actions are subtle, giving the user the possibility to ignore them.

The following passage form their paper describes exactly where the IoT is heading towards and what contribution the designer can make in this development: “Embedding interaction in an IoT context means integrating interaction opportunities into existing artifacts, devices, and environments. Unlike interaction devices, embedded interaction mostly utilizes objects people already use or are familiar with and broadens their impact and functions” *(Kranz et al. 2010: p. 50, 51)*. In fact, there are so many objects surrounding us already that have been shaped over many decades if not centuries. The call for designers should be to stop trying to reinvent the world on the screen with graphical user interfaces and instead turn to creating embedded interactions with already existing objects.

Whether they are interconnected light bulbs, learning thermostats, money linked barcode scanners or speech controlled automation hubs, the world is clearly rotating towards a future where smart objects play a bigger role. These objects have a physical representation and are full of digital technology. They call for new concepts that define their nature and make them understandable and usable. Earlier concepts like tangible user interfaces or embedded interactions began to uncover the beginnings and ends of these ideas. As designers we can use these concepts or, even better, make our own concepts about such products and start with designing experiences and interactions as we would like to have them. Another approach would be to combine those theories in a holistic model that serves as a framework and playground for experiments and prototypes that could enable many to visualize their own vision of a future.

Networked Objects

Computers have become omnipresent in our everyday life. Many interactions with others and our environment is based on digital technology and happens through the various screens we carry around, place in our home or have available in public spaces. Along with computers came the increasing dependency on digital information systems that drive many of today's processes and interactions. As these binary systems are not that intelligent in themselves, humans are needed to transcribe, translate and transfer data from the physical world to the digital and vice-versa. For example, photos are taken, tagged and uploaded to infinite data stores where the data gets analyzed and distributed to others that claim interest.

Cloud-based sharing services, such as the above-mentioned photo stream, are well designed systems that give us the ability to share data with our friends and, through that, the sense of being more connected and socially integrated. While this might hold true, we pay a high price for the feeling of connectedness by spending many hours in front of the screen to maintain our digital identity. It is becoming more and more apparent that this has to change, not only because the work is unpaid and flows into the pockets of others, but more importantly because as users we should use our time to interact with the world around us instead of living our lives in the virtual concepts of an operating system.

Through the IoT, the user's work is no longer required and can be replaced by intelligent objects that are connected to the Internet and can share data autonomously. All the work needed to link both worlds together is abstracted and hidden in objects that offer us natural and embedded interactions within our environment. Taking a picture and making it available to friends interested in our journeys, might be as simple as pressing a button on the camera to make

the picture show up directly in a distant picture frame. Because the camera knows its angle and location in the world, tagging is not necessary anymore and scrolling through an endless feed of photos becomes obsolete as it is intelligently handled for you. Applications and services become invisible and thus leave more space for interactions with the content through the physical objects.

The computerized world of today is heavily based on the availability of mobile devices, like smartphones and tablets, on which the digital experience is given by individual applications that we install on these devices. The screen, which is the central part of the device, is the canvas where the human and the artificial system find common ground and communicate. While the application paints the screen with graphical user interface elements, the human responds by interacting with these virtual elements on the screen. All these interactions happen in the digital space which requires the human to adapt to a limited, artificial system.

In the future, that same experience would be given by the individual networks of objects that reside in our environment. As these objects are located in the physical space, this space becomes the canvas. Rather than downloading applications that each solve a certain problem, we would organize objects in the space and configure them to networks that bring us the desired benefit. On the one hand, the hidden informations system would then paint the space using these objects. On the other hand, the same objects would observe us to give responses to the system. As these natural interactions would happen in the real, physical space, the artificial systems would need to adapt to us rather than the other way around. Suddenly, the human work of keeping both worlds in sync is not anymore necessary but is solved by the machines themselves.

Feedback Loops

As mentioned above, the primary smart objects we interact with today are smartphones and tablet computers. Their mobility, size and versatilely is immense and they are therefore usable in many situations. The built-in app-stores provide a sheer endless extensibility and connectivity to many services and applications in the Internet. Mobile devices are so successful because they offer access to information systems at any time and from any location. Yet these devices and their applications are very traditional, screen-based and inherit many concepts of computers as well as their problems and complexity. In order to replace such closed systems with interconnected objects that reside in the physical space and offer tangible, natural and embedded interactions we must understand the underlying flow of information.

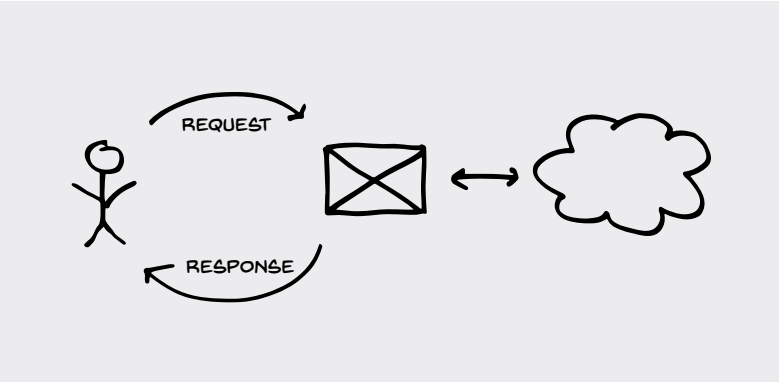
The feedback loop of a smartphone and its applications is mainly traditional *(Fig. 4.1)*. Through interacting with the device, the user gets access to one or more information systems. The interaction with the information system is limited to requests that are initiated by the user and responses that are given by the information system and visualized by the device. With this simple interface the user is able to retrieve data and manipulate it. The actions of browsing the Internet on a computer or interacting with a service through a mobile application are based primarily on the traditional feedback loop. Taking the example from before we would import a picture from our camera or shoot a photo directly with the built-in camera, into the application where we tag the photo with meta information like the location or hash tags. After hitting the upload button, the information system behind will match the entered meta informations with profiles of the other users and populate their feed. As soon as our friends open the application on their smartphone the picture shows

up in their feed and from there they have the ability to call on more functions provided by the application.

Today we can witness the first smart objects entering the market that claim to simplify our life by automating such cumbersome tasks. Mostly, these products are well known, everyday objects that are connected to the Internet through embed radio chips (see chapter 3). The constant connection to the Internet allows the object to send and receive data at any time without a previous action initiated by the user. Most of these objects just have sensors that observe the users or their environment and scan them for events that can be processed and transmitted to the cloud. There are some objects that also have actuators which can be accessed remotely and executed on a given command initiated by the information system. Smartphone operating systems introduced in the past years for example offer similar functionalities by pushing notifications to the user to inform him or her about changes in the connected information systems. Extending the traditional feedback loop we can now speak of a "connected feedback loop" *(Fig. 4.2)*.

In a connected feedback loop, the collection of data is simplified and automated as the object can autonomously upload sensor data without the need for any further instructions. From the traditional to the connected feedback loop we have therefore added the ability of the object to communicate with the information system directly without having to coordinate with the user. In our example, the camera would automatically use the GPS sensor to locate the user's position and add metadata to the picture. The camera could even be mounted and left and thus remotely activated to take pictures and upload them directly. On the other end, our friends would receive notifications about new pictures that have been added to the stream.

As of now, objects that use the connected feedback loop are mostly still screen-based so that the user has to rely on a computer,



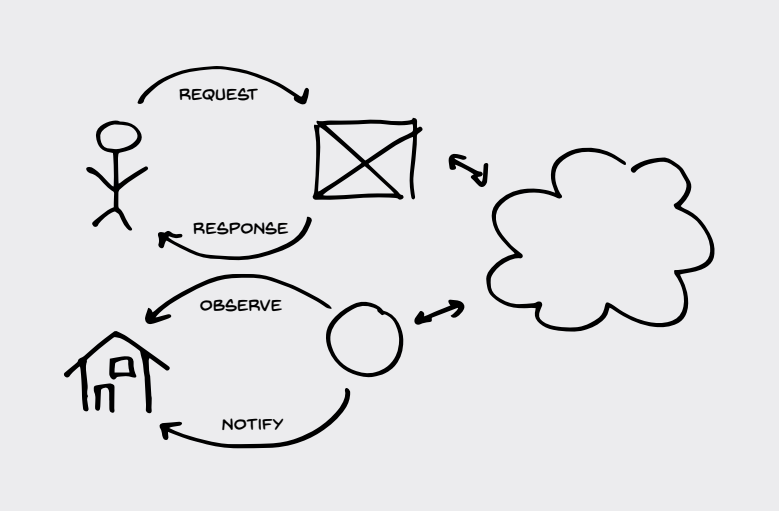
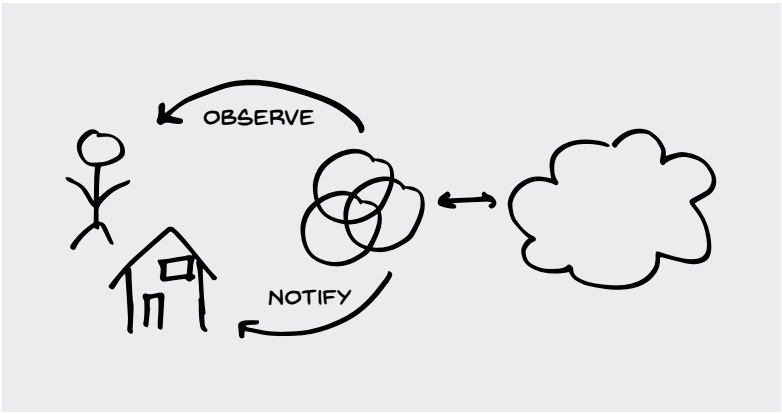


Fig. 4.1 - The traditional feedback loop

Fig. 4.2 - The connected feedback loop

Fig. 4.3 - The networked feedback loop


OBSERVE
NOTIFY

smartphone or tablet to fully interact with their information system. After all, a digital interface makes the interaction with a dynamic and changing information system much easier. However, to make the transition to objects that are truly embedded in our world, the screen needs to be eliminated completely.

