

Designing a Tangible Programming Interface for the Internet of Things

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Daniel Dudo

Matrikelnummer 0925446

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung

Betreuer: Ao.Univ.Prof. Dr. Peter Purgathofer

Mitbetreuer: Univ.Ass. Dr. Florian Güldenpfennig

Wien, 22.01.2018

(Unterschrift Verfasser/in)

(Unterschrift Betreuer/in)

Designing a Tangible Programming Interface for the Internet of Things

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Media Informatics

by

Daniel Dudo

Registration Number 0925446

to the
Faculty of Informatics at the Vienna University of Technology

Supervision

Advisor: Ao.Univ.Prof. Dr. Peter Purgathofer

Co-Advisor: Univ.Ass. Dr. Florian Güldenpfennig

Vienna, 22.01.2018

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Daniel Dudo
Wienerberg City, 1100 Wien

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.“

Wien, 22.01.2018

(Unterschrift Verfasser)

Acknowledgements

First of all, I would like to thank Dr. Florian Güldenpfennig for his great and invaluable support during my work, for his motivation and patience, which have energetically supported me and which allowed me to finish this work. I learned a lot from his academic experiences, and above all, he showed me how to deal with scientific problems - but further, I also was allowed to enjoy some of his culinary delights (tagliatelle with salmon and soy sauce!). Special thanks also to Prof. Peter Purgathofer, who was always available for inspiration and advice, and also for his great guidance on 3D printing and modeling. Furthermore, I would like to thank all participants who have given me valuable feedback which allowed me to push this work forward. I would also like to express my gratitude to my family and to all my friends. I sincerely thank them, because they always stood by my side, helped me to complete this work, and above all enriched my life.

Abstract

The Internet of Things (IoT) aims at making our lives more livable, easier, and more enjoyable. However, the IoT is also facing challenges of how to configure and program applications or of how to participate in the IoT.

The aim of this thesis was to explore how people experience alternative approaches to programming home automation technology (commonly called smart home) and related IoT applications by using tangible computing devices. Following a *Research through Design* approach, this thesis employed an iterative and user-centered design process that focused on creating a series of prototypes based on the feedback of 43 participants. Finally three prototypes were built where the first prototype was utilized to introduce the primary interaction concept to the participants; the other two prototypes were based on subsequent feedback from the participants and considering their suggestions. For the final prototype, several sensor and actuator modules were developed, which could be linked with the help of a tangible controller device. In addition, functions such as time duration, delay or inverting could also be set in the prototype system. In a final evaluation, this prototype was examined with technically experienced prospective students of informatics and technically unexperienced older participants. The results showed that an interesting approach to programming IoT devices was developed, which was also understood and appreciated by older participants. Hence, the design exploration showed potential in empowering end-users in solving everyday challenges with smart technology.

Keywords: Internet of Things, IoT, ubicomp, research through design, interaction, empower, prototyping, sensor, actuator, Arduino

Kurzfassung

Einerseits versucht das Internet der Dinge (IoT) unser Leben lebenswerter und angenehmer zu gestalten, aber andererseits steht das IoT vor Herausforderungen bei der Konfiguration, Programmierung oder sogar beim Einsatz des IoT.

Ziel dieser Arbeit ist es zu erforschen, wie Menschen einen alternativen Ansatz zur Programmierung von Smart Home Technologie durch den Einsatz von greifbaren Computergeräten erfahren und weiters festzustellen, ob dieser Ansatz die Endnutzer und Endnutzerinnen in die Lage versetzt, kleine alltägliche Herausforderungen zu lösen. Basierend auf einem Research through Design Ansatz folgt diese Arbeit einem iterativen Designprozess, in welchem eine Reihe von Prototypen hergestellt und welche mit insgesamt 43 Teilnehmern und Teilnehmerinnen evaluiert wurden um Feedback zu sammeln. Schlussendlich wurden drei Prototypen erstellt, bei denen der erste Prototyp das Konzept dieser Arbeit vorstellte und die anderen beiden Prototypen auf darauf folgenden Feedback der Teilnehmer und Teilnehmerinnen basiert. Für den endgültigen Prototyp wurden mehrere Sensor- und Aktormodule entwickelt, die mit Hilfe eines selbstentwickelten Controllers verknüpft werden konnten. Zusätzlich konnten Funktionen wie Zeitdauer, Verzögerung oder Invertierung verwendet werden. Mit einer abschließenden Evaluierung wurde der Prototyp mit technisch versierten Studieninteressierten-Informatikern und -Informatikerinnen und technisch unerfahrenen älteren Menschen untersucht. Die Ergebnisse zeigen, dass ein interessanter und greifbarer Ansatz für die Programmierung von IoT-Geräten entwickelt wurde, der auch von älteren Menschen gut aufgenommen und verstanden wurde.

Everything is a button

Workshop Participant, 2016

Table of Contents

1	Introduction.....	1
1.1	<i>Research Scope</i>	4
1.2	<i>Goals and Research Questions</i>	4
1.2.1	<i>Limitations.....</i>	5
1.3	<i>Contributions of the Thesis.....</i>	6
1.4	<i>Structure of the Thesis.....</i>	6
2	Related Work	7
2.1	<i>Defining the Internet of Things (IoT)</i>	7
2.2	<i>Defining the Term Programming with Respect to this Thesis.....</i>	8
2.3	<i>Commercial IoT Products</i>	9
2.4	<i>IoT Research Products used in the Academia and for Education</i>	11
2.5	<i>End User Development in the IoT and of Tangibles</i>	14
2.6	<i>Interaction Types in Programming the IoT.....</i>	18
2.7	<i>Summary of Related Work.....</i>	20
3	Approach and Methods.....	23
3.1	<i>Approach: Design Research in HCI</i>	24
3.1.1	<i>Research through Design</i>	26
3.2	<i>Methods for Prototyping</i>	28
3.2.1	<i>Explorative Design Methods</i>	28
3.2.2	<i>3D Printing and Rapid Prototyping.....</i>	30
3.2.3	<i>Electronics Prototyping Tools.....</i>	31
3.3	<i>User Research Methods.....</i>	32
3.4	<i>Research Sequence.....</i>	33
3.5	<i>Participants</i>	35
3.6	<i>Summary</i>	36
4	Iterative Design Process.....	37
4.1	<i>Overview.....</i>	37
4.2	<i>Previous Work.....</i>	39
4.3	<i>Design Workshop</i>	41
4.4	<i>First Prototype: Lo-fi and Made from Wood</i>	45
4.4.1	<i>Prototype Concept</i>	45
4.4.2	<i>Prototyping Process and Functions</i>	46
4.4.3	<i>Evaluation and Insights.....</i>	49
4.5	<i>Second Prototype: Mid-Fi Prototype with Interactivity.....</i>	53
4.5.1	<i>Prototype Development</i>	53
4.5.2	<i>Evaluation and Insights.....</i>	60
4.6	<i>Final Prototype</i>	63
4.6.1	<i>Prototype Development</i>	63
4.6.2	<i>Implemented Features.....</i>	68
4.6.3	<i>Troubleshooting</i>	69
4.6.4	<i>Evaluation and Insights.....</i>	70
5	Findings.....	71
5.1	<i>General Procedure.....</i>	71

5.2	<i>Evaluation Phase I</i>	72
5.3	<i>Evaluation Phase II</i>	80
5.4	<i>Comparison of Phase I and Phase II</i>	86
6	Discussion	90
6.1	<i>Brief Summary</i>	90
6.2	<i>Reflections on the Findings</i>	91
6.3	<i>Comparison with the Previous Work</i>	94
6.4	<i>Prevalent Paradigms in Supporting End-Users to Program the IoT</i>	95
6.5	<i>Boundaries of Tangible Programming and Computing</i>	96
6.6	<i>Final Reflections on the Importance of Evaluations (User Feedback)</i>	98
6.7	<i>Future Directions</i>	98
6.8	<i>Conclusion</i>	100
	List of Figures	101
	List of Tables	103
	Acronyms	104
	Appendix	105
	<i>Design Workshop Appendix</i>	105
	<i>Findings Appendix</i>	108
	<i>Models Appendix</i>	113
	<i>Constants Appendix</i>	116
	Bibliography	118

1 Introduction

The Desktop computer, which we commonly know as an object in the workplace, continues to lose its supremacy among computing technology for personal use. Since about a decade, computing gradually shifted from the workstation to the mobile Internet. The fact that technology will become cheaper and especially smaller means that computers will be more and more ubiquitous. This conclusion has already been drawn by Mark Weiser in the 90s, coining the notion of “ubiquitous computing” (ubiquitous computing) in his paper “The Computer for the 21st Century” (Weiser 1991). Weiser’s vision of “calm computing” (Weiser and Brown 1997) and technology that is embedded everywhere increasingly comes into being, enabled by novel technologies. A manifestation for this can be seen in the recent popularity of the Internet of Things (IoT) and related applications (Walport 2014). Cisco’s Internet Business Solutions Group (IBSG) defined the “birth of IoT” between 2008 and 2009, responding to the growing number of “things” connected to the Internet. Already before 2010, there were more connected devices on the planet than living people and the trend is growing year by year as for example published by Cisco (Evans 2012). According to the Figure 1.1, we currently have over 30 billion devices connected to the Internet with an estimated world population of 7.4 billion. Approximately every 5 years, the number of connected devices is doubled. Hence, Cisco estimates that by 2020, more than 50 billion devices will be connected and the IoT is predicted to have an economic impact of over \$14 trillion (Lee and Lee 2015).

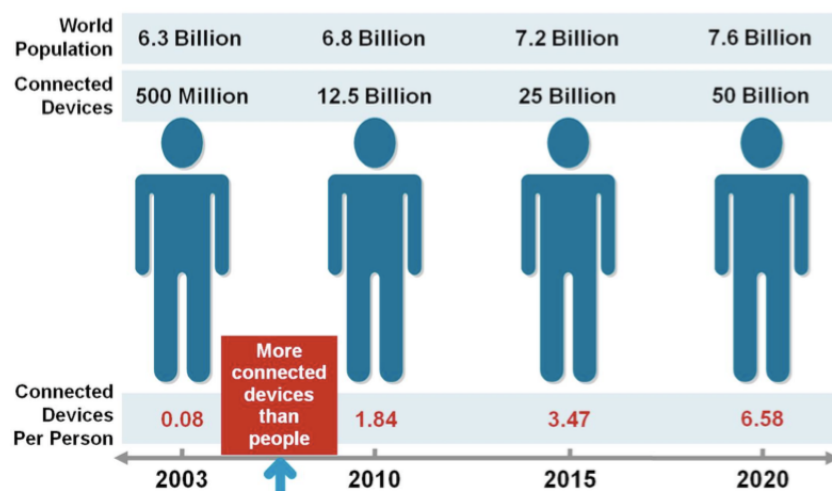


Figure 1.1 A graphic of Cisco IBSG (2011) which shows the number of connected devices in relation to the world population. In 2010 there were over 12.5 billion devices connected to the Internet, while the world population was about 6.8 billion. Hence, there have already been more than one connected devices per person since about 2008.

This massive increase in the number of connected devices per person is also a result of the large sales of mobile devices such as smartphones or tablets. However, these

figures should soon be further enhanced by the sensing and acting devices, which are soon installed “everywhere”, especially in the industrialized countries. Therefore the paradigmatic major drivers to the ubiquity of connected (smart) devices will be “wearable computing devices (smart watches, glasses, etc.), smart-cities, smart metering devices deployed by energy suppliers to analyses consumption at the home level, self-driving vehicles or sensor networks.” (Vaquero and Rodero-Merino 2014, p.1) Also, Google Trends (Google 2006) - an online service by Google, which provides visualizations of how often a search term has been entered by users in the Google search engine - shows a growing interest in IoT topics starting around 2009.

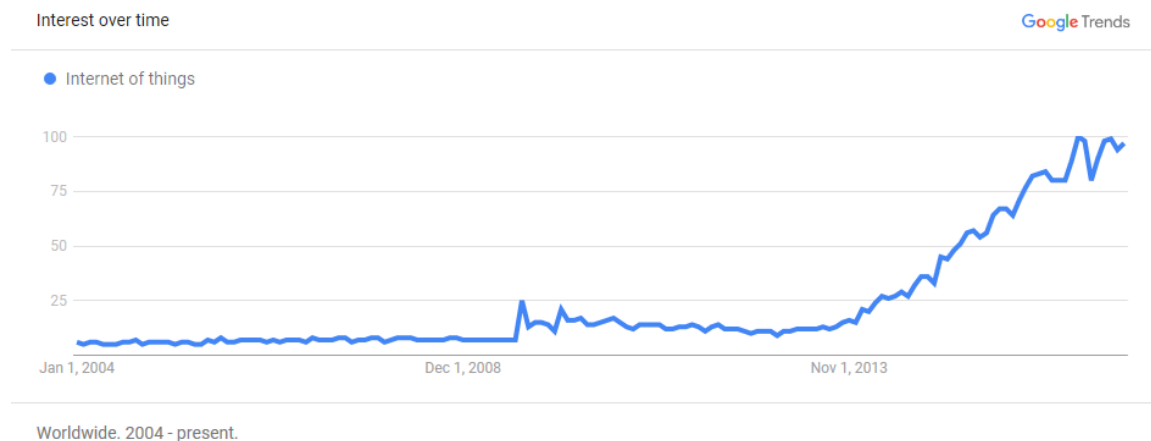


Figure 1.2 Worldwide interest over time on Google Trends for Internet of Things from 2004 to May 4, 2017 – Note: “Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. Likewise a score of 0 means the term was less than 1% as popular as the peak.” (Google 2006)

Many countries have already invested in Internet of Things initiatives, for example, the U.K. government has funded £5 million for projects on IoT developments, the IoT European Research Cluster (IERC) FP7 launched its own IoT project pool and created an international IoT forum for the use of IoT in Europe, and in the U.S. IBM reported in 2009 that “IoT can be an effective way to improve traditional physical and information technology infrastructure, and will have a greater positive impact on productivity and innovation”(Da Xu, He et al. 2014, p.2).

Targeted in particular at the Internet of Things, the Bluetooth Special Interest Group (SIG) Bluetooth (Bluetooth 1994) has designed the new Bluetooth Standard 5 for the Internet of Things and is thus not only aimed at mobile devices like phones or tablets (Bluetooth is currently used mainly for smartphones, headphones or car radios). This Bluetooth SIG was founded in the 1990s and includes companies such as Ericsson, Nokia, Intel and IBM. Together they wanted to establish an “industry standard”, which can be used to exchange data wirelessly, securely and with low costs. The new Standard 5 will not only present a power-saving variant (called Bluetooth Low Energy (BLE)), but also support the Mesh protocol (SIG 2017) to significantly increase the range (currently about 10m). With the mesh protocol, each component can be connected to one or more others. This allows a component to act as a “repeater”, with the component receiving data and sending it to another component. For the development of IoT applications there has not been a uniform standard, so manufacturers are developing their “own” standard without agreeing with other parties. SIG hopes that the products

of various manufacturers will be compatible with each other and thus allows a rapid development. (Hussain, Mehnaz et al. 2017)

Every new technology like IoT brings advantages but also comes with risks and challenges, because it can potentially have a huge impact on our daily lives for the better or, unfortunately, for the worse. On the one hand, the UK Government has recognized the coming of a “new industrial revolution” (Walport 2014), but on the other hand, there is no specific education in school to face this new revolution, and children do not learn about novel technology like IoT in the classroom (Lechelt, Rogers et al. 2016). Still, there are a lot of commercial and academic projects to bring computational thinking to children in their spare time like, for example, Google’s Bloks Project (Blikstein, Sipitakiat et al. 2016), where their idea is that children can use physical blocks to “write” a program by manipulating tangible blocks. Topobo (Raffle, Parkes et al. 2004), Cubelets (Schweikardt 2011), and LittleBits (Bdeir 2009) are related products that allow the user to program applications by assembling physical blocks with embedded computational power (i.e., the blocks can “sense” each other and have different functions). However, this kind of physical programming is restricted with respect to functionality and usage. For example, LittleBits and Project Bloks is restricted by its linearity, that is, the resulting applications feature a rather simple chain of actions and reactions. Nevertheless, Projects like littleBits or Project Bloks are good examples to motivate the teachers to use IoT technology at school so that children can experience their first experiences with IoT.

Besides educational IoT programming kits as described above, there is a growing market for consumer electronic products in the context of the smart home sector. Here, the predominant solution is to provide a visual programming environment to define links between sensor and actuator modules (respectively the physical blocks). The users are often required to map a problem in the “real world” (like ringing a door bell when someone enters the room) to a digital user interface. Specially, for Home Automation Technologies there is very often an app (an application for smartphones or tablet computers) to configure and control the system. This is reflected by a phenomenon described as the “Smartphone Reflex”, the expectation that there is an app for every occurring problem (Dautriche, Lenoir et al. 2013). Hence, in many situations a computer or a smartphone is required and some knowledge of software. Depending on the number of different devices and IoT brands, this can pose a challenge to the users, as every manufacturer comes with its own interface. Hence, programming the IoT and related applications can be hard for users in general and not only for children as described in the paragraph above. Especially, elderly people (an important target group for smart homes and Ambient Assisted Living) can experience difficulties in operating technology that is new to them like smartphone apps (Güldenpfennig, Nunes et al. 2016).

In summary, reflecting on the great number of educational IoT learning-kits and commercial products, there seems to be a pressing need for computer-skills or IoT related skills (Kim, Gajos et al. 2016, Tzeremes 2016). On the one hand, it is important to avoid typical frustrations with new technology, on the other hand, enabling people to

customize IoT systems on their own might be a promising opportunity for living a more independent and entitled life. Often manufacturers also have great “power” through their IoT products. As an illustration, there was an incident where a user had expressed sharp criticism (e.g. “this product is junk”) in a customer report about a smart garage-door (internet-enabled) called “Garadget”. With Garadget, the user is able to open or close his garage door via an app or to check whether the gate has not been left open unintentionally. The manufacturer did not like the customer’s disapproval of their product and informed the user that their device can no longer be connected to the server. As a result, the customer could not control his garage door via the app. Such incidents, where the manufacturer has power of smart devices, could discourage customers from purchasing IoT products (Kelion 2017).

In other words, to make best use of these novel technologies, it is desirable to empower end-users to customize their IoT programs (instead of relying on pre-programmed functions by the developers).

1.1 Research Scope

Motivated by the developments and challenges outlined above, we investigate in this thesis how we can provide end-users (with and without explicit computer-skills) with an innovative way for programming these kinds of IoT applications in an intuitive fashion. The goal is to support a set of interactions for IoT applications which are simple to handle and easy to understand by end users. Hence, the aim of this work is to develop a “controller”, which is intended for programming and configuring networks of sensors and actuators. To this end, we apply a design-based or Research through Design (Zimmerman, Forlizzi et al. 2007) approach for iteratively exploring desired interactions and form factors.

The primary target groups are end-users who are interested in an alternative tangible approach to enhance specific tasks in their (smart) homes (e.g., creating a customized ambient assisted living technology) as well as designers who want to augment their physical prototypes with interactivity in little time.

This scoping of the thesis leads to a number of research questions as outlined in the next section.

1.2 Goals and Research Questions

The primary goal of this thesis is to develop an alternative approach to programming sensors and actuators (i.e., to create programs with sequences of actions and reactions), apart from what already exists on the market. In this way, we also aim at turning “passive consumers” into “mature consumers”, so that end users can run their creativity freely and are able to control their own system according to their specific needs and not as the developers anticipated their requirements. In short, we want to empower them by addressing the following research questions:

What are the prevalent paradigms in supporting end-users to program¹ the IoT and where are the strengths and weaknesses in these paradigms or approaches?

Can we support end-users of different skill levels in programming the IoT using tangible and smart objects?

Are the end-users interested in our approach to this challenge in principal?

What kind of IoT applications or needs would the end-users be interested in?

To what extent can IoT applications be configured with tangibles in contrast to conventional approaches, e.g., using an app?

Where are the boundaries of employing tangibles and similar concepts for programming the IoT?