In our example we can see that taking the photo and linking it to the user as well as the location is simple, but friends consuming the data may have different preferences and ways to do so. To support the individuality and personal interests of all these users the application needs to provide features that leave room to express these needs. On a smartphone, the user has the ability to change an application's state every second and adjust it to his current needs. These changes are constantly reflected on the screen and give the user a canvas where he can explore his individuality and express his interests. As physical objects are not digital pixels, their shape cannot change that easily and they are therefore not able to pick up needs in such a fast, responsive way. However, if the user could play with multiple objects, their location in the physical space, their connections to each other and thus their configuration, he would have the same options to express his individuality and needs. The objects would be networked *(Fig. 4.3)*, able to use each others capabilities and fully configurable by the user.

To our example, we can now add a picture frame that can be configured to always show the last picture published to an arbitrary feed the user has access to and of which he has one or multiple that are programmed in different ways. By placing the frames in his environment he has control over where the data is distributed to and where it is available. He could then add even more objects that can be connected to that information system, that offer a different way of interacting with the data. Therefore, the user does not any longer interact consciously with the information system and manipulate its data structure, moreover he gives it permission to observe him, his

interactions and the environment in order to control the information system in an inherent way. The virtual concepts of a computer are fully abstracted into natural and embedded interactions with physical objects.

With this paradigm in mind, many of the recently developed applications are just a step away from a future where the loop is closed and all the data in the cloud enters the physical space again. The networked feedback loop has the ability to relieve the user from interacting with technology and from the requirement of understandings its concepts and risks. Designers too are freed from designing for a screen and its virtual interface as they now have the chance to truly embed the technology and their service.

The interaction with digital objects will change as the feedback loop between humans and devices takes on another form. Rather than requesting information from remote information systems the object will observe the environment and react to events sensed by the device using either simple or sophisticated sensors. This shift in the experience of digital systems will of course bring technical as well as creative challenges. Building these networked objects that are highly interoperable and interconnected will demand new technologies, standards and tools. Designers will need to think about their products in a greater context, where the user gives meaning to the products through his individual configuration. Principles about functionality and interactivity need to be redefined and adapted to this new idea of symbiosis of the digital and physical space.

Designing Networked Objects

> The 10 principles of good design: 1. Good design is innovative; 2. Good design makes a product useful; 3. Good design is aesthetic; 4. Good design makes a product understandable; 5. Good design is unobtrusive; 6. Good design is honest; 7. Good design is long-lasting; 8. Good design is thorough, down to the last detail; 9. Good design is environmentally friendly; 10. Good design is as little design as possible; *(https://www.vitsoe.com/gb/about/good-design)*

Dieter Rams is a pioneer in designing objects that include new technology and are focused on the consumer home market. His iconic designs for several end-consumer products like radios, TVs, shavers, Hi-Fi systems and many more are famous. His simple and strict style was an inspiration to many designers and with the "10 principles of good design" he shaped his legacy and became an idol. His rules also form a good starting point for designing networked objects which are actually not that different to the objects that Rams used to design in his career.

In general, there are no specific rules for designing a networked object, except that it must be physical and include some kind of digital connection. When designing such an object however, a designer will encounter several design questions that apply mainly to networked objects, such as "What is the core functionality?" or "What is configurable and how?". The following section will look at various topics that will arise during the design process. Of course, design is individual and every idea is different, therefore the following suggestions are to be seen more as guidelines than rules. Together, these guidelines form a reusable framework of topics that can be applied to the design process of networked objects.

Functionality

The idea of networked objects is that they are simple, provide a limited set of features and are interoperable with others. The goal is less to design smartphone-like objects that have lots of functionality but more to aim for small objects that become "big" through their ubiquitous application. Similar to the Unix[1] philosophy we can state: "Design networked objects that do one thing and do it well." Focusing on a distinct set of core functionalities will leave more capacity to go into details and thus perfect the user experience. While the amount of features of each individual object might be limited, the system formed by all networked objects together, has a rich set of functionalities that are more comparable to a smartphone or a classic computer. It is normal that a networked object might be seem very simple and dull by itself, but that the system consisting of many objects becomes more complex and interesting.

The company "Edyn" for example created a set of products that help manage an outdoor garden. The first product called "Garden Sensor" *(Fig. 5.1)* measures the conditions of the soil and the weather. The second product is a water valve that can be controlled remotely. Through interconnecting several "Edyn objects" the user can build a system that will automatically water his garden. For fine-grained control and analysis the user can access his system via

1 The Unix philosophy, originated by Ken Thompson, is a set of cultural norms and philosophical approaches to developing small yet capable software based on the experience of leading developers of the Unix operating system. Early Unix developers were essential for bringing the concepts of modularity and reusability into software engineering practice, spawning a "software tools" movement. *(http://en.wikipedia.org/wiki/Unix_philosophy)*

a mobile and desktop interface. The example clearly shows that the individual objects in itself is not that technologically advanced but the overall system is. Taking that ideology we can formulate the first guideline: Functionality can be distributed among multiple objects, leaving the individual networked objects simple and comprehensible.

Certainly, not every feature can be transferred entirely to the physical space. In applications or systems that serve both the amateur and the specialist, an unrestricted access to its internal system is often inevitable. Additional screen-based solutions like mobile and desktop applications can therefore be considered to complete the experience. This concession leads us to the next guideline: While networked objects provide functionality on their own or as a networked system, mobile and desktop applications provide in-depth access and insights for professional users.

Interactivity

Networked objects themselves and the systems consisting of multiple networked objects are interactive systems. Their interactivity does not stem from the combination of a physical object with a digital connection. Moreover, they are interactive because of their characteristic to abstract and handle things automatically. To do so, the objects and especially the information systems in the background need to process and analyze behaviors to recognize patterns and adapt to the environment. In that sense, a system consisting of multiple networked objects is an artificial, self-regulating learning system *(Dubberly et. al 2009)*. Humans too can be viewed as self-regulating learning systems that, when interacting, converse with each other and with other systems:

> This type of interaction is a like a peer-to-peer conversation in which each system signals the other, perhaps asking questions or making commands (in hope, but without certainty, of response), but there is room for choice on the respondent's part. Furthermore, the systems learn from each other, not just by discovering which actions can maintain their goals under specific circumstances (as with a standalone second-order system) but by exchanging information of common interest. They may coordinate goals and actions. We might even say they are capable of design—of agreeing on goals and means of achieving them. This type of interaction is conversing (or conversation). It builds on understanding to reach agreement and take action. *(Dubberly et. al 2009: p. 75)*

The "Aether Cone" *(Fig. 5.2)*, a combination of loudspeaker and intelligent music player, is a good example for a networked object that stands in a conversing relationship with the user: "Cone is the first music player that thinks. It listens to your requests, picks up on your habits, and learns your tastes to create the perfect soundtrack for any mood or moment." *(http://www.aether.com)* In fact, whenever you change to another track by either turning the rim or naming one using the integrated speech recognition module, the object will remember the settings and create a listening profile. The next time the system gets activated on a Sunday morning for example, it will play music related to the last Sunday morning session. In that way, the Cone is learning about the user's habits and patterns, while the user is also learning about the Cone's algorithm and implementation. Over time, both will be able to understand each other quicker and more precisely. This kind of interaction opportunity reveals another guideline: The interaction between a single networked object or a system of many and the user can take the form of a conversation in which both sides learn from each other and understand each other's behaviors.

Fig. 6.1 - Edyn Garden Sensor
Fig. 6.2 - Aether Cone

Fig. 6.3 - Droplet on a Trash Bin
Fig. 6.4 - A Mother and her Cookies

Configurability

Another important design aspect of networked objects is configurability. As defined earlier, the physical space is the canvas where the networked objects and the user meet and interact. In that sense, the act of configuring, which is traditionally done by toggling buttons on a screen, now happens through placing, attaching and integrating objects in the physical space. The "Droplet" *(Fig. 5.3)* for example is an attachable button that helps keep track of recurring tasks. Creating or configuring this "To-Do" list can be done by simply attaching the button to an object and setting a timer using the smartphone app. In the future, that last step might be replaced with a simple controller on the button, so that the configuration is integrated in the object. Anyhow, the example shows quite clearly that the user could configure such a system by working solely with the physical objects: Networked objects offer configurability by allowing the user to place, attache and integrate them with the physical space and by having a few physical controls for fine adjustments.

Topology

Every user and environment is unique. If not pre-defined, the topology and location of networked objects within the user's space is completely up to him and his needs. The design of networked objects can give hints and lead users towards putting the objects in the intended spots. The design should support any possible variation, giving as much freedom as possible. A simple way to control and lead the user towards a meaningful topology of networked objects is by assembling and distributing packages that are targeted to various use cases. While experienced users can buy the objects individually, newcomers can buy a kit that is tailored to a certain common use

case. Such a package would include the necessary objects and guides to get started.

An illustrative example here is the object tracking system “Mother” *(Fig. 5.4)* by Sen.se. The product consists of a controller, the Mother, that observers many “cookies” which are attached to things by the user. As the equipped objects are used by the user, the cookies send data to the mother, which processes that information and makes it available to other systems. The starter set includes the controller as well as four sensors, with which the user can already start to build his system. To expand the experience the company offers additional packages of sensors to integrate even more objects. From that we can derive the last guideline: Tailored packages of networked objects give the user an example of a topology of objects that can then be extended with additional packages.

Networked objects bring about new design challenges in the areas of functionality, interactivity, configurability and topology. As we are dealing with a fairly recent field, we cannot look back on several decades of learnings and best practices. However, the guidelines outlined in previous chapter give a starting point from which further discussions can originate. Apart from looking at the above-mentioned challenges, the most crucial part of designing networked objects is the overall system experience. What rules are involved when functionality and thus interactivity and configurability are spread among multiple devices that may not even be in the same room? Such a system experience can only be devised if the designer has the ability to explore the possibilities and play with the objects as the end-user would do. Prototyping these objects will be a very important part of a successful design process.

Prototyping Networked Objects

> We can look at prototypes as both concrete artifacts in their own right or as important components of the design process. When viewed as artifacts, successful prototypes have several characteristics: They support creativity, helping the developer to capture and generate ideas, facilitate the exploration of a design space and uncover relevant information about users and their work practices. They encourage communication, helping designers, engineers, managers, software developers, customers and users to discuss options and interact with each other. They also permit early evaluation since they can be tested in various ways, including traditional usability studies and informal user feedback, throughout the design process. *(Beaudouin-Lafon et al., 2000: p. 2)*

What Beaudouin-Lafon & Mackay are describing here is a stage that many design fields incorporate in their daily design process. Especially in the educational context, where new technologies are often experimented with, a prototype allows for a quick evaluation of those technologies and their potentials. A design process that uses prototyping methods in all stages, from sketching the first idea to reaching a final product, can also be called "a prototype driven design process". The advantage of a prototypical approach is that the designer needs to think in a more applied rather than abstract way. Thus, ideas become real artifacts instead of remaining conceptual theories that float about and are difficult to discuss with other stakeholders.

In its fundamentals, a prototype driven design process is a conscious, iterative design process *(Nielsen 1993)*. Starting with a vague idea and stopping when a satisfying result has been created, the designer builds a prototype for each iteration. Between the iterations, in the evaluation phase, the designer also reviews the used tools and methods and may change them for the next iteration. The artifacts

generated during these iterations can be used later to review and track the whole design process.

Building prototypes has the beauty of being very reactive. As the designer is forced to articulate his ideas, the design space gets explored in all crucial directions. The process of building the prototypes and the result itself show various alternative design paths, which can be further evaluated. Based on the built prototypes, the evaluation with the target group or space and the discussion with stakeholders is honest, focused and constructive. All these qualities empower the designer to easily compare ideas and implementations and thus decide on which steps to take next. In that sense, every designer should be encouraged to build as many prototypes as possible on their way to finding the ideal solution to their design problem.

Building a prototype should ideally consume only a fraction of the project's overall budget. However, prototypes have the reputation of being time consuming, especially when it comes to projects in which technology plays a primary role. If the cost-efficiency is too low or the risk of not being successful is too high, designers tend to fall back on more classic design methods. This problem has been identified and responded to with a new technique called "Rapid Prototyping", that enables designers to be faster, more efficient and take lower risks. Today we have an emerging industry that focuses on developing various prototyping tools for every kind of discipline. With the increasing amount of available tools and techniques, designers can explore more and more aspects of the prototype driven design process.

Looking back several years in the field of interaction design and physical computing we see that a prototype driven design process has become more and more a standard. This has mainly been enabled by tools like Processing, Arduino and 3D Printing, which abstract and simplify a technology or process that is usually

inaccessible for designers *(Banzi 2011)*. The development of such tools allows designers to extend their creativity and easily learn and leverage new technologies.

> Processing relates concepts of software to principles of visual form, motion, and interaction. It integrates a programming language, development environment, and teaching methodology into a unified system. Processing is created to teach fundamentals of computer programming within a visual context, to serve as a software sketchbook, and to be used as a production tool for specific contexts. It is used by students, artists, design professionals, and researchers for learning, prototyping, and execution. *(Reas et al. 2006)*

The Processing language by Casey Reas and Ben Fry is a phenomenal example for a technology-driven yet simple design tool. By wrapping the complex process of writing and compiling a java program into a simple graphical user interface and introducing a simple language for drawing on a canvas, designers that did not program before suddenly started to visualize their ideas through code. Furthermore, the tool allowed designers to go into fields like "Generative Design" *(McCormack et al. 2004)* that seemed quite impervious before.

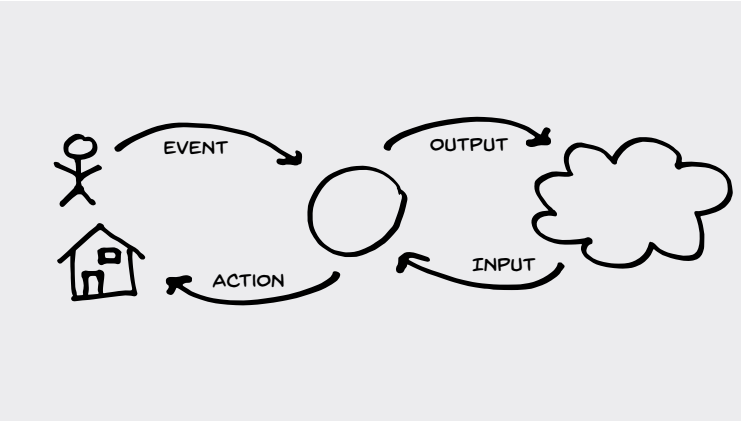
The same holds true for the Arduino project: "The Arduino philosophy is based on making designs rather than talking about them. It is a constant search for faster and more powerful ways to build better prototypes. We have explored many prototyping techniques and developed ways of thinking with our hands. *(Banzi 2011)*" Indeed, the Arduino project encouraged a considerable amount of designers to start playing with electronics. Especially the fairly recent field of interaction design benefited from that development, as students are now able to quickly create objects that can interact with the physical world.

Prototypes of Networked Objects

Networked objects are located at the border between the physical and digital space. They do not only interact with both spaces, but also link them together by bringing the digital into the physical and vice-versa. This implicates engineering tasks towards hardware in the physical space and towards software in the digital space. Therefore, a networked object always includes software development and the building of a physical object. On the one hand, the networked object is responsible for translating events that happen in the physical space to input for the information system in the digital space. On the other hand, the networked object also translates output by the information system to actions that are executed in the physical space. The distinction between event-action and input-output is important as it emphasizes the role of the networked object as a translator *(Fig. 6.1)* and encourages the designer to think about the two pairs in different ways.

Using the example from before, an event would be the mechanic trigger of the camera which signals the networked object to take a photo, gather all available data and finally send it to the central service. Pressing the button is the physical event that gets transformed to a digital output. From the perspective of the information system the picture is the input that gets processed and routed to picture frames that are subscribed to the feed. These networked objects receive the picture as an input and translate it to an action that will change the content of their physical display.