The IoT and the design of interactive products in a user-centered process are an integral part of the master program of Vienna University of Technology. Therefore, the main focus of this thesis will be on developing a functional prototype with the user in mind. More specifically, the focus is on the interaction between the user and an "interactive product". The "interactive product" is a mature prototype, implemented to investigate with participants whether the interactions are meaningful and sufficiently user-friendly. Since user-friendliness is an elastic term, scientific methods (e.g. focus group, field observation) are used for testing, which will be discussed in the method chapter. Implementing all interactive features successfully can be a challenge. For example, sometimes it may be more sensible to remove a feature, but to improve another property. This could be, for example, an expensive material for better handling - which reduces the affordability. Reliability could be realized by concentrating on only a few main features, which then really work reliably instead of offering as many features as possible. Furthermore, too many features can also reduce user-friendliness. Thus, for all of these reasons and to tackle the challenges, we opted for an iterative, design-based, and user-centered approach.

1.2.1 Limitations

The following aspects are out of scope of this thesis. The goal of this work is *not* to create a market-ready prototype and we do *not* want to develop a new computer programming paradigm with this project. It's more about building a functional prototype, with which we can observe the behavior of the users, how they use and make sense of the prototype. That is, the focus is not on the shape and appearance of the devices but their behavior.

¹ by "programming" we primarily mean configuring sequences of actions and reactions of smart devices

We also do not aim at generalizing the research findings. Rather, we provide a qualitative and design-based account of novel concepts that might be interesting for configuring the IoT in the near future.

1.3 Contributions of the Thesis

The contributions of this thesis are threefold. Firstly, we review existing “paradigms” for programming IoT applications and create a summary (see section 2.7). In this course, we do not just investigate how to program IoT products, but we are also looking at how IoT applications are currently used and sold, and whether there are special interaction “paradigms” which are implemented in the different application (see section 2.6). Hence, we examine for each application how they support the users in their IoT tasks and if there are difficulties or problems in their design. Attention is also paid to IoT applications with poor user experience, as we are interested in learning from common shortcomings. Secondly, by means of the acquired knowledge and a user-centered and design-based process, we develop our own interaction concepts and tangible programming interface for IoT applications (see chapter 4). Thirdly, we evaluate these ideas and the tangible interface with a total of 43 participants, and we provide a qualitative report of the findings (see chapter 5). The main focus here is on the user experience of the prototypes and how easy they can be handled.

1.4 Structure of the Thesis

The structure of this thesis is as follows. After this introductory chapter 1, where we want to explain the intention, goals and research questions of this thesis, we continue to review related work with respect to programming IoT in chapter 2. On the one hand, we take a detailed look at commercial IoT products and research products in the academia and for education, but on the other hand we engage also with end user development in the IoT and of tangibles. Then we analyze and summarize all literature and findings from market reviews using a table with different feature dimensions. In Chapter 3 we present the methods and approaches we have used in this work. We deal with the terminology about “design” and the role of design in the HCI community. This leads to the “Research through Design” approach and to user-centered design and related user research methods, which have been used for knowledge generation in this thesis. In Chapter 4, we outline this iterative design process, and we describe the development of the prototype and its evaluations in detail. In the course of the thesis work, three prototypes were developed in an iterative way. The findings of the evaluation of the last developed prototype are described in chapter 5. The evaluation was carried out in two phases with technically experienced participants and technically inexperienced participants. This work concludes with a discussion in chapter 6 in which we respond to the research questions with the obtained findings. As well, we briefly reflect on our personal experiences with designing for the IoT.

2 Related Work

As the UK Government has forecasted, the IoT will be the “next big thing” in computing (Walport 2014), and there is already quite a number of commercial and research products available. In this section, we present an assembled list of *commercial products*, especially in the context smart home, and a list of *educational tools or maker tools* (e.g., IoT software for developing applications or modeling tools). We also recap the related work with particular reference to what we are attempting to accomplish in this thesis work. In this course, we briefly elaborate a list about *End User Development in IoT and of tangibles*. It is important to keep in mind that these are only a few exemplary products respectively projects out of many others as the number of new tools growing at a fast rate, and that each one of these product, software, tools or toolkits comes with its own particular features and target groups.

Before we go on with the list of commercial IoT products, we will briefly take a look at what the “Internet of Things” actually is and how different researchers and manufacturers have defined it.

2.1 Defining the Internet of Things (IoT)

The Internet of Things (IoT) can be traced back to Kevin Ashton, who introduced the notion in 1999 during a presentation he gave at the Procter & Gamble Company, a US consumer goods group, which sales a wide range of cleaning agents, personal care and hygienics products. He used the term in the context of RFID (radio-frequency identification) technology and the Internet (Bude and Kervfors Bergstrand 2015). According to Ashton in 2013, the reason why IoT has become so popular (cf. Introduction) is due to the laws of Moore and Koomey (Rose, Eldridge et al. 2015). Moore's law states that the number of transistors per area is doubled approximately every 18 months. Koomey's law also states that not only the performance of processors improves every 18 months, but also the energy efficiency (Koomey, Berard et al. 2011). This has led to smaller and smaller computing units consuming less and less power, and becoming increasingly cheaper. These developments were important for the breakthrough of IoT.

It is difficult to define the term IoT clearly, because there are different uses for this notion (Atzori, Iera et al. 2010, Mashal, Alsaryrah et al. 2015). For example Vermesan and Friess (2013) described IoT with a broader vision, since the

“Internet of Things (IoT) is a concept and a paradigm that considers pervasive presence in the environment of a variety of things/objects that through wireless and wired connections and unique addressing schemes are able to interact with each other and cooperate with other things/objects to create new applications/services and reach common goals” (Vermesan and Friess 2013, p.8).

Further Tan and Wang (2010) defined in a “things-oriented-paradigm” IoT as

“things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts” (Tan and Wang 2010, p.376).

Another more technical definition can be found in the final report of the Coordination and Support Action (CSA) for Global RFID-related Activities and Standardisation (CASAGRAS), where they defined IoT as

“a global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent cooperative services and applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability” (Serbanati, Medaglia et al. 2011, p.2).

Nevertheless, they all have one thing in common, to integrate physical objects with the virtual world of the Internet. The “things” can be omnipresent objects such as containers, cars or refrigerators, but also rooms or things in the environment (e.g. rivers, glaciers). In principle, any object, place or person can count as a “thing” (Haller 2010). To sum up, we can say that the basic concept of IoT is to combine things that can then exchange information and communicate with people. IoT is intended to facilitate the exchange of information between the real and the virtual world. In the real world, there is a state of information (e.g. outside it rains, package lies at the post office) which is not yet represented in the Internet. With the help of IoT, this information gap between the virtual and the real world can be closed. These information states can then be evaluated either by machines or humans. The purpose of this invisible information is to assist people in their activities seamlessly and unobtrusively. Thanks to the ever smaller computing units, embedded computers are no longer noticeable and do not distract (e.g. Wearable Computing (Starner 1996)). In practice, the information (state) is measured with sensors. Sensors are often used today, even if we do not always notice them. The omnipresent smartphones, for example, today have powerful sensors such as GPS, light sensor, motion sensor or a camera. Kevin Ashton said that IoT technology is already available, but is not necessarily visible to all of us. So it can take a long time until people notice it (Bude and Kervefors Bergstrand 2015).

2.2 Defining the Term Programming with Respect to this Thesis

The term programming has different meanings for many people. Therefore, we want to briefly explain what we mean by *programming* (with our prototype) in this work.

Depending on the technical experience the users have undergone, the word “programming” is used differently by them. For instance, most people would not say that they have programmed a calculation in Microsoft Excel, but there are some people who say that they have programmed a web page in HTML or some say colloquially that they have programmed their thermostat, their car radio or their TV programs for the SAT receiver (Blackwell 2002). Nevertheless, all these examples have one feature in common. They determine a certain behavior, which are to happen in the future. Also in this context, users can set a specific behavior.

In this thesis, we work with the typical “if this do that” or also known as “when trigger do action” patterns (trigger-action model) and the users can *program* these patterns themselves for their activities. By programming, we do not mean that the user must program a module, where they must determine the behavior of the sensor at the hardware level. Instead, the user is allowed on their own to determine which sensors they want to use and which actuators (which are used as triggers of actions) they want to activate with them. Ur, McManus et al. (2014) claimed that this “trigger-action model” can express the most desired behaviors and even inexperienced users can create programs in this way containing multiple triggers or actions.

2.3 Commercial IoT Products

Products for the Smart Home: One of the most popular IoT technologies is the commercial product Philips Hue (Philips 2012) from the Royal Philips Electronics of the Netherlands (short: Philips). The system comprises a LED light bulb (see Figure 2.1) where the user is able to set the light color and brightness over a smartphone application or over a web browser. The Hue light is connected with ZigBee over a bridge, which is connected to a WLAN router. For the Philips Hue light bulb we can find many smartphone apps which offer features like setting the light atmosphere to match with a specific photo (on the smartphone when revisiting images) or turning the light bulb on when a user enters a room carrying the smartphone (recognized via GPS or connected to Wi-Fi). Another example would be an app for changing the lights to a specific color when our favorite football club scores a goal (e.g. when Borussia Dortmund, whose club color is yellow, scores one of their many goals, the light bulb changes the color to yellow, and after a short time back to daylight white). Other commercial smart home applications are the Nest thermostat (Nest 2010), ecobee (ecobee 2007), Fibaro Home Automation (FIBARO 2014), WeMo from Belkin (Wemo 2011) or Netatmo (NETATMO 2011). The Nest thermostat (see Figure 2.1) learns the users’ heating habits to reduce energy costs. The thermostat unit features a display which indicates the desired room temperature and with a controller on the same unit the users can set the temperature. Over WiFi, the users are able to control the unit remotely with a smartphone app. The benefit of this central air conditioning system is, that the users do not have to configure or administrate the temperature in their house, the temperature will be set based on learned behavior.

The Fibaro Home Automation, WeMo or Netatmo all provide a set of sensors and actuators, like a flood or a smoke sensor and wall plugs for power metering. Netatmo provides additionally weather stations like rain and wind gauge. It should be noted that

all of these smart home technologies require a smartphone app to configure the system. Furthermore there is Amazon Echo (Amazon 2015) which is a speech-based home assistant where all connected devices (e.g. Philips Hue) in the smart home system (e.g. WeMo or Netatmo) can be controlled with voice-control. Although voice-controlled handling is simple and convenient, the devices must be configured before they can be voice-controlled. For instance, to use a wireless WeMo switch with Amazon Echo we first need to configure the switch with the Belkin app, and then use the Amazon Echo app to add the switch to the list of voice-controlled devices. The setup takes place without great difficulties, however, a smartphone or tablet is required and knowledge of how to deal with apps (e.g. downloading the software, installing, and operating it). In addition, a separate app is required for each brand, currently it is not possible to configure all devices directly with the Amazon Echo app.



Figure 2.1 – Examples of commercial IoT products: right Philips Hue² and left Nest Thermostat³

Maker Tools for the IoT: There are also some commercial *maker* tools⁴ like 3IoT Complete or Windows 10 IoT Core. The Austrian cellphone provider Drei offers a 3IoT Complete (Drei 2016) product for start-ups and developers. It is an all-in-one IoT box with temperature and humidity sensor, a built-in SIM card for Internet access and an online-platform for collecting the measured data and also for making them available over an interface for other systems (e.g. webbrowser). Also Microsoft offers an Windows 10 IoT Core Operating System (OS) (Microsoft 2015) for free, which supports many publicly available boards like Raspberry Pi, MinnowBoard or Banana Pi. In combination with a Windows 10 IoT Core smart-box there are projects like, for example, the smart refrigerator from Liebherr (Liebherr 2016), which should help people in the storage and procurement of food. These commercial maker tools clearly show that there is a demand for IoT products that can be created quickly.

² <https://www.meethue.com> (retrieved 10.12.2017)

³ <https://store.nest.com> (retrieved 10.12.2017)

⁴ These are tools for “rapid prototyping”, to quickly build a prototype in order to try out design concepts. This includes not only 3D printers, laser cutters or vinyl cutters for fabrication, but also development boards like Arduino Boards, Makey Makey, littleBits and Raspberry Pi.

PRODUCT NAME	APPLICATION AREA	USER INTERFACE	PREKNOWLEDGE	TARGET GROUP
PHILIPS HUE	smart home	app, web browser	little: – to use with apps	technophile
NEST THERMOSTAT	smart home	On device, app	little - none for usage	technophile
ECOBEE	smart home	On device, app	little – to use apps	technophile
FIBARO HOME AUTOM.	smart home	app, web browser	few – to deal with apps	technophile
WEMO	smart home	app, web browser	few – to deal with apps	technophile
NETATMO	smart home	app, web browser	few – to deal with apps	technophile
3IOT COMPLETE	rapid development, maker tool	web browser, app	few to advanced	developers, start-ups, inventors
WINDOWS 10 IOT CORE	Development, maker tool	computer, app	advanced electronics knowledge	developers, start-ups, inventors
AMAZON ECHO	entertainment, smart home	voice control, app	few – to deal with apps	technophile

Table 2.1 - List of commercial IoT products

In summary, while the pleasant user experience of smart home applications suggest a “smart feeling” for the users due to its fluency of interactions and the wireless connections, the systems nevertheless need to be configured beforehand. To this end, often a smartphone app is required and knowledge about how to deal with the app. Table 2.1 summarizes all already mentioned IoT products, with their application area, user interface, required preknowledge by the users and intended target group. For some products the target group was labelled as “technophile”, because the IoT products have not been on the market for a long time and therefore these technically oriented customers are early adopters. Nevertheless, the companies hope for mainstream products.

2.4 IoT Research Products used in the Academia and for Education

Besides the commercial smart home products there are also a lot of non-commercial products or educational products. As summarized in a TEI paper (Güldenpfennig, Dudo et al. 2016), recent examples of IoT hardware-prototyping projects for tangible interaction include Arduino (Arduino 2005), .NET Gadgeteer (Villar, Scott et al. 2012),

littleBits (Bdeir 2009), SAM (SAMLabs 2014) and Project Bloks (Blikstein, Sipitakiat et al. 2016).

One of the most popular physical-computing-platforms is Arduino, which manufactures ready-to-use single-board microcontrollers and microcontroller kits. Arduino allows an easy access to control interactive objects (i.e. sensors and actuators) with the aim that even people without professional programming or electronic knowledge (e.g. artist) can build digital devices that can control and sense objects in the physical world. All boards of Arduino are open-source and every other manufacturer is allowed to create boards with the same equipment. Furthermore, ready-made "shields" are offered, with which the user can easily extend the Arduino with functions (for example Xbee to communicate wirelessly). Thus, the user can put a specific shield module directly on the Arduino without having to wire the module. Further, users can use TinkerKit⁵ to acquire electronics knowledge without soldering, where interactive objects can simply be connected/wired directly to the TinkerKit. Another single-board microcontroller is .Net Gadgeteer, where the user has ready-made modules such as a camera or a display provided. Unlike Arduino, .Net Gadgeteer is developed with a high-level programming language (C#) allowing the user to develop more sophisticated applications. Another project that also gives the user an easy introduction to making devices with interactive objects is BASIC Stamp⁶ from Parallax Inc. The board is programmed with BASIC and an editor software is provided by the manufacturer.

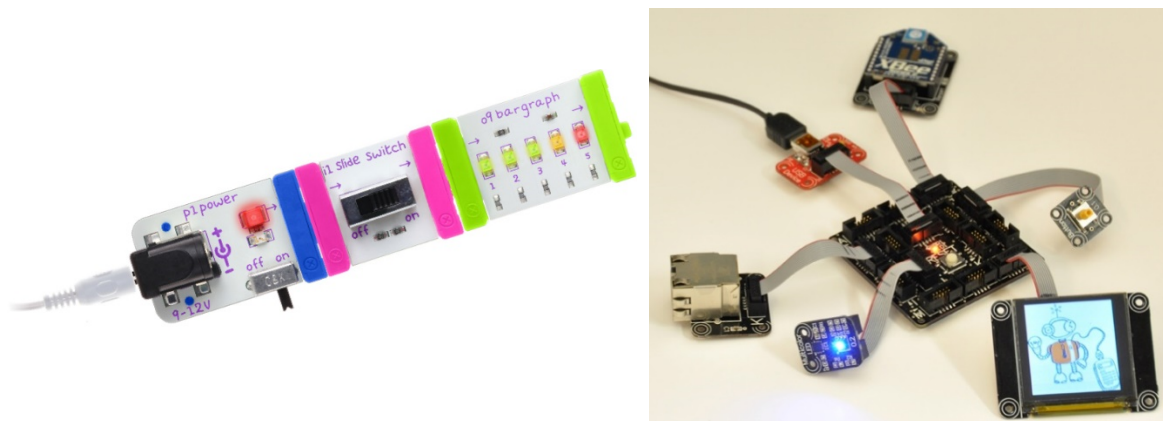


Figure 2.2 - Examples of IoT products for education: right littleBits⁷ and left .NET Gadgeteer⁸

In addition to the microcontroller boards to learn electronics (or programming) skills there is also a product named littleBits. Here (little) modules with magnets are used. A module may be an actuator such as a buzzer or an LED lamp or a sensor such as a photoresistor or pushbutton. These modules are connected together with magnets in a linear arrangement, the magnets should also prevent the modules can be connected wrong. With this simple handling first electronics knowledge can be explore due to an interactivity way, which is particularly well received by children. Another similar project is

⁵ <https://store.arduino.cc/tinkerkit-starter> (retrieved 12.03.2017)

⁶ <https://www.parallax.com/catalog/microcontrollers/basic-stamp> (retrieved 12.03.2017)

⁷ <https://littlebits.cc> (retrieved 10.12.2017)

⁸ <https://www.microsoft.com/en-us/research/project/net-gadgeteer> (retrieved 10.12.2017)

Project Bloks, which is an open hardware platform to enable developers to get a quicker entry into the tangible programming experience (for kids). The team developed a modular system for tangible programming with the help of electronic boards and programmable “pucks”. These devices (pucks) can be connected together like littleBits. It should be noted, that littleBits or Project Bloks prototypes are more restricted than projects like in Arduino, .NET Gadgeteer or BASIC Stamp. A project which avoids this linear arrangement is the SAM platform, which uses wireless modules as sensor and actuator. This means that the modules do not have to be wired at SAM, as in the previous projects, but instead are programmed with software which module should be connected to which other module.

This approach of programming the modules with a visual programming language is also used in the following projects like for example Lego® Mindstorms, vvvv (Joreg 1998), MetaCricket (Martin, Mikhak et al. 2000) or d.tools (Hartmann, Klemmer et al. 2005).

PRODUCT NAME	APPLICATION AREA	REQUIREMENTS	PREKNOWLEDGE	TARGET GROUP
ARDUINO	Rapid development, to facilitate the access of technically less savvy people to programming and microcontrollers	Computer, Arduino-board, electronic components	Basic programming and electronics skills	Students, artists, makers, thinkers, inventors, designer
LITTLE BITS	Rapid development, to experiment with electronics or just explore and have fun	littleBits Exploration Kits	None (learning by doing)	Kids (age of 8), Artists, Students, Educators, Adults
SAM	Rapid development, to experiment with electronics, learning coding by doing	Computer or Tablet, SAM Kit	Basic programming and electronics skills	Students, Artists, Makers, Thinkers, Inventors
PROJECT BLOKS	Research, Framework	Computer, electronic components	Basic programming and electronics skills	Developers, Designers, Researchers
LEGO® MINDSTORMS	Toy for children	Lego, Computer	None (learning by doing resp. trial and error)	Kids, Educators, Thinkers, Inventors
.NET	Rapid	Computer,	Good	Developers,

GADGETEER	development	electronic components, .NET Framework (Windows)	programming skills (c#)	Researchers
METACRICKET	Rapid development	Computer, electronic components	Basic programming and electronics skills	Developers, Researchers
D.TOOLS	Rapid development	Computer, electronic components	Basic programming and electronics skills	Developers, Researchers
BASIC Stamp	Rapid development	Computer, electronic components	Programming skills (BASIC)	Students, Educators, Developers, Researchers

Table 2.2 - List of IoT Research Products in the Academia and for Education

Table 2.2 summarizes all projects from academia and for education as described in this section. As we can see from the table, in most of all applications a computer is required and some knowledge of the software. In conclusion, on the one hand there are sensor and actuator modules as hardware without links and logic between them, on the other hand end user programming is needed to define these links and the logic. To program applications for Internet of Things remains (unnecessarily) difficult, because the developers have to program different operating systems, focus on specific hardware platforms and develop network interactions (Kovatsch, Lanter et al. 2012). Only IoT projects, which have also been developed specifically for children, do not require any programming skills.

2.5 End User Development in the IoT and of Tangibles

We can define in line with Lieberman et al. mentioned, that End user Development (EUD) can enable users to create, modify or extend systems or programs independently without having any programming knowledge (Lieberman, Paternò et al. 2006). Shneiderman argues that “old computing” focuses on what the computer can do for the user, while the “new computing” is designed for what the user can do with the computer (Shneiderman 2003). Therefore, with the help of EUD, users of digital devices are increasingly evolving from “passive consumers” to “active producers” (Fischer 1998) and according to Costabile, EUD is the ideal approach to empower users to create their own IoT applications without realizing that they are developers (Costabile, Mussio et al. 2008). Typical programming paradigms for EUD are “programming by demonstration, programming with examples, visual programming or macro generation” (Costabile, Fogli et al. 2006) or as defined by TOCHI⁹ (ACM Transactions on Computer-Human

⁹ <https://tochi.acm.org/end-user-development-for-the-internet-of-things/> (retrieved 12.05.2017)

Interaction), EUD can take different forms such as: “recording and packaging repetitive interactions into macros, building increasingly sophisticated models to predict future behavior, programming by example, or even developing infrastructure to support new devices using popular programming languages”. However, the EUD of IoT applications does not only include the (graphical) interface and the behavior of an interactive system, but EUD includes all elements in the IoT environment. Thus, we distinguish between activities that can be made on the hardware or software level (Barricelli and Valtolina 2015).

There is already some investigation of EUD for Internet of Things with the goal to define and program smart things. For example, there exist some (proprietary) frameworks for developing Internet of Things applications like WoTKit (Blackstock and Lea 2012), Apple’s HomeKit (Apple 2014), Samsung’s Smart Things (Samsung 2012), Google’s Weave (Google) and Vera3 (Vera) or an open source project Node-RED (IBM 2014). The project User Interfaces for Smart Things (Mayer, Tschofen et al. 2014) represents a “model-based interface description scheme” for developers. Instead of using interactors like text input or value selection in a software application, they used a description scheme for smart things that captures the semantics of an interaction with the device. This allows the developer to quickly develop a smartphone app, which enables the end user to configure their smart devices (for example a dimmable switch). Their goal of all these projects is to enable rapid development of Internet of Things applications with minimal effort for users who want to develop applications for themselves or others. Instead of using General Purpose Languages (GPL) (like java, c++, etc.) the user can use a scripting language (e.g. Groovy), as this gives an easier access to programming for the user, or a browser-based flow editor like Node-RED, which is a visual tool for wiring the Internet of Things. Another example is, that the user can use the free web-based application/service “IF This Then That” which also runs as an app and allows the user to communicate with their internet-connected hardware. As the name implies, the service can trigger notifications that IFTTT receives. For example, we can get notified when rain is expected in our area. The notification (output) can then also take place “via a hardware”, so that, for example, all windows in our house will be closed automatically, when IFTTT notified us that it should start to rain. A big advantage of this service is that predefined “recipes” are provided to the user, with which the user does not have to program a desired behavior, but only has to search for it. The recipes are often provided by the manufacturers themselves. In addition, there are also plans for an own IoT platform, such as Midgar (García, G-Bustelo et al. 2014), which for example has its own Domain Specific Language (DSL) with which the researchers want to handle the “application generation problem” (as they call it) and additionally provide a graphic editor with which the creating of the DSL should be simplified. A similar program called Modkit (Millner and Baafi 2011) supports the user for Arduino projects with a “drag and drop programming” and a special shield for Arduino.

Incidentally, it should be noted that researchers found out that frameworks like Samsung-owned Smart Things could facilitate creating a malicious app to unlock doors or induced a fake fire alarm (Fernandes, Jung et al. 2016). The problem with a proprietary system is that neither the user nor the developers know how these systems work internally and whether these systems can be a threat (e.g. security gaps in the

home network - devices are normally protected by a firewall or an operating system)(Simpson, Roesner et al. 2017).

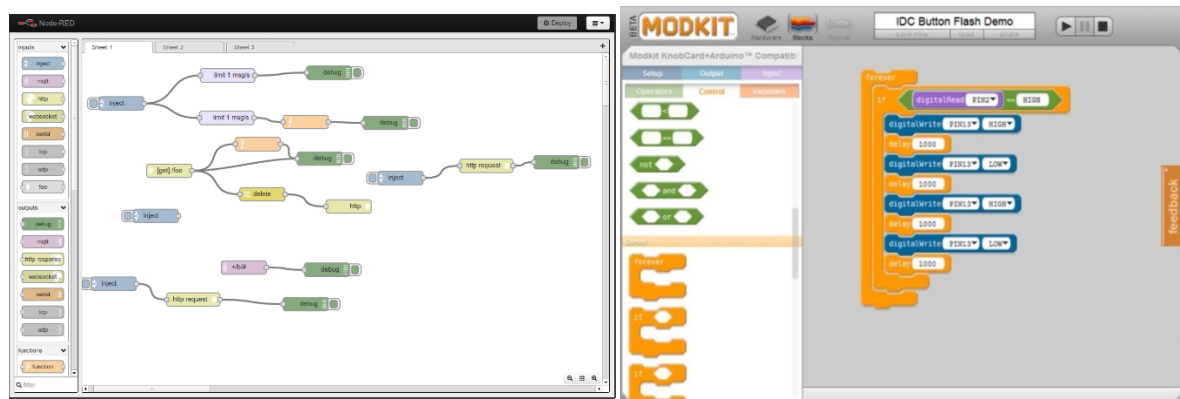


Figure 2.3 – Examples of visual (web browser) tools for wiring the IoT: right Node-RED¹⁰ and left Modkit (Millner and Baafi 2011)

Another example of end-user programming/EUD for smart homes is the use of voice (Kubitza 2016). This requires a computer where the voice control is configured, which is more for advanced IoT users, because much more configuration is necessary here. For example, when we say "turn on the lights in the garage and in the living room", we have to teach the program first which smart light bulb is in the garage or in the living room. As already mentioned, Amazon Echo also works by voice control, the handling is simple and it also offers additional services such as weather forecast, current daily news or traffic announcements with an Internet connection, so it is not just a unit to control devices in the house. As a competitor, Amazon is faced Google Home (Google 2016), which works similarly. Nevertheless, the behavior of which devices the user wishes to control with which voice command must be programmed by the user. For example, in Amazon Echo, the user can program their own "skills" on a web-based page using the "Alexa Skills Kit". This kit includes APIs, tools, documentation and code samples, and additional Amazon promises its users to quickly create new "skills" through a step-by-step tutorial (LaCharite 2016). Furthermore, Amazon offers its own "smart home Skill API"¹¹, with which the user can program cloud-controlled devices. Thus, the entire device logic runs in the cloud, instead on a device of the user.

End User Development of Tangibles: In addition to IoT, we will look at EUD in the context of tangible computing/tangible programming. The core idea of Tangible Interactions is to experience and use interactive systems by drawing on tangible and real objects as the interfaces between human and the machine. The objects are often equipped with RFID (radio-frequency identification) or NFC (Near Field Communication) technology as links to digital information. The interaction is thus no longer visually but haptically tactile, the data is directly manipulated with the hands (Hornecker 2008). A common example for tangible programming is the Durrell Bishop's Marble Answering Machine (Poynor 1995), where incoming calls are represented with colored marbles.

¹⁰ <https://www.ibm.com/blogs/emerging-technology/projects/node-red-3> (retrieved 10.12.2017)

¹¹ <https://developer.amazon.com/de/alexa/smart-home> (retrieved 21.09.2017)

There are two bowls where we can put the marbles. If one of these marbles is placed in the first bowl, the spoken message will be played. If we place a marble in the other bowl, the caller will be called back.

Several projects provide ready-to-use physical widgets (as sensors and actuators), such as Phidgets (Greenberg and Fitchett 2001), VoodooIO (Villar, Gilleade et al. 2007) or iStuff (Ballagas, Ringel et al. 2003). Phidgets provides some USB devices where, for example, any sensor can be connected (assuming it runs at 5V), such as an on / off switch or any other I / O device that is controlled with a bit value. All these devices are controlled by a computer, therefore it is not necessary to program the microcontroller. iStuff works similar, except that it works with wireless devices and focused on rapid prototype applications. Also VoodooIO works similarly, as it provides a set of devices that however must be located on a certain (conductive) ground. This ground serves as a bus for communication between the devices and as a power supply. Another project called Papier-Mache (Klemmer, Li et al. 2004) supports computer vision as well as RFID and barcodes.

In summary, it must also be emphasized that all these research projects or products require programming or electronics knowledge. Table 2.3 summarizes the mentioned examples of EUD in the IoT and the examples of related tangibles, including their application area, interaction type and intended target group. As afore-mentioned some products are labelled as "technophile", because the products have not been on the market for a long time and therefore these technically oriented customers are early adopters.

PROJECT	APPLICATION AREA	INTERACTION TYPE	TARGET GROUP
WOTKIT	creating IoT applications	coding	technophile, developers
NODE-RED	rapid development, creating IoT applications	visual programming language	technophile, developers, makers, thinkers, designer
APPLE'S HOMEKIT	creating IoT applications	coding	technophile, developers
IFTTT	creating conditional constructs	instructing	technophile, developers, makers, thinkers, designer
MIDGAR	creating IoT applications	modeling language	developers, researchers
SAMSUNG'S SMARTTHING	creating IoT applications	coding	technophile, developers
USER INTERFACES FOR SMART THINGS	rapid development, creating IoT applications	modeling language	developers, researchers
MODKIT	rapid development, creating IoT applications	modeling language	developers, researchers, makers, thinkers, designer
GOOGLE'S WEAVE	creating IoT	coding	technophile,

	applications		developers
VERA3	creating IoT applications	coding	technophile, developers
AMAZON ECHO	entertainment, smart home	converse with computer, instructing	technophile
GOOGLE HOME	entertainment, smart home	voice, converse with computer, instructing	technophile
PHIDGETS	research/makers	utilizing materials	researchers, developers
VOODOOIO	research/makers	utilizing materials	researchers, developers
ISTUFF	research/makers	utilizing materials	researchers, developers

Table 2.3 - List of IoT and tangible EUD applications

2.6 Interaction Types in Programming the IoT

Before we make a summary of the background and go over to the methods, we consider it important to recap all previously collated works and products, by elaborating clear table with the approaches across different dimensions. As mentioned in the textbook "Interaction design: beyond human-computer interaction" , the authors Rogers, Sharp et al. (2011) recommend to classify interactive systems according to "Interaction Types". They say, in principle, there are types such as instructing (a command, for example, mouse click or typing in a terminal), direct manipulation (e.g. avatar is controlled by Kinect directly and continuous), conversing (computer is a conversation partner) and exploring (e.g., VR world is explored).

Since we have already explained that there are different concepts, operating modalities and research approaches (for example, selling a toolkit like littleBits versus defining a description language), we now want to offer a clear table of prerequisites (both knowledge and equipment) for operating most IoT devices (Table 2.4). In short, it can be stated that end-user programming or EUD in the Internet of Things can be accomplished using voice (e.g., Amazon Echos), by utilizing tangibles/materials (such as littleBits), by modeling tools (e.g. Node-RED) or by programming/creating code (e.g. .NET Gadgeteer).

IOT APPLICATION AREA	REQUIREMENTS	PREKNOWLEDGE	PROBLEMS	INTERACTION
COMMERCIAL PRODUCTS LIKE E.G. PHILIPS HUE, NEST THERMOSTAT	Additional hardware such as computers or smartphones	To be able to use mobile apps	Each task requires a separate app, Restricted use (manufacturer	Instructing (e.g. clicking on device or smartphone), Converse with

			specifies what the user is allowed to do), Manufacturers have power of smart devices (e.g. can turn off devices), Privacy, Security gaps	Computer (voice)
TOOLS FOR EDUCATION LIKE E.G. ARDUINO, LITTLEBITS	Additional hardware such as computers, tablets or electronic components	Basic programming and electronics skills	Linear arrangements and restricts	Utilizing materials, Instructing, Coding
END USER PROGRAMMING (BOTH IN COMMERCIAL PRODUCTS AND EDUCATIONAL TOOLS)	Additional hardware such as computers	Basic to advance programming skills	It is hardly possible to quickly link a few sensors and actuators without programming, Proprietary software	Converse with Computer (voice), Modeling language, Visual programming language, Coding, Instructing

Table 2.4 – Summary of prerequisites (both knowledge and equipment) for operating most IoT devices.

Table 2.4 provides a summary of different IoT application areas, divided into requirements, pre-knowledge and problems. The subdivisions are a summary of the related works, which criteria are most common or relevant. While there are commercial products (Nest thermostat) that, for example, do not need a smartphone to configure, most other products require such a device or a tablet. Therefore, it can be stated that a smartphone is needed in majority of all IoT applications. These four subdivisions are intended to represent a compact dimensionality or order to structure the initial explorations and concepts in this thesis, because they are, in our opinion, essential for the further approach. The necessity of these four dimensions can be justified as follows. From the requirements follow that the user is bound to certain hardware. Without additional hardware (e.g. smartphone) the user cannot control the device. The pre-knowledge requires, as the name suggests, that the user knows how to configure or operate a new device. This is not just about the user having to read a usage instruction or manual but having some programming skills. The column with problems indicates which issues could be better solved within the scope of this thesis (e.g. restrictions) or what things could be excluded (e.g. security gaps). The last column summarizes different interactions that can be offered to the users. The topic “interactions” plays an

important role in this work (mainly due to the course of studies in media informatics). Which interaction will be most suitable for this thesis will turn out later. Nevertheless, this column should not argue which interactions are bad or good, but that there are different interaction possibilities for different applications.

2.7 Summary of Related Work

By means of an intensive literature review and a market research, we found that most systems or products for home automation need to be configured (intensively) before use. Often a computer, a smartphone or a tablet app is required and knowledge about how to deal with the software. This means that people who want to make their own gadgets “smart”, must first learn how to use smartphones in order to then control their devices, since there are no alternatives offered on the market. Apart from the commercial smart home products, there are also some proprietary programs such as “Apple’s HomeKit” (Apple 2014) or “Samsung’s SmartThings” (Samsung 2012) or the open source software “Node-RED”(IBM 2014), which allows users to develop their own IoT applications. As afore-mentioned their goal is to enable the rapid development of IoT applications with minimal effort for users who want to develop applications for themselves or also for others. Instead of using GPL (like java) the users can use a domain-specific programming language or a browser-based flow editor like Node-RED, which is a visual tool for wiring the IoT. Also, the user can use the web-based service “IF This Then That” (IFTTT) (Linden Tibbets 2011), using a trigger-action (if-then) model, which also runs as an app. Furthermore, IFTTT can be also connect to “SmartThings”, for example, and support users with the trigger-action model. With regard to this trigger-action model, Ur, McManus et al. (2014) claimed that it can express most desired behaviors and also allowing inexperienced users to create programs with multiple triggers or actions. In addition, home automations can also be controlled by voice, such as “Amazon Echo” (Amazon 2015) or “Google Home” (Google 2016). It should be noted at this point that although the operation of household appliances via voice control sounds convenient, this behavior (i.e., which device is to be controlled via a voice prompt) must be programmed by the user themselves. For example, “Amazon Echo” requires the user to program so-called “skills” to control their smart home.

On the other hand, “physical computing platforms” are provided to implement IoT applications, allowing the user to create interactive (physical) systems using hardware and software. One of the most successful physical computing platforms is the “Arduino” (Arduino 2005) platform, whose purpose is to facilitate access to programming and microcontrollers for non-tech experience users. For more technically experienced people, “.NET Gadgeteer” (Villar, Scott et al. 2012) exists on the market, which provides ready-made modules such as a camera or a display, and the board can be programmed with a high-level language. Furthermore, ready-made systems are available, with which the user can link actuators and sensors wirelessly (e.g. “VoodoolIO” (Villar, Gilleade et al. 2007) or “SAM” (SAMLabs 2014)). To this end, in all applications or for all products a computer, smartphone or at least a tablet is required and some knowledge of the software.

In summary, we identified the following interaction paradigms, how the user can program their IoT application:

Instructing/Conversing: this can either be done with a mouse click or with typing on a computer (using a software on a PC) by touching a corresponding touchscreen (using an app). For this purpose, the user must install appropriate software or an app (e.g. IFTTT), with which the user can then program their devices. In addition, commands can also be given via voice. It should be noted that even when we use voice control, the IoT devices must first be configured with software before they can be controlled by voice. The weaknesses of such approaches are that we need a separate app for each task or manufacturer, and manufacturers often dictate what the user is allowed to do. Furthermore, such apps or systems may also have vulnerabilities (for example, that a burglar can trick a "smart door" and grant access for unwanted guests), and that the manufacturers have great power over the smart devices and can control (e.g., turn them off) them when needed (cf. example where a user has negatively rated a "garage opener via app" and where the manufacturer in response has prevented the user from opening (or closing) the garage door via app (Kelion 2017)). Many smart devices are cloud-controlled, which also brings privacy into consideration.

Coding: the user can control their devices by using a (higher) programming language. The devices can be programmed directly (for example, "Arduino") or the device is connected to the Internet and is programmed by the user via a webbrowser (for example, "Amazon Echo" or "Smart Things"). Finally, the user can write a program (with a higher programming language like Java or C#) for a computer, a board or an app for a smartphone, where the program control (wireless) the devices (for example, "Philips Hue" (Philips 2012)). This gives the user many options about how the user wants to program their IoT, but for the user to be fully empowered, that user must have knowledge in programming or electronics. For example, although Arduino aims to make it easier for non-engineers to get started, basic programming skills are still required. Nevertheless, thanks to a large community that deals with DIY projects, we find instructions to many problems on the Internet.

Utilizing tangibles/materials: Several projects provide ready-to-use physical widgets (as sensors and actuators) such as "Phidgets" (Greenberg and Fitchett 2001), "VoodooIO" (Villar, Gilleade et al. 2007), "iStuff" (Ballagas, Ringel et al. 2003) or "littleBits" (Bdeir 2009). Also through the use of NFC readers, which are already present on some smartphones, the user can control his devices or a behavior (for example, the user can program a specific NFC-token and then place it in the living room. When the user (or a guest) holds their smartphone next to the token, the user (or a guest) will be connected to the Wi-Fi without typing the password). This area is currently a novel subject of research (especially in HCI), as these approaches do not require experience in using computers or smartphones. Rather, it is a kind of fusion of technology with real objects (everyday objects) where the user can use real objects to perform a function or behavior with them. As a result, we have implemented our project in this area.

Modeling tools: instead of using a higher-level programming language to configure the users' devices, which requires a lot of experience and time as a user to learn a

programming language, as an alternative, either a graphical programming language or a domain specific language is provided to the user. In a visual programming language (for example, "Node-RED"), the user does not develop as usual via a text input but programs by means of "drag and drop" and based on the idea of "blocks and lines", that is, the user assembles visual blocks (treated as entities) and connects these blocks by means of lines (represent relations). An advantage of visual languages is that the user is assisted in programming and that they do not need to memorize possible commands. Thus, visual programs allow the user to create only semantic and syntactic correct programs, that is blocks can be inserted in the program only where syntactically allowed. In contrast, while working with textual input, the user can also make false entries at any point. Further, visual programs assist the user in the work by structurally editing with automatic help functions or appropriate error messages. As a consequence, by "constructing" the program, programming can quickly become cumbersome (especially as many mouse movements are necessary). Visual programming languages are therefore a good introduction to programming, because at the beginning we often do not know the programming term (syntax and semantics) and in this way we are supported by the program. In a domain-specific language (for example Midgar (García, G-Bustelo et al. 2014)), the programming language is reduced to a particular domain (in this case IoT) and often has a declarative description character.

In the subsequent chapter, we go on to detail the methods that we used in order to add own research to the body of related work.

3 Approach and Methods

In this section we want to introduce the methods we used in this work. Before we introduce the methods, however, we want to make a brief digression about design in the area of human-computer interaction (HCI). Subsequently, we will describe specific methods that we used in our research.