Technically, the networked object can be divided into two main components *(Fig. 6.2)*, the "event-action interface (EAI)" and the "input-output logic (IOL)". The EAI consists of several sensors, which can measure and detect events and actuators that can execute actions. The interface may vary from being very simple to being a



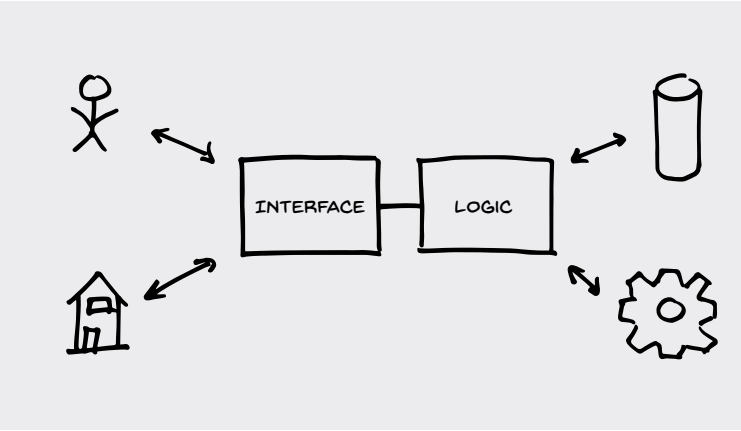


Fig. 6.1 - The networked object as a translator

Fig. 6.2 - The components of a networked object.

complex arrangement of many sensors and actuators. The IOL represents the networked object in the digital space. Events emitted by the EAI are managed and further forwarded by the IOL. The main role of the IOL is to make the raw and unfiltered EAI data available to other networked objects and information systems. This step of preprocessing is necessary to allow other networked objects to use the data in a more abstracted form, rather than dealing with input fluctuations or other technical details. The separation is straight forward as the EAI is the hardware needed to sense and act upon the environment, while the IOL is the software that deals with the communication with other networked objects and information systems.

The camera's EAI would be the photographic system itself, the GPS transmitters and the buttons used to take a photo and make fine adjustments. As not every little event or data is important to further processing, the IOL combines and simplifies the raw data to packets that are interpretable by other systems. In our example such a packet would be a picture with embedded meta informations. The picture frame on the other hand might be very simple as the received picture will be merely put on the display and the data does not need any special transformation.

The EAI is most likely an electronic circuit with sensors and actuators, whereas the IOL is a program that runs mainly on a computer or microprocessor. Depending on the project we might also have an information system that coordinates and stores information running on a personal computer or server. While all these parts can reside in different locations when building a prototype, a final product means that the EAI and IOL are combined together to a miniaturized printed-circuit-board and the information system is running in a high-available data-center.

Building the EAI is the easiest when using an electronics prototyping platform like Arduino. Numerous components and libraries

allow a designer to quickly sketch an electronic circuit and give him the possibility to interact with the user or environment. If the project is simple enough, the IOL can be programmed directly on the Arduino. If the project gains in complexity, the data can be forwarded to a computer where it gets processed by a higher order application written for example in Processing. Processing is a very good tool when it comes to developing a small, central information system that manages data coming in from multiple objects. In order to build bigger systems, web application frameworks like “Ruby on Rails” or platforms like “node.js” are recommended. However, detailed explanations about these tools go beyond the scope of this thesis.

Prototyping has become an almost standard stage in most design processes. Rapid prototyping tools allow designers to conveniently express and transfer ideas into a working physical representation. As designers we should make use of such rapid prototyping tools as quickly as possible, since the communities behind them provide a lot of stability and reduce the risk of a potential failure. In regards to building prototypes of networked objects, we now have multiple sensing and acting objects, an appropriate design strategy and maybe even an informations system that is ready to process data. However, there is one thing that remains unsolved: How can we network all these things together in an easy way?

The Missing Link

> Machine to Machine (M2M) refers to technologies that allow both wireless and wired systems to communicate with other devices of the same type. M2M is a broad term as it does not pinpoint specific wireless or wired networking, information and communications technology. This broad term is particularly used by business executives. M2M is considered an integral part of the Internet of Things (IoT) and brings several benefits to industry and business in general as it has a wide range of applications such as industrial automation, logistics, Smart Grid, Smart Cities, health, defense etc. mostly for monitoring but also for control purposes. *(http://en.wikipedia.org/wiki/Machine_to_machine)*

The industry picked up on the need for technologies to connect constrained and small devices with each other a long time ago. Research was mostly done in the field of "M2M" which, as described above, is broad and includes many other IoT related areas. The core assumption is based on the fact that an ubiquitous communication infrastructure, which is the basis of any IoT and Ubicomp concept, will firstly need low-cost radio modules to bring connection to any object and secondly, services that make sending data between any amount of these devices as easy as sending an E-Mail. As usual, scientists and the industry think big and aim at solving all problems at once by researching technologies such as "IPv6" and "6LoWPAN" [1] *(Minoli 2013)* that should bring ubiquitous

---

1 IPv6 is the successor of the IPv4 protocol that we use today to connect to the Internet. As the address space in IPv4 is limited and soon depleted, the IPv6 protocol will provide enough unique addresses for a long time. With 6LoWPAN, even devices that are very constrained and limited should have a unique address and be part of the Internet and thus the IoT.

connectivity to all devices in the future. However, the proliferation of those technologies depends heavily on telecommunication companies that are commissioned to build the Internet itself. Therefore, common usage of those technologies may still be postponed to several years or even decades from now. While those companies still dream big and look towards the real ubiquity of communication, we already have many application developers that want to build interconnected products today.

The market for applied M2M and IoT technologies has heavily grown in the past decade and now offers technologies for everyone from professionals building the next smart object to tinkerers that want to play with such future technology. Companies emerged that provide micro-controllers with integrated radio modules while software companies are building cloud infrastructures to connect millions of devices. What still remains undecided however, is the methodology of how these devices should actually communicate which each other — the protocols. They are the pinholes that control the transport of data from the little embeddable chip to the sophisticated software that runs in a data center. Without going into many details, one can state that the complexity in defining such protocols is that the developers need to assess the prospects of the architecture of future networks. This assessment is mostly based on individual assumptions and brings with it discussions such as whether sensors should be connected locally over some proprietary radio frequencies or connected to the Internet. Nevertheless, what we remember from the beginning of the Internet as the "protocol wars" *(http://www.computerhistory.org/revolution/networking/19/376*) is happening all over again for the IoT, but in an even more complex manor.

Although the market changes fast, things have become easier and more comprehensible in the past years. Recent developments in the prototyping and do-it-yourself (DIY) sector gave us hardware

platforms *(Fig. 7.1)* like the "Raspberry Pi[1]" , the "Arduino Yun[2]" and most recently the "ESP8266[3]". These platforms are all easy to program microprocessors with the built-in ability to connect to the Internet. Using an "Integrated Development Environment (IDE)", like the one from Arduino, the writing of programs for these processors is equally simple as writing a Processing sketch. To enable newcomers to start easily with these boards, the already strong but still growing community provides many tutorials, guides and screencasts that elaborate on the specifics. Along with the community came many shops like "Sparkfun", "Adafruit" and "Seeedstudio" that sell boards, components and other components you need to develop these kinds of projects. The simplicity of these products also caught the attention of professionals that are accustomed to complex and proprietary platforms. Fresh companies like "spark.io[4]" provide sim-

---

1 The Raspberry Pi is a low cost computer that can be used with standard peripherals like monitors, keyboards and alike as it runs Linux based operating systems. Its size as well as the ability to connect also other low-level components makes it a good tinkering and learning platform.

2 The Arduino Yun is the combination of an standard Arduino and a WiFi chip that runs a linux based operating system mostly found in router hardware. Through a simple interface every Arduino sketch can connect to the Internet and upload and download data.

3 The ESP8266 is a special WiFi module that has space for custom programs. It is a very small chip and does not have a lot of interfaces, but its cheapness and ability to be programmed with the standard Arduino IDE makes it a powerful component.

4 The company Spark offers hardware as well as cloud services that can be used to manage the deployment of many "sparks", upload a new firmware over the air or transmit data to the in-house aggregation services.

ilarly simple platforms that are targeted at professionals and enterprises that deal with way bigger amounts of devices than hobbyists.

A similar story can be told about softwares and services that simplify collecting and processing data from multiple devices. As professional services are not very useful to designers and tinkerers due to their complexity, they started to build platforms themselves that aim at the DIY and art sector. One of the first of its kind was "pachube" *(Fig. 7.2)* developed by Usman Haque, a know IoT and design researcher. The web service allowed everyone to connect multiple devices that stream sensor data to the platform, where it can be shared with others and even used to control actuators. The platform operated on the specially developed "Extended Environments Markup Language (EEML[1])". Pachube was a great success even if the EEML protocol was a little too expressive and therefore didn't get widely adopted. However, the community was growing and many people started to add their sensors and collect any kind of data. Before the platform was bought and relaunched under the name "Xively" the community used it heavily to log environmental data for example during the radiation crisis in Japan.