From a broader perspective, we follow a *Research through Design* (RTD – see section 3.1.1) approach in this thesis, since we think it is most suitable for the kind of task we want to accomplish. That is, prototyping and evaluation are key in this work. Further, we are also oriented towards a user-centered design approach, which coarsely consists of the following three main phases: Conception, Development and Evaluation (Figure 3.1). These three main phases started with a design workshop for ideation and ended with a final evaluation, followed by the final report and the discussion. Furthermore, we have developed our knowledge and theoretical understanding of the subject in the conceptualization by means of literature and market research. This design process is based on several iterations and tries to generate knowledge from the feedback from the users and experiences of the designer through “hands-on design explorations” or RTD. Hence, knowledge is generated by engaging in the effort to produce a working prototype instead of, for example, analyzing something existing (Zimmerman, Forlizzi et al. 2007). This approach is especially suitable for dealing with “wicked problems” (Rittel and Webber 1973), that is, challenges that can be difficult to solve with conventional research strategies. According to Rittel and Webber (1973), solving a wicked problem can be as complex as defining the problem. The problem solver does not know exactly if they have found the solution, since there is no true or false solution, but only good or bad solutions.

We argue that the objective or problem of this thesis (see research questions below) can be considered a wicked problem as we seek to explore something that does not yet exist and that strongly relies on human interactions with no right or wrong answers.

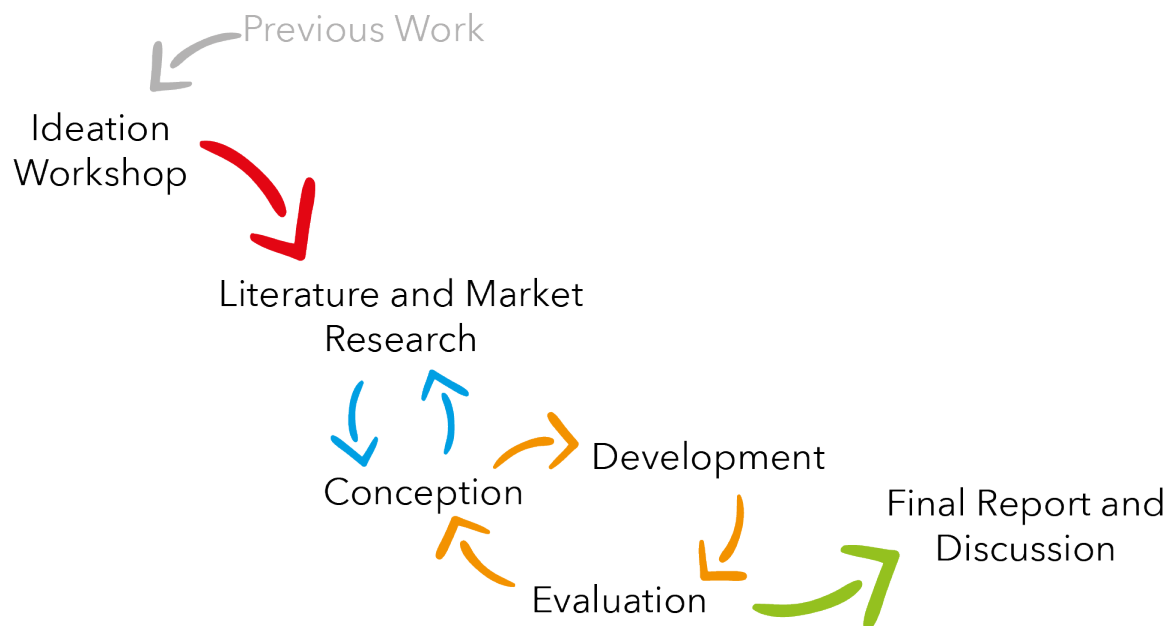


Figure 3.1 - Iterative design process of this thesis which led to the final prototype and concluding evaluation.

3.1 Approach: Design Research in HCI

Before we get started, we want to begin with reflecting on the definition of the elastic term “designer” (Mareis 2014), because there are sometimes inconsistencies in using the term within the HCI community. Since approximately the early 2000s, HCI has increasingly been concerned with design, design research, and its role in the production of knowledge. Thus, detailed thoughts about design is a pretty new movement to HCI.

Bill Buxton ironically said at the CHI 2006 conference that “if everyone is a designer because they select their own clothes, then everyone is also a mathematician, because we all count our change”. With this quote he wanted to show how different the term is interpreted or used. In the HCI community a designer is often seen as a software developer, usability engineer, etc., while in the design community a designer is regarded as someone who “has had training or extensive practical experience in a discipline such as architecture, product design, graphic design, or interaction design.” (Zimmerman, Forlizzi et al. 2007). In this thesis, we also understand the term as it was described in the latter and define ourselves as interaction designers who actually create artifacts in order to generate knowledge about design.

However, HCI was also described as a design-oriented field of research (Fallman 2003) by influential researchers. There are several researchers who are concerned with how far the design term is related to HCI. Löwgren (Löwgren 1995) makes a critical distinction between “creative design” and “engineering design”. In his opinion engineering design

“assumes that the ‘problem’ to be solved is comprehensively and precisely described, preferably in the form of a requirement specification. The mission of engineering design is to find a solution to the problem. The solution must

satisfice the requirements and other constraints, such as cost or performance. Engineering design work is amenable to structured descriptions and seen as a chain of transformations from the abstract (requirements) to the concrete (resulting artifact). Moreover, this structurability makes engineering design impersonal.” (Löwgren 1995, p.87)

In contrast, he describes creative design as being

“about understanding the problem as much as the resulting artifact. Creative design work is seen as a tight interplay between problem setting and problem solving. In this interplay, the design space is explored through the creation of many parallel ideas and concepts. The given assumptions regarding the problem are questioned on all levels. Creative design work is inherently unpredictable. Hence, the designer plays a personal role in the process.” (Löwgren 1995, p.88)

Hence, Löwgren strictly describes the approaches to problem solving. However, it cannot be denied that engineering design can be creative as well and creative design is also seeking for solutions. This stark contrast to the process of creative design is intended to show that practices in HCI often reflect engineering approaches (Wolf, Rode et al. 2006).

A crucial part in design practice is prototyping, and also in the HCI community prototyping is increasingly considered to be important. Fallman (2003) for example, argues that HCI is a research-oriented field where the prototype serves as a “proof of concept”. However, he also mentions that “the design process tends to remain implicit as researchers are embarrassed by not being able to show evidence of the same kind of control, structure, predictability, and rigorousness in doing design as they are able to show in other parts of their research.” (Fallman 2003, p.230)

As mentioned above, a design process can coarsely be divided into three phases: the concept, the prototyping and the evaluation. In his seminal paper, Fallman (2003) identifies problems in the phase of prototyping in some research in HCI. In his words, this phase was often given too little attention, and researchers in HCI regularly considered design as “black art”.

In this thesis, we base our work on design activities, and explicate the design process by a number of iterative prototyping activities, resulting in an interactive and fully-implemented working prototype. We argue that this approach is appropriate in addressing our research questions (RQ) and subquestions (see also section 1.2):

What are the prevalent paradigms in supporting end-users to program¹² the IoT and where are the strengths and weaknesses in these paradigms or approaches?

Can we support end-users of different skill levels in programming the IoT using tangible and smart objects?

¹² by “programming” we primarily mean configuring sequences of actions and reactions of smart devices

Are the end-users interested in our approach to this challenge in principal?

What kind of IoT applications or needs would the end-users be interested in?

To what extent can IoT applications be configured with tangibles in contrast to conventional approaches, e.g., using an app?

Where are the boundaries of employing tangibles and similar concepts for programming the IoT?

3.1.1 Research through Design

RTD can be considered a subcategory of design research which is a general term for theoretical research in the design field - this can be HCI, car manufacturing, product design, etc. Basically, design research is "systematic inquiry whose goal is knowledge of, or in, the embodiment of configuration, composition, structure, purpose, value, and meaning in man-made things and systems" (Bayazit 2004, p.16).

In the field of design research projects, in particular in HCI, *Research through Design* (RtD) has increasingly become established in recent year. There are different approaches and theories for design-oriented HCI research, which we want to briefly discuss in this chapter.

Zimmerman, Forlizzi et al. (2007) claimed that RtD is a suitable method for interaction design research in HCI. They think that the RtD is a good approach for the interface between technical and social science. The design decisions and problems are accompanied by an iterative process, such as prototyping, testing, hands-on activities and critical confrontation. The design decisions are constantly considered from different perspectives to finally get the "right thing", as Zimmermann calls it to make "the right thing: a product that transforms the world from its current state to a preferred state" (Zimmerman, Forlizzi et al. 2007, p.1). The authors believe, that an interaction design researcher can act as the interface between technology and society, because the designer combines knowledge from the social sciences such as a field study with knowledge from technology. Furthermore Löwgren (2007) claims that "designing" is considered to generate knowledge, because "it follows that designers and other actors in a design field, together with their communication and the artifacts of the field, can be seen as a *community for collaborative knowledge construction*." (Löwgren 2007, p.3) Schön (1995) is also of this opinion and calls this as "Knowing-In-Action" because according to Schön, "the knowing is in the action and is revealed by spontaneous, skilful execution of the performance, which one is characteristically unable to make verbally explicit." (Kinsella 2007, p.400) In other words, Schön argues that we get knowledge through *doing something* rather than just *talking about something*.

A number of HCI researchers are working hard on theoretical underpinnings to better integrate RtD findings into HCI. Zimmerman demands more standardized guidelines for RtD projects for the research process. On the other hand, Gaver emphasizes the creative advantages for a research approach where the method has few specifications

(Gaver 2012). He believes that a not strictly specified method allows more exploration, speculation, and thus does not restrict the design space. He also emphasizes that an unspecified method is more suitable for RtD since the theories of RtD can not be falsified (c.f. Karl Popper's scientific "falsificationism" and Rittel's "wicked problems"), because of an unpredictable number of contextual factors during the design process. That is, in contrast to a scientific problem (e.g. mathematics), a design problem can not have a right or wrong result.

There have been some criticisms of RtD that it focuses too much on "artifact creation". Stolterman and Wiberg (2010) have presented their own research model called "Concept-driven Interaction Design Research" with a strong focus on theoretical knowledge. Unlike Zimmerman, they do not focus on the quality of the prototyping process but on the theorizing of how something is used. Building on the approach of Stolterman and Wiberg (2010), Höök and Löwgren (2012) came up with a model for "Strong Concepts" (SC). They claim that interaction design has features of "intermediate design knowledge", which means that

"knowledge can come in many different forms. We are all aware of universal knowledge such as laws of nature, i.e. knowledge in the form of theories that are universally true and applicable. We are also aware of the kind of knowledge that is closely related to a particular artefact or situation, that is, highly contextual and situated knowledge. Moreover, on an intermediate level between universal theory and specific instances there is a variety of forms of knowledge, which are produced, refined, elaborated and sometimes refuted in the ongoing discourse of interaction design research."(Höök, Dalsgaard et al. 2015, p.2)

The key message of these authors is that there is a space between specific design instances and grand theories that can be occupied by knowledge generated through interaction design research. They hope that they can encourage designers to collaborate and communicate their ideas in this way.

Another commonly used approach in the HCI discipline is user-centered design (UCD). While UCD is "less designerly" than RtD it is nevertheless a very powerful and established approach. As stated above, we are also oriented towards UCD. This approach seeks to create (interactive) products with high usability and a great user experience. This goal is achieved by placing the user at the center of the development process with their tasks and needs. The basic idea of UCD is that the product is customized to the user. UCD therefore carefully considers the users have to do, what the users want, how the users work with the product, and so on, rather than forcing the user to use the product as it has been developed (Norman and Draper 1986). The UCD design process is divided into several iterative phases: specify context of use, specify requirements, design and development and at last evaluation.

To some extent, UCD can be considered a proven method for designing systems that fit peoples' needs. RtD, on the contrary, is a research approach which is concerned about producing knowledge by designing product.

Concluding remarks about knowledge

So far we talked a lot about “knowledge”, which plays an important role in epistemology. The question is, when can we talk about knowledge? The Austrian philosopher Popper (2005) has already dealt with this subject in his writings, which revolutionized contemporary thinking on science and knowledge. In his opinion, there is no knowledge in the natural sciences, but only assumptions or hypotheses. This is a basic fact of our lives: we know nothing, we can only conjecture or guess. Popper says that science is a search for truth. He emphasizes that this is not a certain truth, because we can assert or attain the truth, but we cannot attain certainty. So he claims, that “science is the quest for truth, not for certainty” (Popper 1999).

We go on to detail specific design and user research methods as applied in this thesis. We start with a brief description of mainly qualitative research methods, followed by an overview of practical prototyping technologies (like 3D printing or electronic prototyping platforms).

3.2 Methods for Prototyping

In order to answer our research questions, a working prototype and a number of intermediate prototypes have been developed. For the realization of the prototype some common practices in HCI and user experience design were used, which we will briefly discuss in this section.

After and during the built process of the prototypes, we got feedback from potential users, study participants, and experts. Details about these procedures and about the feedback of the participants can be found in the user research methods section and in the findings section. In the following, we focus on prototyping methods, starting from ideation and leading to tools to physically build products.

3.2.1 Explorative Design Methods

There are some approaches for ideation, like making sketches or conducting a workshop.

Tovey, Porter et al. (2003) argue that designers use sketches to “generate concepts, to externalize and visualize problems, to facilitate problem solving and creative effort, revising and refining ideas”. Buxton (2010) puts a lot of emphasis on sketching in through the whole design process. He claims that sketching can be used to detect possible solutions at a very early stage and thus to refine the user experience at the beginning of the design process. According to him, sketching should be an iterative process to develop from the core idea to the optimal solution.

Löwgren and Stolterman (2004) describe that brainstorming is a very useful technique to quickly generate a large number of ideas in a group. “Brainstorming broadly consists of three steps: collecting a group of people, generating ideas without judgment or analysis, and structuring the results to make them useful for further work.” (Löwgren and Stolterman 2004, p.71)

The conceptualization can often be difficult at the beginning of a project without the involvement of other participants. Therefore it is recommended to organize a workshop

at the beginning of a project in order to get interesting input for the ideation. We also started a workshop to facilitate ideation and describe the details in section 4.3 Design Workshop.

In "Thoughtful interaction design" the authors describe that such a workshop (e.g. a "Future Workshop") can be started with brainstorming on problems in the current work situation at the beginning. The entries should then be noted and categorized into problems and then small groups should be formed where each group is to disprove a problem. Then the results should be discussed with each other and each participant should vote for the best results (e.g. 2-5 favorite answers). Afterwards groups will be formed again and the best results will be refined. This is an example of how to hold a workshop based on brainstorming. The basic concept here is that we work in small groups and each group elaborate them together. It is important that everyone is allowed to speak out, even if it sounds silly we should not immediately start criticizing people. This is important, so that there are no "restrictions" and everyone dares to contribute something.

Other elements of design workshops can be the participatory method "World café", which is according to Juanita Brown "a simple yet powerful conversational process that helps people engage in constructive dialogue, build personal relationships, foster collaborative learning, and discover new possibilities for action. Café dialogues enable large groups, often hundreds of people, to think together creatively as part of a single, connected conversation". (Tan and Brown 2005, p.83) The process is such that 4 to 6 participants sit together at a table and are equipped with pens, markers and a paper tablecloth. A moderator explains the process and explains the working method of the workshop and defines several questions at the beginning. Additionally a "table host" sits at each table. The basic concept during the workshop is that after a certain period of time the groups will mix again, that is the participants will change tables and discuss further with the new participants. While the participants have changed the table, the participants can use the paper tablecloth to see what the previous participants wrote on it and the "table host" welcomes the new participants and summarizes the previous conversation briefly and tries to start a new discourse. (Brown 2010)

Furthermore, workshops can also be conducted online using electronic meeting systems (EMS), which provide a range of electronic aids.

Explorative design methods can be used to get a closer look at a topic, if barely "knowledge" exists about an investigative topic, common methods are literature search, market research, focus groups or interviews with experts. In all these solution options an exploratory approach explores the problem without having a lot of concrete information about it. It should also be noted, that the focus group (optionally with design workshop elements) is an attractive alternative to a one-to-one interview, as we often need 30-60 minutes to gather inputs from the participant and with a focus group we can discuss with several participants at once. A benefit of a focus group in contrast to a one-to-one interview can be, that we may have a non-communicative participant, and so in a group discussion, participants can encourage each other to discuss. But this can also lead to that a talkative participant can monopolizing the conversation and thus other participants would not participate in the discussion and so maybe other

interesting points of view would be lost. Here, some sensitivity is required, without being rude to the talkative participant, telling him to let other participants speak as well (Lazar, Feng et al. 2017, p.192).

3.2.2 3D Printing and Rapid Prototyping

The theoretical fundamentals in the field of rapid technological innovation are still quite young, which means that they are constantly changing. As a result, no proper (gold) standard for the approaches has been established in this discipline.

In this work, we use a rapid prototyping research approach, which is enabled by rapid prototyping tools such as Arduino (microcontroller board) and MakerBot (3D-printing, see Figure 3.2). Rapid prototyping means - as the name already suggests - to quickly build a prototype with particular tools in order to try out design concepts. In a broader sense, this includes not only 3D printers, laser cutters or vinyl cutters for fabrication, but also development boards like Arduino Boards, Makey Makey, littleBits and Raspberry Pi (cf. Related Work).

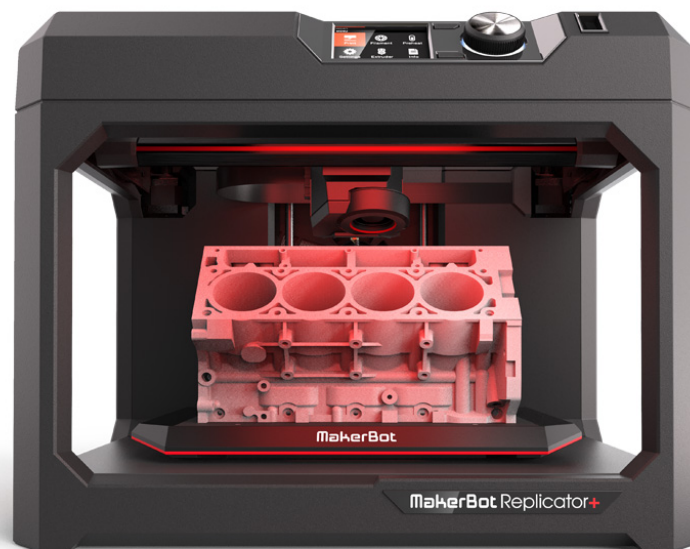


Figure 3.2 – MakerBot Replicator¹³ 3D-printer for rapid prototyping to quickly create physical objects and which was also used to manufacture the prototype.

These useful tools have enabled us to build a working and interactive prototype in a short time and without much effort for our iterative research process. As stated by Rogers (2011), HCI researchers used to explore existing interfaces, and now HCI researchers can use rapid technology to craft their own system and then study how users actually react to it - in the past, this involved a lot of effort and needed experts who had an idea of electronics or manufacturing objects.

Prototypes will mainly be used to test assumptions or for checking the user experience but also for aesthetic explorations. We distinguish between three types of prototypes namely between low-fidelity (lo-fi), high-fidelity (hi-fi), but mid-fidelity (mid-fi). A Lo-fi only represents the crude concept without functionality (for example making sketches,

¹³ <https://store.makerbot.com/printers/replicator> (retrieved 12.09.2017)

paper-prototype with post-it notes or a 3D printed mockup), whereas hi-fi is already very close to the final product and is almost full of functionality (for example a 3D printed prototype equipped with Arduino) (Egger 2000). The functionalities of the lo-fi prototype were simulated by us with the “Wizard of Oz” method (Hanington and Martin 2012), so that the participants could get an idea of how the lo-fi prototype would work technically and we also could better observe how the participants react. Since the hi-fi prototype is quite similar to the final product, often a kind of mid-fi prototype is built (e.g. for proof-of-concept), which, in contrast to the lo-fi prototype, is already interactive, but things like aesthetics, form and materials are not important at this stage

3.2.3 Electronics Prototyping Tools

In order to bring interactive prototypes “to life”, we need to implement electronics. Fortunately, the market already has some ready-to-use electronics tools like single-board microcontroller such as Raspberry Pi¹⁴, .NET Gadgeteer¹⁵ or Tinkerforge¹⁶. Single-board microcontroller are boards, where all the necessary electronic components are combined on a single circuit board. One of the most popular electronic tools is from Arduino¹⁷ (see Figure 3.3), an open source computer hardware and software company, which is partly because there are numerous derivatives. And Arduino finds more and more application in the HCI area. On the one hand it should help the educators to teach “core concepts associated with interaction design such as design process, design methods, psychology models, social science research methods and related software development technologies” to the students (Jiuqiang 2015). On the other hand, as already mentioned above, using Arduino technology, interactive prototypes can enable researchers in HCI to investigate the reactions and feedback of participants.

In a broader sense, using electronic tools like Arduino can also be considered as a form of sketching, where we roughly “sketch” interactive ideas with hardware, rather than using pen and paper.

There are numerous communities online drawing on Arduino boards and similar technologies to create powerful open source solutions based on emended technologies like, for example, home automation systems (smart home). For our prototype we used a completely developed open source framework called MySensors¹⁸, which is a community focusing on DIY home automation and the Internet of Things. Since we are using nRF24L01¹⁹, we will follow the protocols issued by standard Internet of Things protocols set by MySensors where we “can directly find the NRF data sending and receiving methods on conventional WSN networks which are using NRF and Arduino to form nodes.” (Gupta and Raspaille 2015)

¹⁴ <https://www.raspberrypi.org> (retrieved 12.09.2017)

¹⁵ <http://www.netmf.com> (retrieved 12.09.2017)

¹⁶ <https://www.tinkerforge.com> (retrieved 12.09.2017)

¹⁷ <https://www.arduino.cc> (retrieved 12.09.2017)

¹⁸ <https://www.mysensors.org/> (retrieved 14.09.2017)

¹⁹ <http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01> (retrieved 14.09.2017)

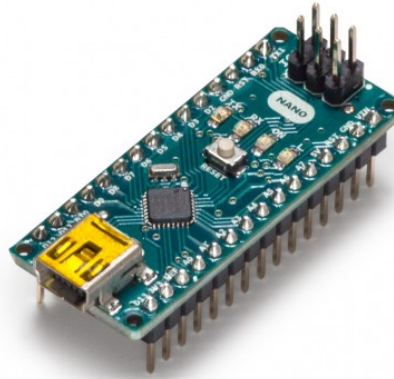


Figure 3.3 – Arduino Nano²⁰ which was used for the interactive prototypes.

3.3 User Research Methods

As already discussed, the general approach in this thesis is to generate knowledge through RtD and user feedback, which is then mainly to be analyzed qualitatively. In this section we detail specific methods that we employed for conducting user research methods.

To answer a research question qualitative (like interviews, transcription, focus group) or quantitative methods (like descriptive statistics) can be employed. According to Laurel (2003), qualitative methods play an important role in design research, and accordingly this thesis deals mainly with qualitative methods. Furthermore, Bryman (2012) believes that qualitative methods such as interviews can provide a deeper understanding of a situation compared to quantitative methods such as filling in a questionnaire. Therefore, he recommends qualitative methods to be used where either little knowledge of a situation exists or if insights from other participants are needed.

Data collection

We used qualitative methods like thinking aloud (Lewis 1982) in conjunction with user interviews and user observations to collect data. We gave our participants tasks which had to be solved. During testing, an audio recording was made (with their permission) and people were asked to think aloud what they were doing or thinking. During the discussions with the participants, we also followed the “interviewing rules of thumb” (Blomberg and Burrell 2009), which helped us to keep the interviews more interesting. For example, the authors advise that we should not “underestimate the value of casual conversation. Some of the most insightful information comes from informal conversations when social barriers are lowered” (Blomberg and Burrell 2009, p.78) or that we should “use lack of knowledge as a discovery tool. Participants will always know more about their own experiences than the interviewer will. In this context, don't interrupt unnecessarily, complete a participant's sentences, or answer the questions” (Blomberg and Burrell 2009, p.78). In short, the advise that we should learn about the participants' views and not ours. Furthermore, the duration of each task was stopped in

²⁰ <https://store.arduino.cc/arduino-nano> (retrieved 14.09.2017)

log files to subsequently have a comparison between the individual participants and tasks. To gather information and characteristics about the participants like age, gender, education and experience in programming or with home automations we used a questionnaire (see Findings Appendix for more information) and asked the participants to complete it.

Data analysis

We employed a general inductive approach for analyzing qualitative evaluation data (Thomas 2006) which basically means to

"condense extensive and varied raw text data into a brief, summary format; to establish clear links between the research objectives and the summary findings derived from the raw data and (3) to develop of model or theory about the underlying structure of experiences or processes which are evident in the raw data." Thomas (2006, p.237)

The aim of Thomas' approach is to identify specific characteristics from individual texts and finally to obtain a general view of these characteristics in an "inductive" way. We also used quantitative descriptive statistics to analyze the log files of the task durations. We then visualized these statistics with bar graphs and with statements from participants.

3.4 Research Sequence

At the beginning of this thesis, a workshop was held with experts in interaction and industrial design to form an idea. The primary goal of the workshop was to explore what kind of actions, reactions, and sequences etc. can be implemented by TOP (thingy oriented programming) in a purposeful way. It should be noted here that before another prototype called TOP (Güldenpfennig, Dudo et al. 2016) was developed and this should be "re-invented" now with – if necessary – completely new features. How exactly these functions should be implemented, we wanted to find out in the workshop. Subsequently, an intensive literature research and market research was conducted, using Google Scholar²¹ and ACM²² with keywords such as "Internet of Things", "prototyping with Arduino", "end user development" or "HCI research methods", to name just a few. With the collected knowledge a first paper prototype was developed and then tried out and discussed with five participants. This process was repeated until finally the final prototype was developed (see Table 3.1 for details). At all these stages, we recorded audio during our conversations with the participants, except at the workshop where we recorded everything in writing.

PHASE	TYPE / DESCRIPTION	METHOD
EXPERT WORKSHOP	Ideation	Workshop

²¹ <https://scholar.google.at/> (retrieved 23.09.2017)

²² <https://dl.acm.org/> (retrieved 23.09.2017)

PROTOTYPE I	<p>Ideas were collected with 4 experts to design a prototype</p> <p>Paper Prototype (lo-fi) Based on the workshop, a first paper prototype (made of wood) was developed and tested with some participants</p>	General Inductive Approach (Interviews, Transcription), Wizard of Oz
PROTOTYPE II	<p>Mid-fi Prototype Iterative process: after receiving feedback from the previous prototype, the prototype has been refined and made interactive and again evaluated</p>	General Inductive Approach (Interviews, Transcription), Thinking Aloud
PROTOTYPE II (IMPROVED)	<p>Mid-fi Prototype Iterative process: after receiving feedback from the previous prototype, the prototype has been refined and made interactive and again evaluated</p>	General Inductive Approach (Interviews, Transcription), Thinking Aloud
FINAL PROTOTYPE	<p>Mid-fi Prototype Finally, a fully interactive prototype (made of plastic) was built to explore our research question in an experiential way</p>	Thinking Aloud, General Inductive Approach (Interviews, Transcription), Log-File Analyze

Table 3.1 – Overview of all phases in which the research methods were applied

3.5 Participants

During our design process, we recruited 43 participants aged 17-60 and with different educational backgrounds. The expert and designer workshop took place at the HCI institute at the Vienna University of Technology. The evaluations took place either at the participant's home or at the HCI institute.

Before the evaluations and prototype deployments, we asked some students and people from our extended social networks if we could introduce them to our prototype and ask for their personal opinion (coded with A* and B*). In order to also involve experts with a background in "interaction design" in our evaluations, we invited one participant each (pre-)phase from the HCI Institute as they gave us valuable "expert" feedback to drive the research forward (A3, B0)

For the final evaluation, we presented our prototypes to 27 interested computer science students during the "Open Day" at the Vienna University of Technology and the evaluation took place in the rooms of the HCI institute. During the Open Day, 7 participants volunteered to participate in our test setup and fill in our questionnaire (which are coded with M*), to the other 20 participants our prototype was introduced and then we asked them to complete our questionnaire (coded with M8-27). Thanks to the questionnaire which we handed out to the participant, we noticed that these participants had technical experience. Therefore, an additional evaluation with three older participants took place in their house (in-situ), which we classified as technically inexperienced (coded with N*).

Note: AHS, HAK and HTL are educational institutions with a high school diploma.

PHASE	PARTICIPANT ²³	AGE	GENDER	HIGHEST DEGREE
EXP. WORKSHOP	W1	41	male	postdoc in interaction design
EXP. WORKSHOP	W2	30	male	predoc in interaction design
EXP. WORKSHOP	W3	36	female	postdoc in interaction design
EXP. WORKSHOP	W4	29	female	predoc in interaction design
PROTOTYPE I	A0	22	female	AHS
PROTOTYPE I	A1	25	female	AHS
PROTOTYPE I	A2	33	male	HTL
PROTOTYPE I	A3	29	female	predoc in interaction design
PROTOTYPE I	A4	30	male	AHS
PROTOTYPE II	B0	29	female	predoc in interaction design
PROTOTYPE II	B1	28	female	HAK
PROTOTYPE II	B2	33	male	AHS
PROTOTYPE II	B3	37	male	apprenticeship
FINAL PHASE I	M1	24	female	AHS
FINAL PHASE I	M2	20	male	HAK
FINAL PHASE I	M3	19	male	HAK
FINAL PHASE I	M4	27	male	AHS

²³ To ensure anonymity, the names were coded.

FINAL PHASE I	M5	32	female	Graduate
FINAL PHASE I	M6	19	male	AHS
FINAL PHASE I	M7	20	male	AHS
FINAL PHASE I	M8	21	male	AHS
FINAL PHASE I	M9	20	male	HTL
FINAL PHASE I	M10	18	female	AHS
FINAL PHASE I	M11	17	female	AHS
FINAL PHASE I	M12	22	female	AHS
FINAL PHASE I	M13	24	female	AHS
FINAL PHASE I	M14	22	male	HTL
FINAL PHASE I	M15	23	female	AHS
FINAL PHASE I	M16	19	male	AHS
FINAL PHASE I	M17	20	female	HTL
FINAL PHASE I	M18	24	male	HTL
FINAL PHASE I	M19	20	male	HTL
FINAL PHASE I	M20	20	male	AHS
FINAL PHASE I	M21	21	male	AHS
FINAL PHASE I	M22	25	male	HTL
FINAL PHASE I	M23	23	female	HTL
FINAL PHASE I	M24	20	male	HTL
FINAL PHASE I	M25	22	male	HTL
FINAL PHASE I	M26	27	male	HTL
FINAL PHASE I	M27	34	male	AHS
FINAL PHASE II	N1	56	male	HTL
FINAL PHASE II	N2	60	female	AHS
FINAL PHASE II	N3	59	male	AHS

Table 3.2 Demographic data of all involved participants

3.6 Summary

In this chapter we have presented the methods and approaches, which we used in this thesis. We dealt with the Research through Design philosophy (Zimmerman, Forlizzi et al. 2007) which proposes to generate knowledge through engaging in design activities. We have learned that this approach is still rather new to HCI and that the process should not be too rigid in order to preserve the strengths of design. In general RtD allows us to gain insights into and through the process of creating prototypes. To answer a specific research question, for example, designers address this question in their design proposals. This process will be iterative in most cases, in which the designers repeatedly asks a few people and refines the prototype more and more. We used the thinking aloud method (Lewis 1982) for collecting data. For the analysis we used the general inductive approach for analyzing qualitative evaluation data (Thomas 2006).

4 Iterative Design Process

In this chapter, we report all design activities conducted as part of this thesis. This includes a design workshop, a paper prototype, a mid-fi prototype, and the final prototype. All prototype descriptions are followed by a short summary of their evaluation, except the final prototype. The assessment of the final system is reported in a separate chapter (the findings section) as it constitutes the primary user study of this master thesis.

In addition to the prototypes (paper, mid-fi and final), we present an earlier prototype (Güldenpfennig, Dudo et al. 2016), which informed the design of our endeavor, but which is not part of the thesis. We include this report here (and not in the related work section) as to support the readers' understanding, and because it is a prototype of our own that we created before the master thesis.

4.1 Overview

With the combined knowledge from the literature search and market research, which was summarized by us in the "related work" chapter, we started to build our first prototypes including a design workshop. The initial versions were so-called lo-fi prototypes, which means that the prototype had no technical functions. These prototypes were used to get first experiences, opinions and feedback from the participants or experts, such as, what their first impression with the prototype is, whether the concept of our project is clear to them, what they would like to improve or would make better.

The approach of this thesis has an iterative design character, which started with a design workshop for ideation. By *iteratively* (see also Figure 4.1) we mean, after a prototype was built, it was immediately tested with a few participants (in the form of interviews involving a "Wizard of Oz" technique) in order to subsequently refine the concept and build an improved prototype drawing on the feedback from the participants. In the next iteration this was then tested again with more people and so on. Schön (1984) denotes this cyclical procedure as a "reflective conversation with the material" (this are in our project electronic components, form factors, various materials e.g. plastic or wood, etc.), which was decisive for the knowledge construction process.

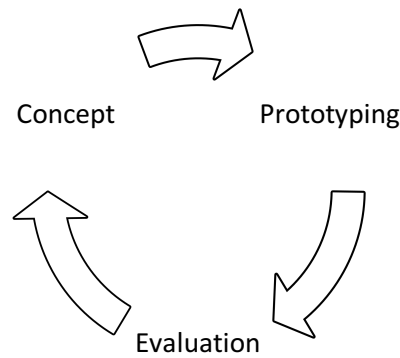


Figure 4.1 – Presentation of the iterative design process. After conception, a prototype will be built which will be tested with participants, thus gaining new insights leading to a new or improved concept. After this process has been repeated several times, the process ends with a finalized evaluation followed by the conclusion (cf. Figure 3.1).

Subsequent to the TEI short paper (Güldenpfennig, Dudo et al. 2016) and the implementation of the proof-of-concept demonstrator (not part of this thesis; see below), we held a design workshop to advance the TOP (thingy oriented programming) concept. During the design workshop we had interesting discussions and addressed fundamental design challenges as well as technical questions. Questions such as how to start the programming or learning process of the system or how to design the form factor of the sensor or actor modules.

As next step, we considered a few meaningful scenarios and textually implement these as lo-fi prototypes. We built a paper prototype, drawing on the feedback from the expert workshop and from the collected knowledge from the literature review. In the lo-fi phase, we worked with the *Wizard of Oz* method, which means that the participants are made believe that they are working with an interactive prototype, but in reality we have simulated the reactions from the prototype (Hanington and Martin 2012). In every prototyping step or iteration we involved some potential end users in the design process to get some feedback. Finally, we focused on a number of TOP features and implemented a hi-fi prototype (with respect to user interaction) of TOP and deployed it in a field study so that we could address our research questions. In the Figure 4.2 below, we present our design process in a simplified and chronological way with all major phases.

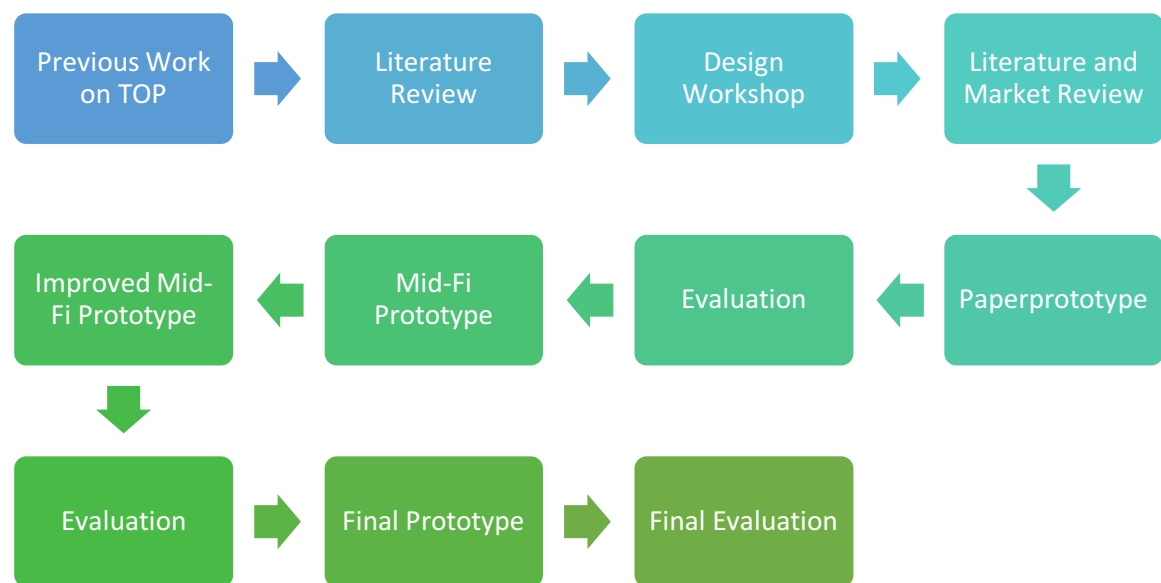


Figure 4.2 – Chronological time line of this thesis with major design phases.

As already described above, after every completed prototype we tested it with participants to gain feedback for improvements. We aimed at asking at least three or four different people about our prototype, except for the final prototype where we included 10 participants into the final evaluation. Table 4.1 displays how many attendees were involved in each phase.

DESIGN PHASE	NUMBER OF PARTICIPANTS
Design Workshop	4
Paper Prototype	5
Mid-fi Prototype	4
Final Prototype	7 (technology-affine) + 3 (non-technology-affine)

Table 4.1 – Overview of the 25 participants of each phase in this thesis.

4.2 Previous Work

Before we discuss the design workshop, it should first be clarified what had happened before the workshop. The reason for this is that everything that took place in the workshop and also in this thesis is based on work done previously.

Before the workshop, we started an experimental and alternative approach to prototyping simple electronics applications and systems that involve networks of sensors and actuators. We have implemented an interactive prototype early in the design process to clear questions around technical feasibilities and to probe some initial reactions from potential users. This demonstrator enabled the users to define or “program” wirelessly connected objects. We have published these preliminary results in the renown TEI conference in 2016 (Güldenpfennig, Dudo, and Purgathofer 2016).

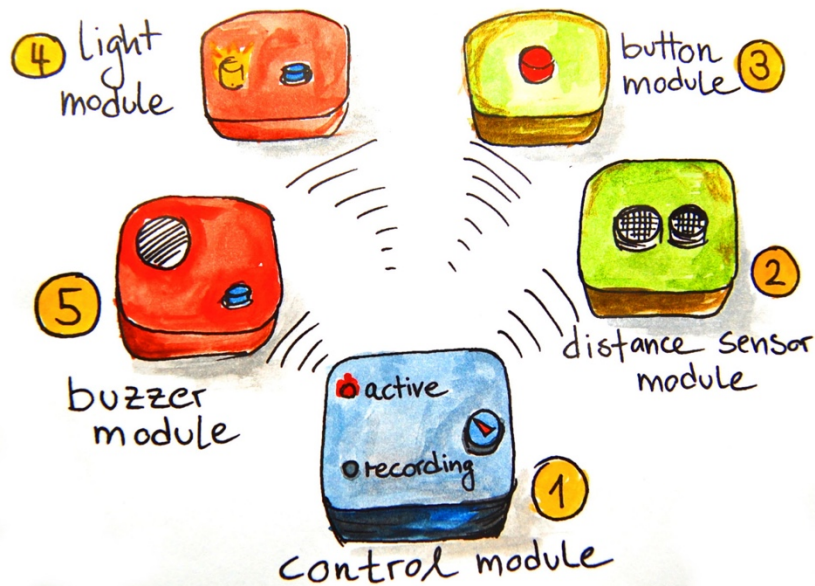


Figure 4.3 – Illustration of "Play the Record" (internal/unofficial prototype name) which enabled to connect sensors and actuators together in a simplified fashion. Using the buttons on the modules, the user was able to select the sensors and actuators they wanted to use. Some sensor modules (e.g. distance sensor) had no extra button, instead, these sensors were detected by the system after an event was triggered. Note, there are fundamental differences between "Play the Record" and the design concepts as implemented in this master thesis.