The acquisition of pachube left a gap in the toolset used by many around the world to explore the possibilities of the IoT. But soon other platforms launched and continued where pachube had left off, trying to provide similar and extended services. These alternatives

---

1 EEML supports installations, buildings, devices and events that collect environmental data and enables people to share this resource in realtime either within their own organizations or with the world as a whole via an internet connection or mobile network access. It can enable buildings to "talk", sharing remote environmental sensor data across the network in order to make local decisions based on wider, global perspectives. *(http://www.eeml.org)*

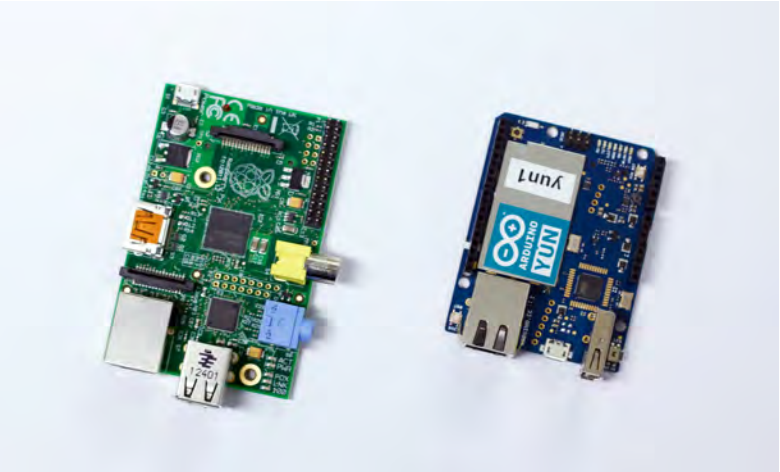
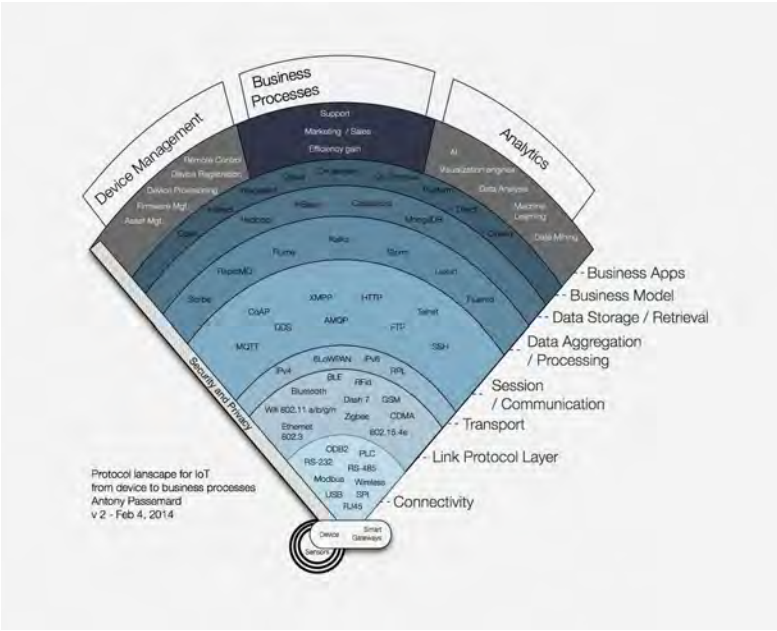


Fig. 7.1 - A Raspberry Pi and Arduino Yun

Fig. 7.2 - The pachube platform

Fig. 7.3 - The IoT Landscape


Business Processes
Device Management
Analytics
Support
Marketing / Sales
Efficiency gain
Remote Control
Device Registration
Device Provisioning
Firmware Mgt
Asset Mgt
AI
Visualization engines
Data Analysis
Machine Learning
Data Mining
Business Apps
Business Model
Data Storage / Retrieval
Data Aggregation / Processing
Session / Communication
Transport
Link Protocol Layer
Connectivity
Security and Privacy
Protocol lanscape for IoT
from device to business processes
Antony Passemard
v 2 - Feb 4, 2014

provide mostly proprietary "web API's[1] " that are hard to learn and hinder developers in switching to competitors easily. Looking again at the world of web application development, similar problems have been successfully solved in the past years. New services like "Heroku[2]" and "GitHub[3]" provide straight-forward platforms which can be appropriated in minutes and are based on standards that let developers switch between different providers. The ongoing competition results in better services for developers and allows them to be faster and more efficient. Cleary, this development has to happen also in the IoT field. The most critical part to this endeavor is finding a simple, widely adopted and easy-to-learn protocol that is open and does not show limits in any direction.

---

1 A server-side web API is a programmatic interface to a defined request-response message system, typically expressed in JSON or XML, which is exposed via the web — most commonly by means of an HTTP-based web server. *(http://en.wikipedia.org/wiki/Web_API)*

2 Heroku (heroku.com) is a platform as a service (PaaS) that allows developers to host their web applications by simply connecting their code repositories to the service. On every change in the repository, the platform automatically takes care of updating the running applications.

3 GitHub (github.com) is platform that allows developers to host their code repositories. Besides versioning every little change on the codebase, the platform provides complementary features that simplify the process of developing big applications. The openness of the platform has allowed it to become the number one open-source project hosting service in the world.

The Killer Protocol?

Building prototypes of networked objects includes sending data between any number of these objects and higher order applications running on a computer or server. As microprocessors like the Arduino have limited program space and memory, the protocol we are looking for needs to be very small and simple. More importantly we need to be able to send messages in real-time between the networked objects. For example, the effects of interacting with one object should be instantly visible on the other object. Lastly, the protocol should be able to allow transportation of any kind of data like images, text or numbers. This becomes important when we venture into the arts, where ideas may be more abstract and may not be covered by protocols like EEML that concentrates on applied use cases.

Looking at the IoT technology landscape *(Fig. 7.3)* today we see that there are many possibilities to transport data from the device up to the cloud (session communication). Unfortunately, most of these protocols are rather bloated and complex and are not suitable for the kind of experimentations we are looking for. Also HTTP, which is the de facto protocol of the Internet, lacks features that allows sending messages in real-time. However, there is one protocol called "Message Queue Telemetry Transport (MQTT)" that supports the previously stated requirements:

> MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small

> device scenarios. It is also ideal for mobile applications because of its small size, low power usage, minimized data packets, and efficient distribution of information to one or many receivers. *(http://mqtt.org)*

MQTT facilitates the publish-subscribe pattern (pub/sub) *(Fig. 7.4)* which is a well-known messaging pattern that allows sending messages between an arbitrary amount of clients. In order to send and receive messages, clients have to connect to a central broker that manages the flow of messages. After that, connected clients can publish messages that, for example, include current sensor readings, detected events or similar information. Each message has to be associated to a specific topic which can be made up by the designer much like a hashtag on twitter. Clients that are interested in certain messages can then subscribe to these topics and receive all subsequently published messages. The topic tree structure *(Fig. 7.5)* allows clients to communicate without needing to know who is receiving their messages. This "loose coupling" is very helpful in situations where components get developed independently and put together later in the process.

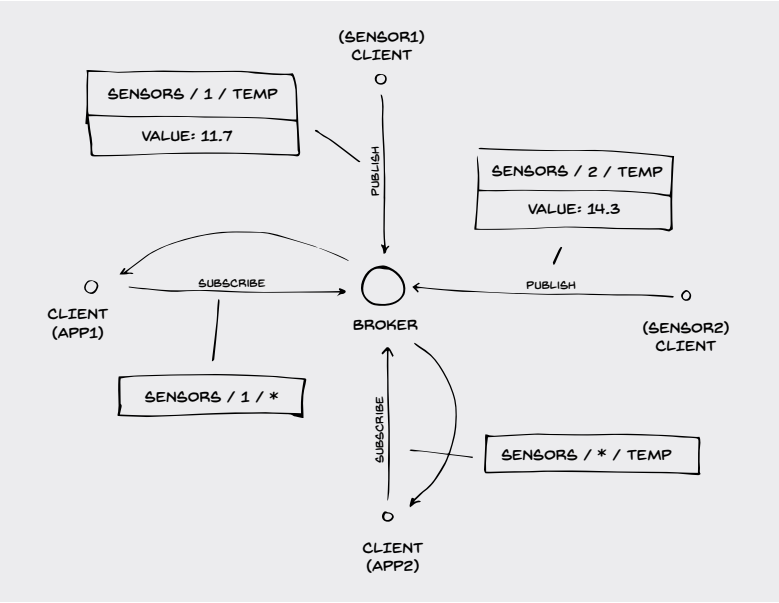
While the first version of MQTT already appeared in 1999, it did not make its breakthrough until 2013 when IBM submitted a final draft for standardization. In the meantime, many developers started to build client libraries that allow connections from any kind of software and hardware platform including Processing and Arduino. However, there is still one drawback that might hinder people from using it for their experiments and prototype development: Configuring and installing a broker needs a lot of resources and understanding of low-level system architecture. Also, renting a server and maintaining it is another obstacle for many out there who just want to play around a little bit. A service was needed that abstracts this tedious work and lets people simply use this technology in their projects, at best even for free.

The shiftr.io platform

> With shiftr.io the process of interconnecting objects, devices and apps becomes more accessible and less complex. Regardless of wether you are building an interactive installation, prototyping the next connected product or simply playing around with new technologies, shiftr.io lets you add connectivity to your project in an early stage. As a service, shiftr.io provides a rich publish/subscribe communication infrastructure, that is accessible through various protocols. A custom broker engine enables the built-in realtime graph that visualizes all events happening in your namespace. As a platform, shiftr.io provides you with the ability to share your data and access data of others. Sharing data publicly is encouraged by the platform's design. In the future, we plan to have additional features that allow more interactions between users and their namespaces. Using shiftr.io everyone is able to rapidly prototype connected objects and build a network of connected things. Start building prototypes for the Internet of Things now! *(https://shiftr.io)*

The shiftr.io platform *(Fig. 7.6)* basically provides an MQTT broker with the simplicity of a click, eliminating the need to install any software or maintain a server. It follows the concept of offering a service that provides the resource "connection" for any kind of project. Any developer or designer can register for a free account on the website and create an unlimited amount of "namespaces". Each of those namespaces is isolated, has its own topic tree and acts as a virtual broker. Therefore, the user does not need to think about hardware resources or costs of maintaining a server, but can simply use as much connection resource from the service as he needs. As of now, shiftr.io even provides these services for free.

As a platform, shiftr.io tries to extend the basic functionalities of a broker with tools that support the developer in his process of building a networked object. The real-time graph *(Fig. 7.7)* is the most



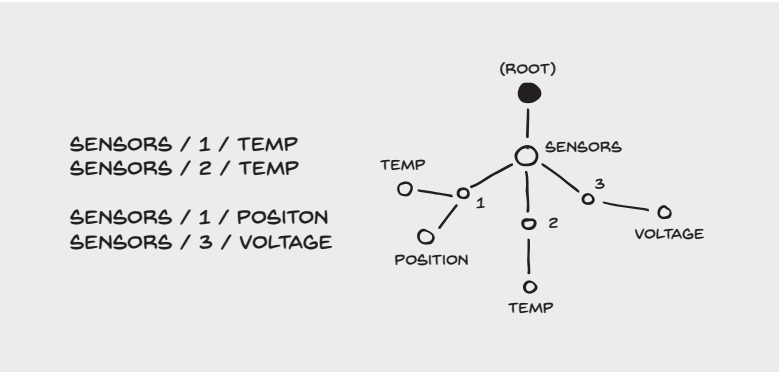


Fig. 7.4 The Publish-Subscribe Pattern

Fig. 7.5 Example Topic Tree

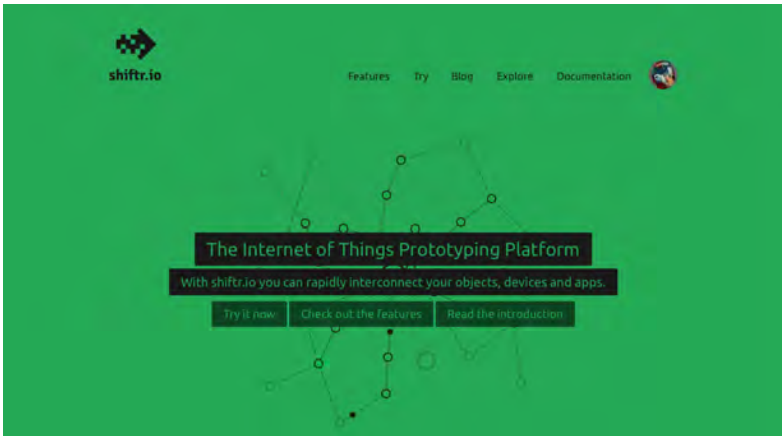


Fig. 7.6 - The shiftr.io Platform

Fig. 7.7 - The Real-Time Graph

Fig. 7.8 - The Chart Feature

central feature and allows for a live visualization of all data flowing in a namespace. The force-directed graph that is based on a physical model shows the topic tree structure, all current connections, their relations to the tree and messages flowing through the system. This feature is very important to newcomers that are learning to understand how the pub/sub system works as well as for professionals that need to debug or observe a running system. Yet the most important things in a communication system are the messages that get exchanged between devices. In most use cases we send numerical values, like the temperature reading of a sensor or the level of a knob on a control panel. To support working with that kind of data, the "chart" *(Fig. 7.8)* feature allows specially marked topics to be viewed as a line diagram that represents the numerical values published to the topic over time.

Having a broker as a services that includes custom-made additional features makes shiftr.io an important component of every IoT project. The MQTT protocol on which shiftr.io is based has been widely adopted in the past years, causing many developers to write libraries for various hardware and software platforms. The libraries for Arduino and Processing are simple and intuitive and can be added to a project within minutes. In that sense, a mix of Arduino, shiftr.io and Processing is a good starting point for prototyping networked objects.

After having defined the tools we need to build prototypes of networked objects quickly, we now have a way of networking them together. On top of this, the shiftr.io platform and the MQTT protocol bring the ability to build the components individually and later couple them together. This feature might become very important to building prototypes of networked objects as they tend to be technologically complex and not built by one designer herself. A design method that combines the theoretical model proposed before as well as the presented prototyping tools could support designers in finding an appropriate design process to build such networked objects. The method should help with exploring, planning and handling a design space and its vision.

Networked Exploration

Designing a networked object is not an easy task, as there are many design questions to solve and the object itself might be rather complex. In many cases, the designer might not even have a concrete idea in mind about how the particular object could work and look like. Luckily, the communities behind the prototyping tools and rapid prototyping techniques mentioned in the last chapter enable designers to use prototyping as a way to explore a design space by building objects and to learn more about their form and functionality while building them. Such a design process can be very powerful for design spaces that are hard to grasp fully and explore beforehand. As developing networked objects brings up many design challenges like privacy, latency, complexity and interusability a prototype-driven design process is very reactive to those influences *(Rowland et al. 2015)*.

The design method "Networked Exploration" supports the designer in developing his design space through multiple iterations into networked objects. The individual stages are not a strict framework, they are more of a recipe that can be taken as it is or adapted to the specific situation.

The design method introduces a design process *(Fig. 8.1)* where the designer is generally forced to focus on prototyping the individual components of a networked object, rather than obsessing over the final outcome. This eliminates the risk of stopping to discuss the meaningfulness of the still intangible experience, relieves the designer of thinking too far ahead in a way too early stage and lets him parallelize work better in a team. To help uncover the necessary components and thus work units, the design space is organized first using the "Event-Logic-Action Board (ELAB)" *(Fig. 8.2)*. The ELAB is inspired by the project management technique Kanban which helps organizing a team and structuring a development process

(Oza et al. 2013). Just like Kanban, the ELAB also uses cards that represent a unit of work, which in our case would be an event, logic or action component. Filling out these cards will help the team to formulate the individual components and to get a big picture of their goal. The board does not just help finding these components, moreover it should also aid throughout the process and serve as a central point where the team can agree on ideas and incrementally plan the project.

The designer starts with a defined design spaces that includes background research, mood-boards, videos and notes to a certain topic chosen by himself. From this material he defines multiple, different event, logic and action components that his networked object could be made of. By noting down each individual component onto a card, he creates the ELAB. After selecting a first set of events, logics and actions, the designer builds a prototype for each one of those components, which allows him to test them individually. After that, he will assemble at best two or more networked objects out of these working components. In the next step, the designer networks the prototypes together and complements the whole experience. The resulting system is then applied to the target environment and evaluated. The gained insights and learnings are merged with the design space and used to refine the concepts. Then, a new loop begins in which the designer repeats the steps above. The overall idea is to go through the whole loop as many times as possible and to master each stage as fast as possible. In later iterations more attention to details can be given to reach a higher quality.

The following sections explain the individual stages of the design method on the basis of the project Eden, a networked plant pot. The premise of this project was to create a plant pot that is capable of managing itself and can control water irrigation by reminding the owners if conditions get too bad. It was important to find natural and embedded alternatives to traditional sensor-smartphone

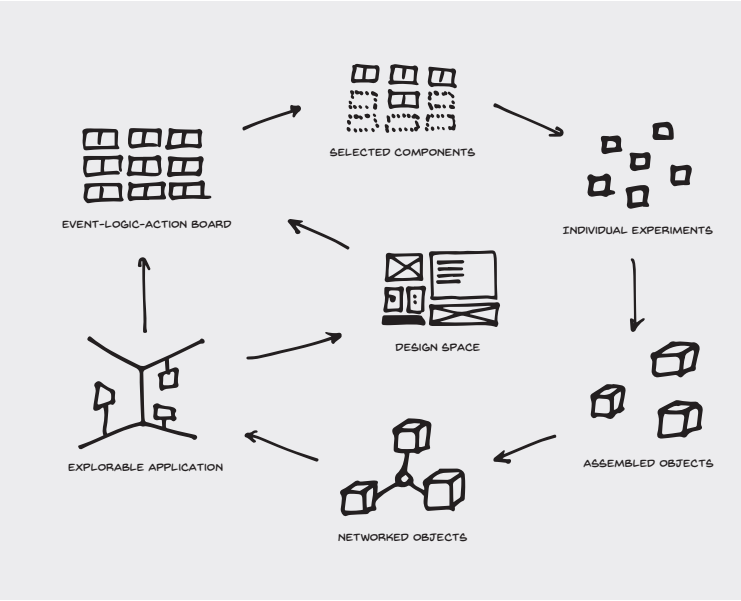
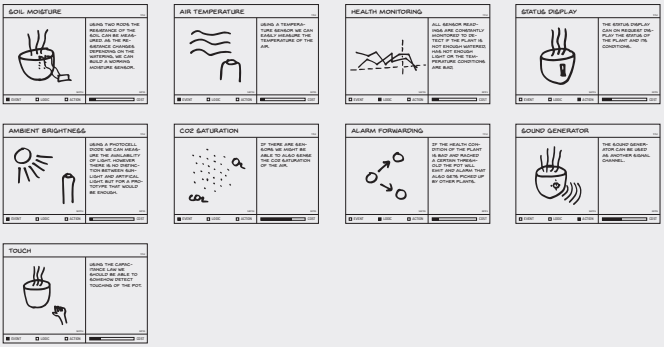