Figure 4.3 shows how the very first idea of TOP was considered (to avoid misunderstandings, we will use "TOP" as a generic term for "Play the Record" and this thesis. This means, that "Play the Record" stands explicitly for everything that has happened before this thesis and "TOP" represents the evolved state of "Play the Record" respectively this thesis). In this version of TOP, the programming is based on the recording of sequences of actions and reactions using tangible objects. The tangible objects which we provided were one control module as well as multiple sensor and actuator modules. As actuators, a buzzer and a remote socket (as RF transmitter) were implemented, and as sensors there were a motion sensor and a distance sensor, in addition, all modules were equipped with an RGB LED. With the control module the user could change all other modules into two states: *active/play* or *recording*. In record mode, the user could record a sequence which modules the user wanted to use by pressing the button on the module. Its underlying concept is to connect actions with reactions in analogy to recording macros in software packages such as *Microsoft Excel*. That is, the user records a procedure or sequence of interactions by setting the system into record mode and triggering all events that should be part of the procedure manually. Later, in play mode, the record of this procedure can be replicated automatically. This enables the user, for example, to associate the triggering of a pushbutton on device A with the buzzing of a noise generator on device B. The current status was recognized by the color of the LED. In record mode it was blue, in play mode it was green. In addition, the user was still informed whether the module is connected to the control unit, otherwise the LED has light red. If the module was used in record mode, the user was informed by a yellow LED. For more details see (Güldenpfennig, Dudo et al. 2016).

The prototype was built for two reasons. On the one hand, we wanted to test whether our project is technically solvable, and in addition, it was investigated if we could find an affordable solution. On the other hand, we wanted to test our project with a number of participants, so we get a first feedback on this project. For the technical implementation we have used Arduinos, electronic components and standard 9V batteries. For communication between the modules, we chose NRF24L02 because they are easy to use, very cheap, small and have low power consumption.

While this work provided answers to our research question, it also raised new questions that eventually motivated the:

“what kind of interactions can be programmed using a system like TOP? Which system logic can be implemented using TOP? What kind of sensors and actuators should be incorporated? What affordances should we design to make programming intuitive and how can the users learn about the functions of the different modules? Should we colour all sensors green and actuators red, or should we employ some sort of lock-and-key icons to indicate how the modules work together?” (Güldenpfennig, Dudo et al. 2016, p.6)

Note, the approach to addressing these questions in this thesis is fundamentally different from “Play the Record”.

Among other things, we wanted to discuss these questions with a few experts by means of a workshop.

4.3 Design Workshop

By means of a workshop, we wanted to clarify basic questions to find an answer on how to proceed and also get some tips on design from experts who were knowledgeable in the field of interaction design.

The primary goal of the workshop was to explore what kind of actions, reactions, and sequences etc. can be implemented by a technology like TOP (tangible programming without coding) in a purposeful way. We were particularly interested in the following questions and therefore we tried to find answers in the workshop because we felt that they could be groundbreaking for this thesis:

- **Actions:** Which possible actions are favored by the experts?
- **Reactions:** Which reactions actions are favored by the experts?
- **Programming in TOP:**
 - How should the end-user define interactions, that is how should they record macros?
 - How simple/complex should this be? What metaphors etc.?
 - How to start the learning process of the system? (e.g., use a “magic wand” for linking reactions to actions)
 - What kind of feedback, where and when?
- **Feedback:** What feedback mechanism are necessary?

- **Form factor/design:** *How should the system physically be built? What kind of mounts/stands are needed?*
- *Optional: What kind of use cases for TOP can the experts envision?*

The workshop (see Figure 4.4) had a duration of 2 hours and the following attendees participated (we don't present names for anonymization):

- W1: postdoc in interaction design (background in computer science; experience in industry and academia)
- W2: predoc in interaction design (background in computer science; experience in industry and academia)
- W3: postdoc in interaction design (background in industrial design; experience in industry and academia)
- W4: predoc in interaction design (background in industrial design; experience in industry and academia)

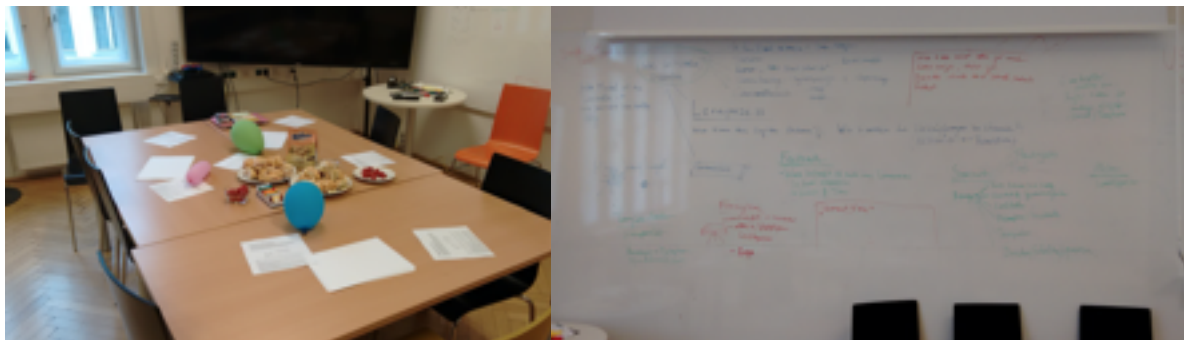


Figure 4.4 - Pictures from the workshop. Right the workspace and left some collected ideas on a whiteboard

To find answers to these questions, we came up with several tasks or activities (see Table 2 for details). The participants were asked to work in two-parted groups. The first task served to introduce the problem. We tried to explain the problem to the participants in a simplified and illustrative way by using "agents" who are "stupid" and unmotivated at first (see Figure 4.5).



Figure 4.5 – Explanation with illustrative agents: the agents represent small parts of a technical system. Each agent has a special ability (“dudette with stethoscope” = microphone, “screaming dude” = making noise, “flashlight dudette” = lamp, “dude with a spyglass” = detecting movement (from left to right)) (CC BY 3.0 AT Dooder 2017). The challenge for the workshop participants was: how do we teach the agent when something happens, that they respond in a certain way?

By default, the agents do not do anything. The task now was, how can we teach these agents to “cooperate” with each other to respond to issues as described above (Figure 4.5)?

In the second task, we asked the participants to imagine the following scenario and then come up with a solution as a group: “one agent detects movement, the other then turns on the light. How do we tell the agent to watch if another agent has noticed movement and then do something in response (e.g., turn on the light) without coding?” The other activities were similar but became more complex. The exact sequence of the workshop including all tasks is attached. In this partially playful way we wanted to engage the participants in open-minded discussions. Table 4.2 summarizes the course of the workshop.

Activity	Duration (minutes)
Introduction	5
Two-part group task (explanation + doing)	5+15
Discussion	15
Videos	10
Tasks (explanation + doing)	5+15
Form factor/design	15
Final discussion	As needed – about 30 minutes left

Table 4.2 – Workshop-Agenda

These tasks resulted in an interesting and critical discussion. We were able to gain the following insights from the workshop, which we will summarize here as short “design briefs”. Also, we will state which impact the insights had on our design process:

Insight and impulse: We can't accomplish everything within one product/prototype. We should do less but better, which means to focus. Kids, for example, get frustrated easily if the product is not perfect. The product should be robust.

Impact: We followed this idea and focused on just a few features instead of covering as many as possible. For our prototype, we used certain "action tokens" (see below for more details), which would allow us to offer many features, but we've only implemented basic functions in the demonstrator. As a side-effect, this smaller number of features made TOP a robust prototype with few bugs.

Insight and impulse: To little surprise, the product/industrial designers in the workshop seemed to be more product oriented. That is, they wanted to create a cool gadget or product or user experience. They also had lot's of disruptive, abstract, and not very practical ideas (on the short-term). For instance, they suggested that TOP components could come in the shape of furniture. Sitting on a chair could start the learning mode, and so on.

Impact: We personally think these ideas are a bit complicated and are not straight forward, however, useful for generating novel ideas. Still, we must remember that the product needs to be able to build within the course of a master thesis eventually. We wanted to take on the idea to have a smart and aesthetically pleasing vase to be part of the TOP system. In the end, we agreed that we should disregard the idea about aesthetics (e.g. vase) and instead focus on how the participants interact with TOP. That means, we aimed at implementing a sound user experience, but we were not attempting to address particular aesthetical elements.

Insight and impulse: *The unique selling point (USP) of TOP should be worked out as much as possible.*

What is this project actually aiming to accomplish?

Can the TOP prototype be also a toy, or whom is it for? Is it about kids learning to program? Is it about programming the smart home, the Internet of Things?

In order to strengthen USP: "Aim not to be SAM"; aim not to be a phone app extension etc. TOP is about tangibility. It should not involve an app etc. As little conventional computer stuff as possible.

Impact: These considerations ultimately led us to avoid using a display on the controller.

Furthermore, we learned with respect to the challenge of *initializing the learning and programming process* that "everything" could be considered as a button as a thought experiment to inform interaction design. By "everything" the focus group meant that any object could be used as a button in tangible programming. For example, "pressing a button" could be analogous to "touching any surface" (table, chair, window glass, etc.).

Some of the ideas *about the programming process* were that taking an object in the hand and putting it back again, could program the devices, and so on. The focus group spent a lot of thought on how to remove additional hardware. They came to the idea that each object could be a controller, so an additional controller wouldn't be needed for programming. Interestingly, we learned that industrial designers distinguish between *object-related* and *body-related* activities. That mean, that TOP could be programmed with the help of an object (e.g., a remote control or some sort of magic wand) or by a human actor using his or her body to program the system (natural user interaction).

We unfortunately didn't learn much about possible form factors for TOP as well as about the boundaries of tangible programming. We could only take forward that probably not too much complexity can be mapped to a tangible prototype and metaphors and analogies have limitations.

In summary, we decided that programming smart objects (e.g., as part of Internet of Things applications) should be without conventional computers or mobile phones/tables etc., but through physical interaction with the objects and that we should choose a target group (e.g. children, older people, non-technology-affine people, etc.) and focus on a few features, but then implement them as error-free as possible.

4.4 First Prototype: Lo-fi and Made from Wood

To realize our project, we focused on the controller after the design workshop. The aim of the first prototype was to investigate how the user can interact with a *controller* to program modules like sensors and actuators. For this, a lo-fi prototype was made and then discussed with participants.

4.4.1 Prototype Concept

For our first prototype, we used the *paper prototyping* method in the design process to explore the users' needs, which is a common method of lo-fi prototyping. The aesthetics were not of primary interest at this state; it was much more important to discuss our project with a few people and then incorporating their feedback in the next prototype. It should also be mentioned that the design of the actuators and sensors were not of primary interest either. Rather, we wanted to ensure that the interaction with the "controller" to program actuators and sensors is understandable.

To this end, we considered how we can program everyday IoT tasks without a smartphone or computer, since there are already many app-solutions commercially available and our focus was on tangible computing. Our core idea for this first prototype was to offer our "links" (these are logical operators like "and", "or", "smaller" but also functions such as "time functions", "if an event was detected", "triggered" – later we called these operators and functions "actions") in the shape of bits comparable to removable screwdrivers for an electric drill.

In the beginning, our intention was to offer tangible objects for the user, where each object serves as a "link" and had a special function. It was our idea that the user gets a set of links provided and can assign specific functions to each actuator or sensor by taking a link and inserting it into the controller. Then the user would have to hold the controller to a sensor or actuator, with which the user would have programmed a function to the module. In addition, the controller had a display to let the user know what is happening, so the controller would inform the user which function he had assigned to an actuator or sensor or what behavior the user is currently programming. Furthermore, the controller had buttons to accept or reject functions and potentiometers to enter numerical values, that is, to increase or decrease a numerical value.

With a simple example we want to explain this programming process in detail. Let us assume that a user wants to turn on a lamp when a pushbutton has been pressed. To this end, the user would have to take the link "active" and insert it into the controller, which has the function to trigger the system which then knows that the button was just pressed (active). Afterwards, the user holds the controller next to the pushbutton and is informed on the display that he assigned the function "active" to the pushbutton. This link can now either be accepted or rejected by the user pressing one of the buttons on the controller. Then she/he leaves the "active" link inside in the controller, and also inserts the "on" link into the other side of the controller, so that the system "knows" that when one sensor is active, the lamp should be switched on (the reason for the wording "one sensor" is that at the beginning we thought that if several sensors were used, they would be linked together with "or" and thus one sensor would be enough to

turn on the lamp. In contrast, all actuators would be linked with "and", that is, if a sensor is active, all used actuators will be activated). Finally, the user holds the controller next to the lamp to finalize the programming of this lamp. Eventually, the user would be informed on the display that the lamp should turn on as soon as a sensor is active, which the user can either accept or reject.

4.4.2 Prototyping Process and Functions

In the beginning we sketched our controller prototype as a form of a wrench (see Figure 4.6), because we thought that a wrench is a useful craft tool that increases efficiency and productivity at work. We split the wrench into two halves. One half was intended for reactions and the other for actions. Actions can be, for example, the following events: something has been detected in the form of movement or someone holding an RFID chip next to the sensor. But also logical connectives such as comparisons or conjunctions, and functions such as resetting or time programming. A Reaction is then the result of a previous action, virtually a response. If, for example, a defined threshold value (greater operator) has been exceeded, as a reaction follows that the actuator is to be activated. Reactions can therefore be on or off commands, but also time-shifted on and off (e.g. turning on a device for 10 seconds). The drive profiles (action and reaction) of the wrench are exchangeable. The user should therefore select which action or reaction he needs for his programming.

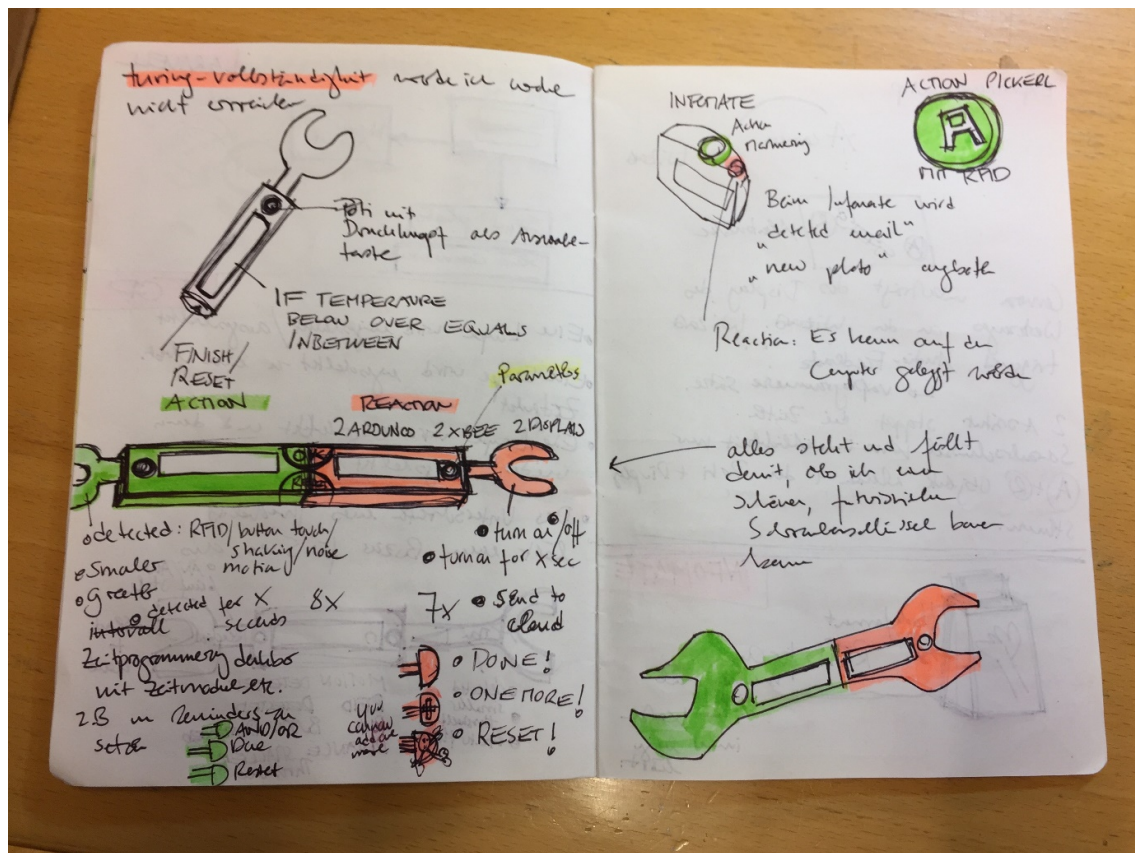


Figure 4.6 – Some sketches for the first prototype. The initial idea was to design the prototype as a form of a wrench.

Later in the process, we discussed which links for action or reaction are most likely to be used to reduce them to a certain number (cf. workshop “the product should be robust” and “focus on the USP”). At the beginning we set the number to 7 links: and, or, greater, smaller, not, triggered, time. We have investigated basic applications or use cases and which links are needed in each case. Table 4.3 and Table 4.4 are intended to provide an overview of all links which we have finally selected.

Action	Meaning	Example
<i>Greater</i>	If the actual value is greater than a threshold	<ul style="list-style-type: none"> • If the temperature is greater than 20°C • If the relative humidity is more than 60% • If it is brighter than measured now
<i>Smaller</i>	If the actual value is less than a threshold	<ul style="list-style-type: none"> • If the temperature is smaller than 20°C • If the relative humidity is lesser than 60% • If it is darker than measured now
<i>And</i>	Linking two or more sensors. Only valid if all sensors have triggered something	If the temperature is less than 23°C and the relative humidity is greater than 55%, then activate the actuator
<i>Or</i>	At least one of the sensors was triggered/gave alarm	If the temperature is greater than 20°C or a button is pressed, activate the actuator
<i>Not (renamed to „not active“)</i>	Invert / Negation (Opposite)	<ul style="list-style-type: none"> • If no movement has been detected, activate the actuator • If no key is pressed, activate the actuator • Invert the active state (turn off something that is currently on)
<i>Triggered (renamed to “active“)</i>	When the sensor has detected something (e.g., movement, sound, key pressed)	<ul style="list-style-type: none"> • If a button has been pressed, activate the actuator • If a movement has been detected, activate the actuator • If a noise has been detected, activate the actuator
<i>Time</i>	A sensor has triggered something for a certain time	<ul style="list-style-type: none"> • If a button has been pressed for 5 sec, activate the actuator • If the rel. Humidity is longer than 15min over 55%, activate the actuator

Table 4.3 - Overview of all necessary actions to cover as many applications as possible

Reaction	Meaning	Example
<i>On</i>	Activate an actuator	If something has been triggered, the actuator is activated immediately
<i>Off</i>	Turn an actuator off	If something has been triggered, the actuator is turned off immediately
<i>Time</i>	Turn an actuator for a certain time on or activate it after a certain time or activate it at a specific time	<ul style="list-style-type: none"> • Activate the actuator for 15 sec. (Note: Time module + On module) • Activate the actuator after 11sec. (Note: Time module + Larger + On module)
<i>Send to Cloud</i>	If something has been triggered, send it to the cloud	If something has been triggered, send this event directly to the cloud

Table 4.4 - Again an overview of all the reactions which were defined

With these considerations, we then built our first prototype. The first prototype was cut out of wood and plastic using a laser cutting machine, which can be found at the Vienna University of Technology in the HCI institute, as it was a very quick way to explain our idea to our participants. Since the laser cutter machine works with vector graphics, we created our prototype in *Adobe Illustrator* because the units of measurement were the most consistent with the machine (i.e. 1 cm length in the program was cut identically).

We did not split the actions and reactions as in Figure 4.6, so the user was forced to put the links into the left side for actions and for the reactions on the right side. Instead, it did not matter which side the user chose for actions or reactions. As evident from Figure 4.7, the controller had a rectangular shape, as it was much easier to design. We used three layers for this, where the middle layer on the left and right sides each offered an opportunity to insert the links. On the top layer there were two buttons for accepting or discarding the programmed link, in the middle was a plastic frame, where we could insert our paper cards that represented the display. By means of the paper cards, we were able to show the user that the text on the display changes with each action (e.g. link inserted or controller held next to a sensor). The links were also cut out of wood and labeled by us. On one side, the links had a larger rounding, so that the links could only be plugged into the controller in a certain direction.