Fig. 8.1 – The Individual Stages of the Design Method



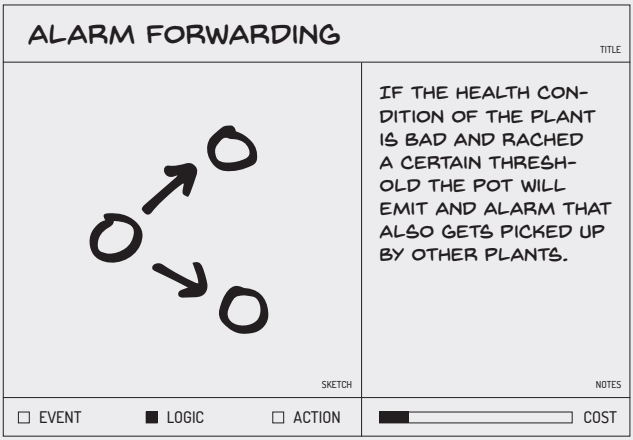


Fig. 8.2 - The ELAB from the Eden Project

Fig. 8.3 - A single ELAB Card from the Eden Project

combinations that are available today. Finally, by networking the pots together, they should act as a system and support each other when for example a plant has been forgotten and its conditions are bad. In collaboration with inhabitants, the product can revegetate big modern buildings that lack green and have no central resources to cultivate such plants throughout the building.

1. Analysis of the Design Space and Mapping of Components Using the ELAB

In many cases designers start by defining a design space that includes all necessary information and material around the topic or field they would like to investigate. This process is inevitable and should be executed carefully to create a stable groundwork for the project to grow on. Now, as a first stage of the design method, the designer can analyze his design space and assemble the ELAB. When defining the individual event, logic and action components he should not think about feasibility or time constrains but simply focus on finding all possible components his objects could be made of. Apart from formulating the ideas, the designer can already roughly estimate the costs of these features by filling out cost bar at the bottom-right of the card *(Fig. 8.3)*.

The ELAB *(Fig. 8.2)* of the Eden project is very conservative and includes components with a low cost due to the project's one week time frame. Obviously, this project has an emphasis on events as observing the plants' condition is a key element in the project. From the definition of the design space it was already clear that a visual way for displaying the plants' conditions was intended, however the ELAB also includes an alternative auditory action that could be investigated optionally.

2. Selection of the First Set of Components From the ELAB

By selecting a set of components from the ELAB the designer starts one iteration of the process. Depending on the cost of the individual components and the available time frame, a designer may chose just a couple or several components to start with. It is recommended however to select at least one event and action so that the networked object is interactive and better testable later on in the process.

For the first iteration of the Eden plant pot the "moisture sensor", "touch sensing", "health monitoring" and "status display" component was chosen with a rough schedule of one and a half days.

3. Experimenting and Prototyping of the Individual Components

After selecting components, the designer can focus on these individually and build first prototypes. Events and actions can be built easily with tools like Arduino and some electronics, whereas basic logic components can be developed using Processing or similar frameworks. To get a first feeling for the resulting interactivity it is recommended to already use a tool like shiftr.io to network the experiments together. That way the designer can improve them hands-on without having to fake any features. The lose-coupling approach also helps with parallelizing the tasks and putting the things together easily.

The moisture sensor *(Fig. 8.4)* for the Eden plant pot was developed after a tutorial found on the Internet which simply uses two metal nails that measure the resistance of the soil between the two rods. The circuit was complemented with an Arduino Yun that uploads the values every second to a shiftr.io namespace. Using the

real-time visualization features, the values can be reviewed online. The pre-produced display ring *(Fig. 8.5)* was then connected to another Arduino Yun, taking care of displaying the right values received from shiftr.io. This simple setup made it possible to explore these basic features very quickly without thinking too far ahead.

4. Assembling Multiple Networked Objects from the Individual Component Prototypes

Once the designer has completed the individual component prototypes he can begin with assembling the networked objects. In this step, the individual functionalities of the components have to be merged into one object. Technically, this means that electronic circuits and software that were isolated before, have to now be put together to one piece. As the individual components are fully functional by now this step should be simple. Also, the designer might need to replicate the components to have multiple networked objects ready for the next stage.

In the beginning, the individual components for the Eden plant pot all used their own Arduino Yun. In order to spare resources and create more compact controllers, the electronics and code got merged into one codebase and circuit per pot. The setup was also replicated two times to have three fully working pots *(Fig. 8.6)*.

5. Final Networking of the Networked Objects

In this final step of the iteration's development part, the now assembled networked objects get networked together. This means that from this point on, data should be shared between the objects automatically and without any help or manual intervention. Of course, there might still be an application in the background that "animates" the objects to give the feeling that they are already connected to an information system. Yet, the goal is to have objects that fully work

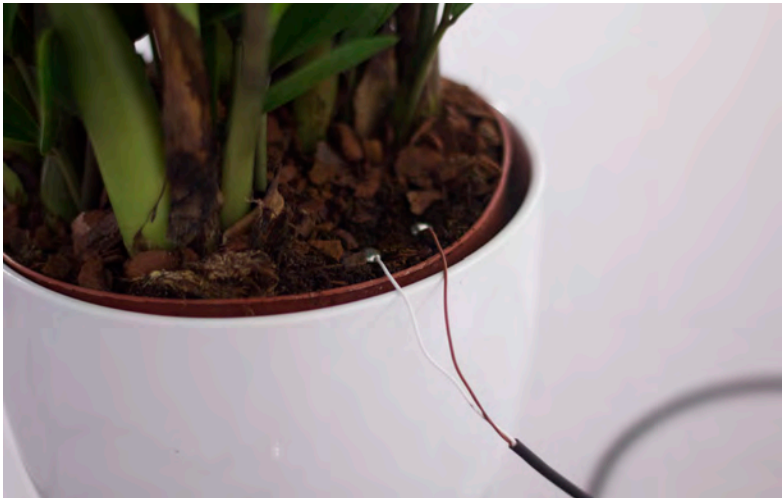
so that the designer can test them without interruptions in the next stage.

As the Eden plant pot didn't have any networking-dependent features up until now, another component was added in this step. The "alarm forwarding" logic component would allow the pot to emit an alarm as soon as its condition get bad. Other pots would pick up this alarm and visualize it through their displays. As this component is totally software related it was added without any further changes to the objects. Because of the project's small, manageable scale, all the logic got implemented directly on the Arduinos without having any centralized information system.

6. Evaluation of the Networked Objects in Their Target Space

At this stage the designer has a first set of working networked objects. For a proper evaluation of the experience, it is important that the objects get applied to their target space. In its best, the experience should be tested by people that are uninvolved in the project and can give unbiased feedback. Also, a designer may consider testing the experience over a longer period of time instead of making just short test sessions. It is highly recommended to "harvest" the testers for further ideas as they often raise interesting issues concerning the perception and the usability of the object.

As the Eden pots where intended to inhabit a school building, the classroom in which they were developed was at the same time also the application space. Their dependency on a special Wi-Fi connection prevented a more widespread application at that time. However, the pots did get separated and placed on other students' desks. The most obvious user feedback was related to the visible technology in form of the Arduino and wires that made the pot look fragile. The controlling of the pot through the touch sensor and the display ring



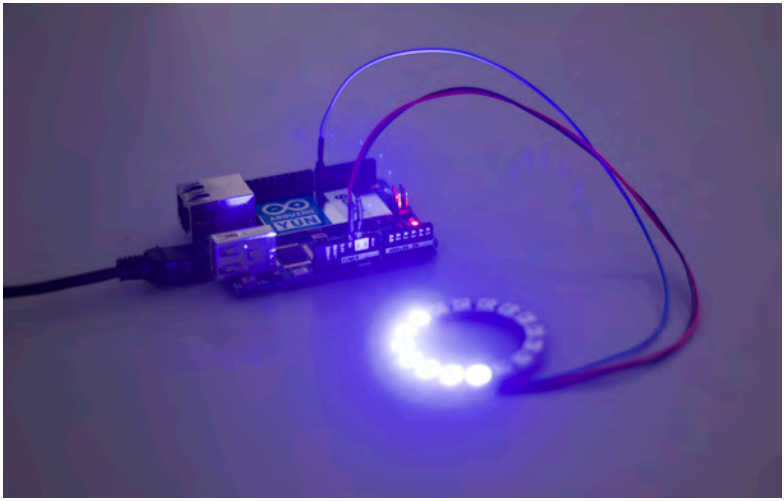


Fig. 8.4 - The Eden Moisture Sensor
Fig. 8.5 - The Eden Display Ring
Fig. 8.6 - The Finshed Eden Pots

was understandable, but did not seem to be important to the user. The alarming feature however was intriguing to many: A watering of the plant could be observed within minutes when an alarm was emitted. The "alarm forwarding" also caused students to kindly remind other students to water their plants.