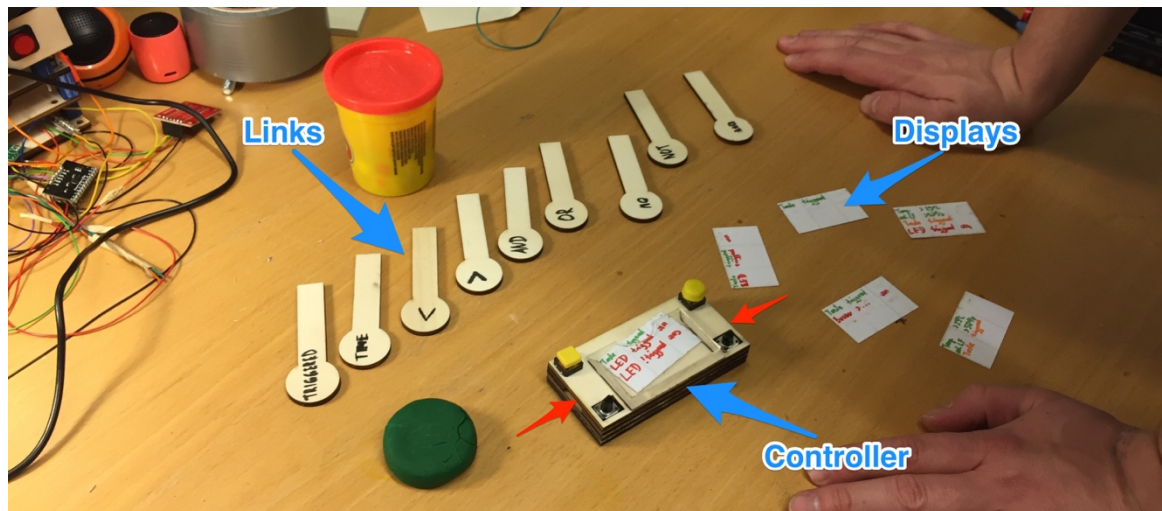


Figure 4.7 - The first lo-fi prototype. The “lollipop-looking” wooden elements are the links, the rectangular wooden element is the controller and the small paper cards represent the display. The red arrows show where the links could be put in. In addition, Play-Doh should represent a sensor or an actuator.

4.4.3 Evaluation and Insights

By means of the prototype, the discussions went beyond theorizing and allowed us to observe first user experiences in the field. A challenge for us was how to explain the logical connectives “and” and “or” comprehensible to people who are not tech-affine. For technicians or researchers in the field of technology, the logical connective may be self-explanatory, as they are constantly confronted with it, but for non-tech-affine people, or rather, people who were hardly involved in engineering and have no relation to technology, this is not common. Therefore, we have chosen not to use these two links when testing the prototype with the participants. Instead we have specified that all sensors are linked by default with “or” and actuators with “and”. If one of the test subjects had asked how we intend to solve the logical connectives, we would have explained to them that we had thought about it and originally we had it as links as part of the set, but we have taken them out for simplicity.

Participants and procedure: The evaluation took place in a quiet and separate room with the door closed, where we presented the prototype with all its components on a table. The entire process of the evaluation was a face to face discussion with the participant (see section 3.5). We did not prepare any specific questions for the discussion, much more we asked the participants what they liked or where they were confused, especially during we demonstrated our prototype. Thereby, an interesting discussion emerged around these “simple” questions with the participant, giving us valuable feedback from the participants for the ongoing design process. To explain our prototype to the participants (see Table 4.5), we briefly introduced our motivations in this thesis and then explained our prototype with two examples in the context of smart home applications. In the beginning, we decided to show all the links, but after observing that the “large” number of links confused the participants, after the second round of testing, we decided to put only the links on the table which were necessary for the respective task. As a result, participants focused more on interacting with the prototype rather than worrying about which features the links have and why they are on

the table. All participants (coded as A0-A4) had an idea about using technical devices (such as smartphones) and also one participant (A3) had experience in product design. Concretely one participant was from the HCI institute (A3), two participants were students in the field of media informatics (A0, A1) and the other two participants were technically enthusiastic, but had no technical education (A3, A4).

PARTICIPANT ²⁴	AGE	GENDER
A0	22	female
A1	25	female
A2	33	male
A3	29	female
A4	30	male

Table 4.5 - Overview which attendees participated in the first test run.

The first example was a basic “lights on and off” with a push button setup, the second example was to program a timer, so that after a certain time the buzzer should make noise. After demonstrating the prototype to the participants, the participants were asked if something was incomprehensible or whether they were confused for some reason. The participants were then asked to replicate the two examples on their own. In doing so, we were able to observe if they were able to solve the same problems which we explained before and at what stage there were complications or what aspects they did not understand. They were politely asked to say aloud every step they intended to take (see Figure 4.8). Finally, we had a short discussion about our prototype to get more feedback.

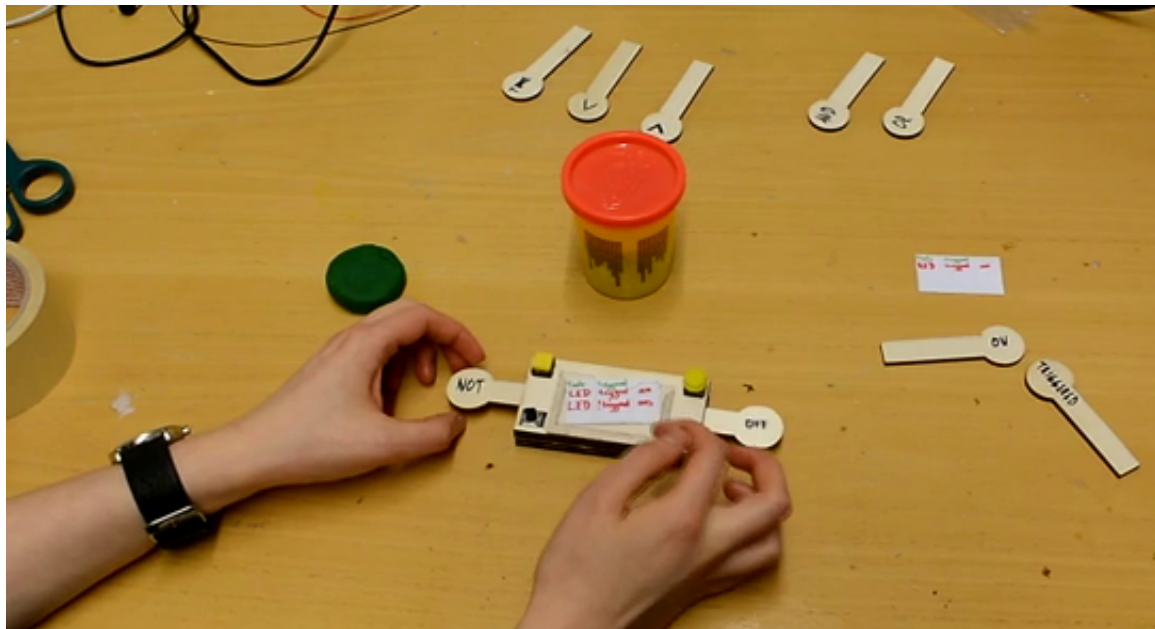


Figure 4.8 - Snapshot where a participant interacts with the prototype.

The tasks were not always clear to all people immediately. As long as we showed the attendees how to connect actuators to sensors, the attendees could follow us. But as soon as they were asked to do their own linking between sensors and actuators, we

²⁴ To ensure anonymity, the names were coded.

quickly realized that people felt insecure some steps. This does not necessarily mean that the process is incomprehensible, but this could be due to the fact that the system was not interactive enough. We were able to observe that if the participants had the opportunity to try it out by means of trial and error, they would learn the process themselves. Although we always tried to simulate the system interactively at every step taken ("Wizard of Oz") by the participants, we still thought something like a "guide" was missing. This was also our first thought during the conceptual design phase and that's why we were pretty much fixated on giving the controller a display so people could go back to some point of reference to see what they were doing or what they should do.

We also found out in the discussions with the participants that most participants (A0, A1, A3, A4) have emphasized that they prefer it if the order was not important, for example, A0 said: *"I think that will also invite a lot of experimentation, if you do not have to know in which order you must doing now."*

A1 wanted to advise us that suggestions are especially important for people who have no idea about programming: *"I find suggestions extremely important. Because my experience was in programming, you were lost without any suggestions. If you do not know what you can do, you will find it hard to figure out what to do."* Attendee A3 also said that the display should show what can be done: *"It could also be a little smart, so you could avoid this plug in and plug out, for example, if you go to the lamp, then you can only switch it on and off and you select directly on the display, which state you want to have instead of something to plug in or to plug out. So you can take the work out of people's hands."* A1 made us consider keeping every interaction minimal and make elements "automatic" instead of demanding the user to set everything on their own. Of course it depends on the target group.

A3 told us that this prototype was already too concrete in her opinion. We should *"focus on one scenario at the beginning, for example, how do I connect a pushbutton with a lamp instead of providing a whole set of functions."* A3 suggested that we should only try this one scenario with several people. *"The prototype may also be a little interactive so people can better understand what the process is"*, but we should not put too much effort into it. Still, A0 stated that the lamp should really start to light when the user presses the pushbutton. It was also mentioned by A0 that we should generally not offer too many loose links, because it increases the perception that there are many commands, and because *"such small parts like to get lost or someone is nasty and hides the off-link, then you can turn everything on, but not off again."* It was also mentioned that too many links might take the user a long time to find the function they are actually looking for.

The analogy with the drill (cf. Prototype Development) did not appeal to participant A1, because the participant had the opinion that *"a bit is necessary in the drill as the motor drives the bit. On a drill, you change a bit and then work with it, but your prototype always has something to put in or out. And considering that I have no programming experience and I have to deal with the controller even more, it might be inconvenient for me"*. Thus, for most participants (A0, A1, A2, A3), the "putting the bits (links) in and back out again" was thought to be annoying and cumbersome given the many

interactions with the controller. As a suggestion for improvement, we were advised that it should be enough to put the link on the controller instead of putting it in and out. Alternatively, we could also hold the links directly next to an actuator or a sensor, so for example, if we want to turn on the buzzer in x seconds, then we hold directly the “time-link” next to the buzzer. Hence, most of them (A0, A1, A2) were enthusiastic about the *modularity*, but not in this precise form, as it is far too laborious. The minimal setup was also much-praised by A0, given that the controller was equipped only with two buttons and potentiometers.

Interestingly, we were asked by each participant *“which target group do you want to cater? What do you want with it (the prototype)?”*. Participant A0 claimed that if we aimed at a specific target group, we could then reduce many functions, specifications, requirements, and design decisions to this one target group. And in hindsight, we could see if it is usable for other target groups as well. This should greatly facilitate our design process. A3 wanted to inform us with respect to the target group, that *“your target group are people with an interest in technology, but without a training in technology or computers.”*

It was also mentioned several times that it could be a nice toy for kids and we should also consider evaluating the finished prototype with children. For instance, A0 told us that *“especially for children this would be a lot of bauble stuff. Not that I want now that there are children constantly changing things in my apartment, but I guess this could be a nice toy too”*. Participant A2 stated: *“I think this plug in and plug out is for children really funny. The children could get to know programming procedures like this. It's just playful and it also has this programming logic. Now it depends on what the target group is. When it comes to playfully understanding what programming is like, it's cool. I find it visualizes the programming flows quite nicely.”*

A3 said that there were already home automation solutions on the market comparable to a bus system. This was also not complicated to use, if one wanted to “program” something with it, there was even a manual available, where corresponding programming steps were explained. A3 suggested implementing a similar manual using our display. The advantage of our system was according to A3 that the manufacturer does not prescribe which elements can be linked, because our system was with IoT much more powerful and generic. He also critically asked us: *“how many people really want to deal with a problem themselves and find a solution themselves? Are you sure that a user programmed his light on the toilet how long it should to light? Does not the user usually say to his electrician, “I want to the duration of the light on the toilet longer”?”*

Participant A2 claimed that the display is a kind of “pain point” for people who are not technically-informed and said that the quality of interactions depends very much on what appears on the display. Because with the right “instructions” on the display, we could guide the people. Therefore, he mentioned as an example, if on the display appeared that *“you have just linked the push button, please go to the next object”* or at least it should be on the display *“wait for next target object”* or something similar of this kind, since we (the user) do not need the display while we are walking to the next object”.

In his opinion, this could be used as a guide function ("a tutorial"), and further he meant *"when we are at the next object, we can use the display again as an information, thus which functions the object has available."*

In summary, the "plug in and plug out" mechanism was found by most of the participants to be tedious and, in addition, the procedure of which link the user should choose as a next step was not always clear. The attendees reminded us that if someone has no programming experience, hardly any user will be concerned with "how the prototype should work" (mainly because the user is not really supported enough by the controller with our prototype). Each participant emphasized that this would depend heavily on the target group. The process now was more like visualizing in a playful way "programming processes", which could be particularly useful to children. But in order to configure their own home automation in this way, most of the participants were rather skeptical (note: people were prejudiced with *home automation* because we explained our examples in this way). We were told, that the display is an important area, which allows us to guide the user. In addition, the display should inform the user what is possible with the selected module instead of the user having to independently select which linker the user needs. Therefore, in the next step, we will focus more on how we can support the user with our prototype. Hence, "the controller should support the user and not the user the controller".

4.5 Second Prototype: Mid-Fi Prototype with Interactivity

In this section we want to briefly explain how we developed the next prototype. First, we recapitulate the feedback about the previous prototype and subsequently start developing. This time, we designed the prototype with implemented interactivity, a so-called *mid-fi prototype*, but did not deeply care about design issues like form factor, as we were still most concerned about the interaction with the controller. To test the interaction, we invited a few people and discussed our mid-fi prototype with them. During the development process we iterated the mid-fi prototype and implemented some improvements. The improved version was identical to his predecessor with respect to its functions, but the size of the prototype was shrunk. During the evaluations, both prototypes were presented to the participants. The participants were allowed to decide for themselves which prototypes they wanted to use, as the functionality was identical.

4.5.1 Prototype Development

In the previous evaluation (cf. 4.4.3) of our first prototype we got a lot of feedback. We tried to consider as much feedback as possible in the design of the next prototype. To aid the readers' understanding, we want to recapitulate the most important participant comments up front:

- *The display should work as a guide and assist the user in what should be done next.*
- *The controller should be smart, for example, the display should suggest which configurations are possible with the selected actuator or sensor.*
- *The user should not have to “plug in or plug out” elements into the controller (this could take over the display).*
- *To make a task as interactive as possible, so that a sensor and actuator is technically working and really reacts.*
- *We should choose a specific target group and if possible fulfill their needs and requirements.*

Now we will briefly explain what we have improved, which important design decisions we made, how the prototype was developed and what difficulties or challenges we encountered. In order to avoid using the links in the next prototype, we instead opted for our prototype to have a touch screen. We thought that we would now represent the links on the display and that the user could then select them on the display, rather than using tangible objects as links. We also employed the display to let the display tell the user what the user has just done and what the user should do next in the form of text messages. We now work object-related, that is, when the user holds the controller to an object (sensor or actuator), the user gets notified on the display, which functions this object offers.

We will explain the process again with a simple example. The basic concept can be imagined as follows. Metaphorically speaking, the user holds a controller in their hand that “enchants” the sensors and actuators so that they can communicate with each other. For example, if the user wants to turn on a lamp when a (certain) switch is pressed, they hold the controller next to the switch and then go to the lamp and hold the controller next to the lamp. Now these two elements would be connected with each other, hence if someone pushes the switch, the lamp would turn on.

In this example, there would appear a problem nevertheless. The lamp would only go on when the switch is pressed and would go out again when the button is no longer pressed (except when a toggle switch is used). There is, for example, no possibility to let the light stay on for a certain duration. Therefore, the basic idea to respond to this shortcoming is to support two modes: a simple mode - which already covers many applications - and an expert mode. The simple mode will work similar to “Play the Record” (TOP before this thesis), with the difference that there are no buttons on the modules (cf. Figure 4.3). The function of these buttons is now provided by the controller. In the simple mode, the user only has to hold the controller next to the sensors and actuators. In contrast the expert mode offers many more functions. Only the available functions of the actuator or sensor are displayed on the touch screen. For instance, for a lamp, the display shows for the user functions such as *brightness*, *duration* or *light color*; but in the case of a socket, only the state to turn on or off is presented. The user can switch between *simple* and *expert* mode at any time. For example, suppose the user wants to turn on the lamp for 10 minutes with the pushbutton. For this, the user holds the controller next to the pushbutton and is informed on the display that it was linked. The user could now select the *expert* mode, but in this case no action is necessary. Furthermore, the display shows that the

controller is *waiting for a next device*. Therefore, the user goes to the lamp and holds the controller towards the lamp. Now the user wants the lamp to light for 10 minutes. For this the user clicks on the display on *expert mode* and the user will see all available functions on the display. Now the user can select the function *duration* and thus sets the timer.

Furthermore, as with "Play the Record", the user should be able to create several events or programs. Each program contains a number of sensors and actuators which are linked with each other, which are not influenced by other programs. Hence, we may assume that every program has one task. By way of example, *sensor A* and *actuator A* are stored in the first program and *sensor B* and *actuator B* are stored in the second program. *Sensor A* can activate only *actuator A*, and *sensor B* only *actuator B* but not *actuator A*.

As soon as a sensor or actuator is detected by the controller, the controller emits an acoustic signal. If the user selects several sensors, they are linked with *or*. On the other hand, several actuators are linked with *and*. This means that it is enough if *one* sensor of several selected sensors has detected an event to activate all actuators.

We developed the case for our prototype as a box made of wood. This box was fabricated with a laser cutter machine, as it is a very fast method for manufacturing compared to 3D printing. Alternatively, we could have cut this box from a cardboard, but we decided for wood, as this allows to build a robust box with the same effort and creates a better user experience. The box was again designed with the help of *Adobe Illustrator*.

As technical components we have used the following:

- Arduino Mega 2560 R3 microcontroller
- R3 2.8 "TFT touch screen
- nRF24L01+ 2.4GHz as radio frequency (RF) module
- 5V Buzzer
- 125KHz RFID Reader
- RFID-Transponder Key-Tag 125Khz
- 433MHz RF Transmitter
- 5V Output Powerbank as a power source

The *Arduino Mega* was used because we could connect the development board from the touchscreen directly to the *Arduino Mega*. This saved us a lot of unnecessary wiring because the touchscreen needs to be connected with over 10 pins. For the touchscreen, we decided to go for a size of 2.8 inches, because it provided enough room for our text instructions. To communicate with the modules, we have still (as in "Play the Record") decided for the RF transceiver *nRF24L01+*, as this has all the necessary features, such it works with a license-free 2.4GHz ISM band operation, has up to 2Mbps data-rate, is affordable (compared to *XBee*²⁵), easy to operate, has a low power consumption and is small. Furthermore, as a network infrastructure we have used the *MySensors* library²⁶, which supports these RF modules. So that the controller also

²⁵ <http://www.digi.com/xbee/> (retrieved 02.12.2017)

²⁶ <https://www.mysensors.org/> (retrieved 02.12.2017)

knows which sensor or actuator the user is considering using, we worked with RFID technology to recognize the modules. For this project we have used an RFID reader which works with a 125KHz frequency and so that can read the most standard RFID chips. As an acoustic feedback we used a customary 5V buzzer and as a power source we used a portable power bank with an output of 5 voltage from its built-in batteries through a USB port, like those nowadays used for recharging smartphones or other portable electronic devices. The advantage of the power bank is that we can also recharge the power bank through a USB port. Last but not least, a toggle switch was mounted on the outside of the case which allows the user to turn the controller on or off.

As mentioned earlier, we were mainly interested in the interaction between the user and the controller, so we ignored sophisticated considerations about the form factor and implemented the prototype as the electronic components allowed. The physical design was dominated by the power bank and the *Arduino Mega*, as these were our largest parts. Our first design had a rectangular shape, where on the lowest layer was the antenna from the RFID reader and then the power bank. One layer higher was the *Arduino Mega* with all its remaining components. The top layer represents the touch screen, which was directly plugged into the *Arduino Mega* (see Figure 4.9).