7. Reflection on the Iteration, Networked Objects and Design Space

In the last stage of the iteration the designer reflects on the previous evaluation and the networked objects as the result of the development process. The created experience should be compared to the desired goal of the project to critically review what is already there and identify things that are still missing. All the reflection work should be channeled into changes to the ELAB in form of adding new components or also removing things that failed or do not make sense anymore. Also, the review might create questions and ideas that flow into the design space and need to be further explored and researched. With finishing this step the designer either is happy with his result or starts over and begins the next iteration.

The feedback of the evaluation phase showed that the Eden project is on the right track. Of course, the custom made sensors needed to be more precise and the system more sensitive as to when to emit an alarm. In addition to the soil moisture sensors the pot should also have also sensors further down in the earth to protect it against overwatering – a problem that has arisen due to the alert system. And lastly, the technology of the pots needs to be more hidden and subtle so as to not put it in the foreground and instead retain the plants' organic feel.

Conclusion

From the beginning of this thesis project, the goal was to create a framework for designers to allow them to investigate the IoT, explore possibilities of networked objects and create new experiences that involve interconnection. With establishing the concept of a networked object that is on the border of the physical and digital space, designers have a point of reference from which they can start their own investigations. This concept and especially the design guidelines described in chapter five are the beginning of creating a foundation for designers that can be extended and built upon to explore this field even further. The tool shiftr.io plays a central role in abstracting the technology and making it available in a simple and usable form. It helps the designer to take ideas from their rough concept to a working prototype of networked objects in a very short time. In conjunction with other rapid prototyping tools, designers are now equipped with a set of tool that allows them to join the discussion about near future scenario involving the IoT. To allow an even easier integration of the theoretical concept and the practical tool in a design process, the design method Networked Exploration helps to structure a design process that includes both theory and hands-on work. Furthermore, this method can be used to explore a design space without having a clear idea of the outcome but the will to find interesting new experiences.

During the one and a half years of my master's project, the platform shiftr.io did not remain unnoticed: After its launch, shiftr.io was received enthusiastically by a small community of IoT hobbyists. Although the impact wasn't huge, these users actively tested and used the platform, giving me helpful feedback for further implementations. Shiftr.io was also integrated in a course at the ZHdK with bachelor students around the topic Embodied Interaction. Based on their encounter with this technology, three bachelor students have

chosen to work with the tool in their diploma project. One of the students is even using it to develop networked objects, while the others profit from the ability of shiftr.io to connect things and for example send data from an Arduino to a website. As the philosophy of shiftr.io is to bring connectivity to any kind of undertaking, these projects and users show how the platform fulfills its purpose and lives up to this idea. I will definitely continue working on the platform, develop it further and show other interested people how easy connecting things could be. As a first step I decided to launch the "showcase.shiftr.io" page that will present projects that use that platform. The first videos will be about the above-mentioned course and the bachelor projects.

Compared to shiftr.io, the conceptual framework and the design method have not yet been extensively tested. This is why I see my master's thesis as the foundation for further investigations, experiments and projects in the realm of prototyping networked objects. There are many areas in which this can be done: On the one hand there is the educational context, where the theory can be transformed into exercises and design methods that will shape the thinking of future designers. On the other hand, the arguments in this thesis could be a starting point for further research projects that take details from the framework and work out ways of thinking about and implementing them. Future steps from my side include a website called "networkedobjects.info" that firstly, establishes the philosophy behind "Networked Objects" and secondly, serves as a harbor for people who are interested in collaborating and researching in this field. While my approach to the topic has been influenced strongly by technology, I believe that it is now possible to continue the research from a more design theoretical or artistic point of view.

In any case, I hope that my efforts will contribute to the shared idea of an interconnected future. And that one day, this idea will spark a revolution that will free us from the screens and bring back the objects.

Weiser, M. (1991). The Computer in the 21st Century. Scientific American.

Dourish, P., & Bell, G. (2011). Divining a Digital Future. MIT Press.

Evans, D. (2011). The Internet of Things, 1–11.

Clark, A. (2009). Dispersed selves. Leonardo Electronic Almanac, 16(4), 1–7. Retrieved from http://www.leoalmanac.org/wp-content/uploads/2012/09/02_clark.pdf

Kortuem, G., Kawsar, F., Fitton, D., & Sundramoorthy, V. (2010). Smart objects as building blocks for the Internet of things. Internet Computing, IEEE, 14(1), 44–51. doi:10.1109/MIC.2009.143

Ishii, H. (2008). Tangible bits: beyond pixels. Tangible and Embedded Interaction 2008, xv–xxv. doi:10.1145/1347390.1347392

Kranz, M., Holleis, P., & Schmidt, A. (2010). Embedded Interaction.

Vitsœ | Good design. (n.d.). Vitsœ | Good design. Vitsoe.com. Retrieved March 6, 2015, from https://www.vitsoe.com/gb/about/good-design

Dubberly, H., Pangaro, P., & Haque, U. (2009). On Modeling - What is interaction?: are there different types? Interactions, 16(1), 69–75. doi:10.1145/1456202.1456220

Aether Cone | The Thinking Music Player. (n.d.). Retrieved April 20, 2015, from http://www.aether.com

Beaudouin-Lafon, M., & Mackay, W. (2000). Prototyping Tools and Techniques.

Nielsen, J. (1993). Iterative User Interface Design. IEEE Computer, 26, 32–41. Retrieved from http://www.useit.com/papers/iterative_design

Banzi, M. (2011). Getting Started with Arduino. O'Reilly Media.

Reas, C., & Fry, B. (2006). Processing: programming for the media arts. Ai & Society, 20(4), 526–538. doi:10.1007/s00146-006-0050-9

McCormack, J., Dorin, A., & Innocent, T. (2004). Generative Design: A Paradigm for Design Research.

Machine to machine. (n.d.). Machine to machine. Retrieved April 3, 2015, from http://en.wikipedia.org/wiki/Machine_to_machine

Minoli, D. (2013). Building the Internet of Things with IPv6 and MIPv6. Wiley.

Protocol Wars - CHM Revolution. (n.d.). Retrieved April 13, 2015, from http://www.computerhistory.org/revolution/networking/19/376

Haque Design Research Ltd. (n.d.). Extended Environments Markup Language: EEML. Retrieved April 6, 2015, from http://www.eeml.org

MQTT. (n.d.). Retrieved April 6, 2015, from http://mqtt.org/

Rowland, C., Elisabeth, G., Martin, C., Alfred, L., & Ann, L. (2015). Designing Connected Products. O'Reilly Media.

Oza, N., Fagerholm, F., & Münch, J. (2013). How Does Kanban Impact Communication and Collaboration in Software Engineering Teams? CoRR Abs/1311.1323, 125–128. doi:10.1109/CHASE.2013.6614747

Fig. 3.1, Retrieved March 2, 2015,
from http://www2.meethue.com/en-au/what-is-hue/get-started

Fig. 3.2, Retrieved October 11, 2014,
from https://nest.com/press/#product-images

Fig. 3.3, Retrieved March 2, 2015,
from https://fresh.amazon.com/dash

Fig. 3.4, Retrieved April 6, 2015,
from http://www.myrocki.com/mediakit

Fig. 3.5, Retrieved March 11, 2015,
from http://en.wikipedia.org/wiki/Amazon_Echo

Fig. 6.1, Retrieved April 23, 2015,
from https://edyn.com

Fig. 6.2, Retrieved April 23, 2015,
from http://www.aether.com

Fig. 6.3, Retrieved April 20, 2015,
from http://www.dropletlife.com

Fig. 6.4, Retrieved April 23, 2015,
from https://sen.se/about/press

Fig. 7.2, Retrieved April 6, 2015,
from http://www.envirolyser.com/2011/01/30/the-internet-of-things

Fig. 7.3, Retrieved April 7, 2015,
from https://entrepreneurshiptalk.wordpress.com/2014/01/29/
the-internet-of-thing-protocol-stack-from-sensors-to-business-value