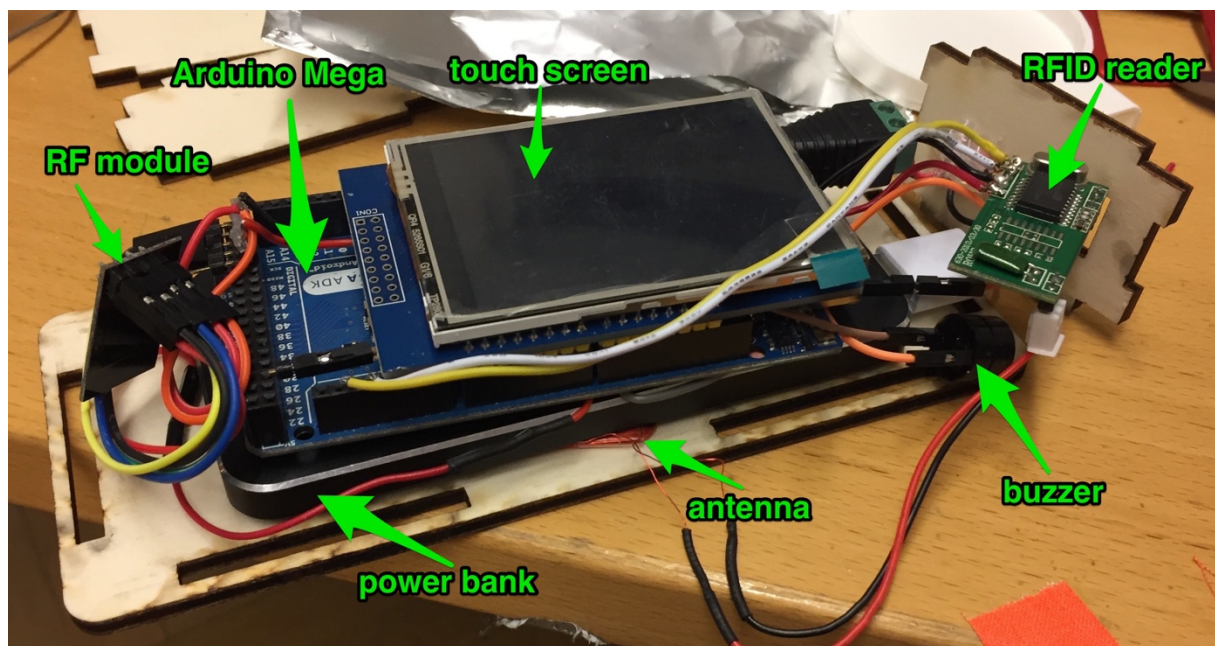


Figure 4.9 - The first mi-fi prototype with all its components

After we've wired all the pieces together and put the box together, we started testing for the first time by putting a test code on the *Arduino Mega*. As a test code, the RFID chip's unique identifier was shown on the display and the buzzer briefly beeped as a reaction that the RFID chip was detected. When reading the RFID chip we had to realize that the reading does not always work or the chips are not always recognized. After troubleshooting, we found out that the power bank is apparently an interference factor to the RFID antenna (see Figure 4.10). "Apparently" because we did not pursue the reasons, we just accepted the fact that it does not work that way. Therefore, this

prototype was due in this form for us and we had no other choice than to change the shape so that the antenna is not (directly) in contact with the power bank.

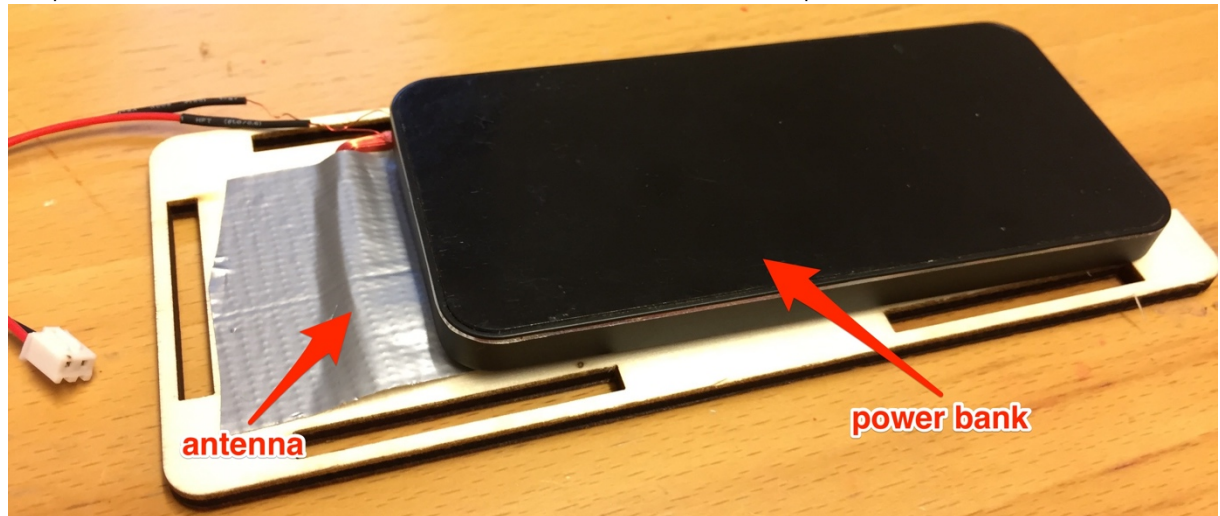


Figure 4.10 The power bank acted as an interference factor on the antenna, which led to redesign the prototype.

The next prototype (see Figure 4.11) had a square shape and this time the components were all separated on the ground and not on other parts. We used the space in such a way that the power bank was on the left side and all electronic components were on the right side, in the middle was the *Arduino Mega* with its touch screen on it. This time the box was quickly created with an *Online Box Designer*²⁷, where we only had to specify the dimensions on the website and then we could download a PDF. Then we were able to import this PDF into Adobe Illustrator, where we could prepare the layout for cutting.

²⁷ <http://boxdesigner.connectionlab.org/> (retrieved 02.12.2017)

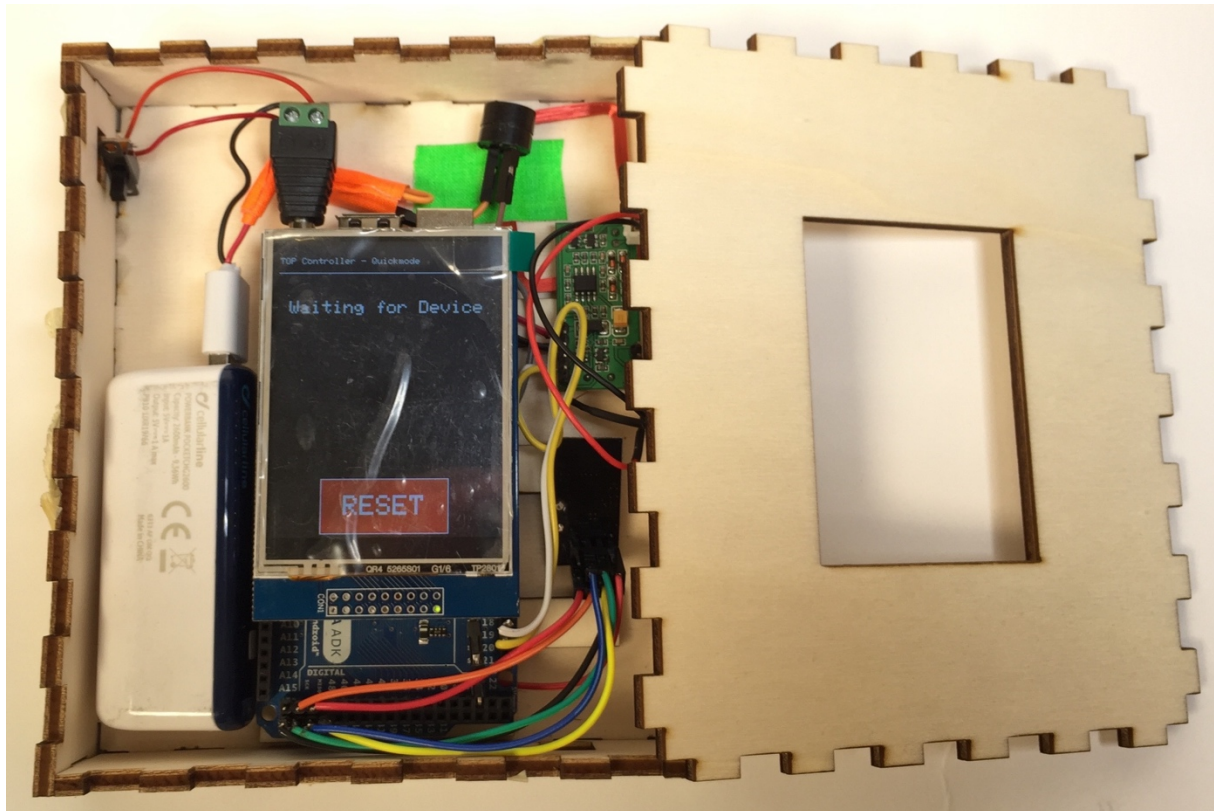


Figure 4.11 - Redesigned prototype where all components are separated

Then, we dedicated to program the *Arduino Mega* to provide the user with more interactive tasks, as described above. We were able to use the sensors and actuators from “Play the Record”, because we had already implemented a functioning *pushbutton*, *buzzer* and *RF socket* there, and these were also equipped with *nRF24L01* components, which we use for communication. We have kept in mind that we should concentrate on just a few examples instead of covering every possible use case with our controller, as we were advised in the evaluation before. We focused on the following examples: “turn the light on and off with a pushbutton” and “turn the buzzer on after X seconds”. We did not want to spend too much time on coding because we did not know if the interaction would be the same in the final version. Still, the examples were as far implemented as the user could choose between *simple* and *expert mode*. In simple mode, the user “simply” had to hold the controller to the objects and in this way the objects were linked together. Whether the controller has recognized the object has been communicated to the user through an acoustic feedback and a text message on the touchscreen. In addition, the touch screen indicated that the user can either *reset* everything, switch to *expert mode* or start a *new program*, and the controller waits for the next device. Furthermore, after the user held the controller to an object, the user could switch to expert mode. In expert mode, the user could see all the functions on the touch screen that the object offered. Only a few offered functions were implemented, the remaining functions were stubbed implemented as a mockup. The implemented functions were *delay* and *duration*, where the user could select between 5, 10, 15 or 20 seconds after clicking on one of these functions (see Figure 4.12).

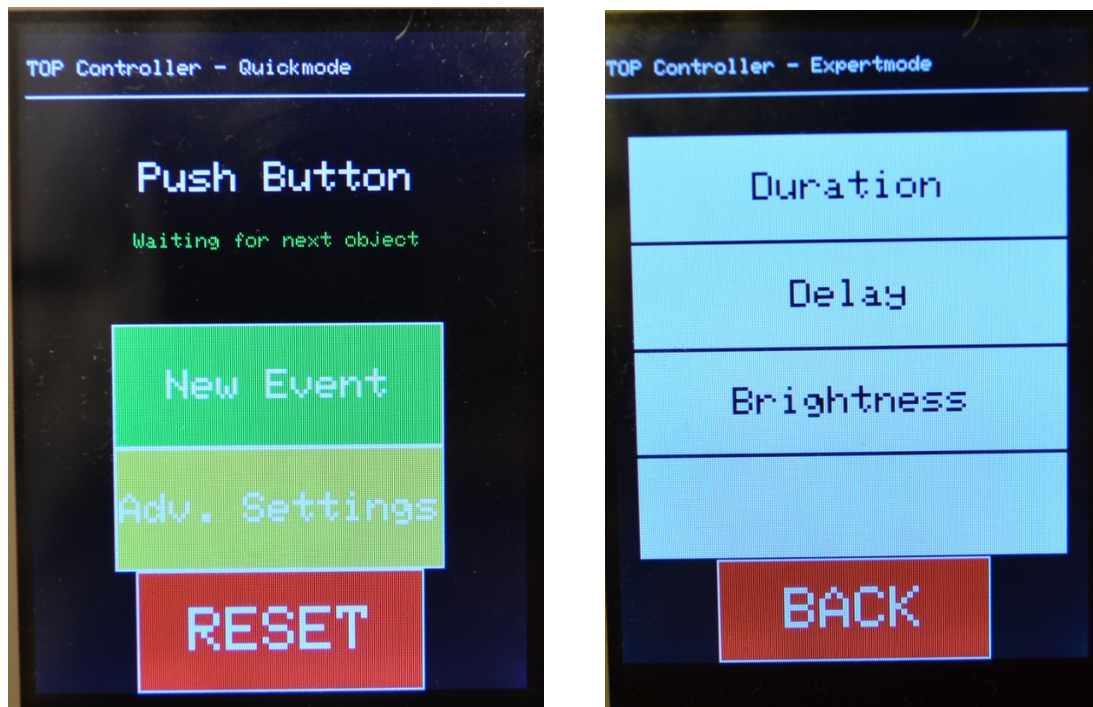


Figure 4.12 – Snapshots of the display: right “push button” is recognized – left: “expert modus” of a lamp

We realized that the prototype was a bit uncomfortable to hold in one hand, and that's why it would suit us if we could reduce the shape of the prototype a bit. In so doing, we tried to replace the *Arduino Mega* with an *Arduino Nano* in the prototype, because it requires considerably less space. At first we were not sure if the *Nano* has enough pins to connect the touchscreen and all other components to the *Nano* in a stable fashion. After omitting some unnecessary pins (for example, reading an SD card, as an SD reader was mounted on the touch screen), there were just enough pins to connect all components to the *Nano*. Thanks to the *Nano*, we were able to make the next prototype (see Figure 4.13) considerably smaller, so that instead as in the previous prototype that all components were to the right of the *Arduino Mega*, all components were now in the middle and the power bank to the left. The equipment and all the functions remained exactly the same as in the previous prototype.

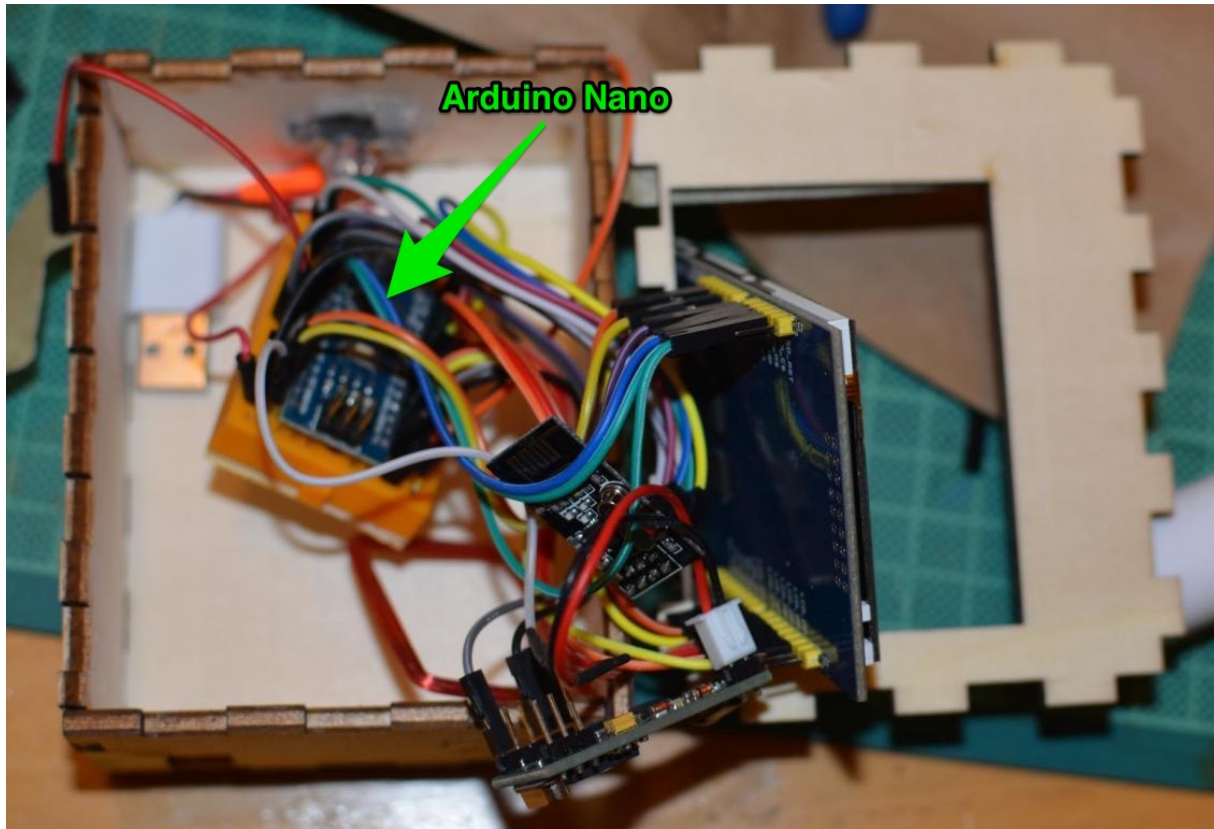


Figure 4.13 - Improved form factor of the prototype by using the Arduino Nano instead of an Arduino Mega.

4.5.2 Evaluation and Insights

For the evaluation of the mid-fi prototype, we invited four people (see Table 4.6) who had experience with computers and smartphones and were interested in new technology. One participant was from the HCI institute with experience in interaction design (*B0*), the other participants had no technical education (*B1-B3*). Except with the institute participant, the prototype interviews took place at the participants' homes, where the prototype was presented in a cozy atmosphere, followed by discussions with the participants about our project and especially the interaction. Also, future development directions for this thesis were discussed, which will be reported in the section about the development process of the final prototype.

It should be noted here that due to the fact that this prototype looked more sophisticated in appearance and functionality, we observed during the evaluation that the participants shied away in voicing bad criticism (compared to the first prototypes). This has led us to deliberately ask people if they found certain things inappropriate about this mid-fi prototype.

PARTICIPANT ²⁸	AGE	GENDER
B0	29	female
B1	28	female
B2	33	male
B3	37	male

Table 4.6 - Overview which attendees participated for the second evaluation.

Enclosed we want to summarize the most important outcomes of this evaluation. As before, we have ignored the problem with “and” and “or” (cf. previous evaluation) and we again have specified that all sensors are linked by default with *or* and actuators with *and*. The evaluation was similar to the first evaluation. After a short introduction about our motivation in this thesis, we explained our prototype to the participants with two simple examples, which were the same as before (see section 4.4.3). The first example was to switch a light on and off with a pushbutton, the second example was to program a timer, so after a certain time the buzzer should make noise.

The process was clear to all participants after we demonstrated it to them. Unlike in the previous evaluation, we no longer felt that the participants were unsure about testing our prototype. Instead, we observed them in having fun finding out how to solve the problem with the controller. In their own account, the participants also considered the procedure as interesting and they had no trouble understanding it. For example, B2 said: *“actually you can not do much wrong, you just have to know where you have to hold the controller”*. B3 meant: *“even if you have no plan at the beginning, after you have played with it for a while, you understand the process anyway”*. The participants were already of the opinion that the display leads them and that the procedure is very playful. For example, B3 meant *“it is like a toy for children, but also for adults.”*

We explained to the participants our intentions of this thesis and that with this prototype we are trying in an experimental way to explore how far home automation can be programmed without smartphones, and we explicitly pointed out that we want to get away from the conventional computer. Thereupon all the participants gave us to consider that we have also developed a kind of “display computer thing” and have actually returned to the computer, so for example, B3 wanted to tell us that *it has its own small box (“Kasterl”), it’s just not an app on a smartphone, but it’s also something proprietary*. B0 has meant that she sees very strong potential for an app or B1 also said that *“(we can) solve this with the smartphone, NFC or maybe something similar, it does not even have to be NFC, but WLAN is enough and then you can control this things from anywhere”*. B2 also suggested that our controller was strongly reminiscent of a smartphone with NFC, because *“apart from iPhone, we already have NFC smartphone, so you can already do many cool things. You have a token at home that you can use to turn the wifi on and off. Just keep it on your phone. I find it very intuitive”*. He also pointed out that this could be used analogously for our concept. So instead of using our controller, we could use a smartphone to use this to link the modules.

²⁸ To ensure anonymity, the names were coded.

We got also other positive feedback, but it was clear to us after all the participants told us that we built a “replacement smartphone” that was not that different from these conventional handheld devices.

The participants liked the approach of working directly with the objects. For example, B1 meant that she thought *“it's cool to go directly to the lamp instead of looking at a list on the smartphone. It's almost as if you were directly asking the lamp to turn on when I press the button here. That makes the whole programming process somehow nice. And yes, programming is so much more fun”*. B3 criticized that *“connecting walking over it as a pretty interesting idea, but what are you doing with things that are hard to reach? Like, for example, the light here on the ceiling, you need a stool so that you can hold the controller up to it?”*

Participant B2 complained that it does not make much sense for him that the sensors are linked with “or”. For him it would be much more intuitive if all modules in an event or program are linked with “and”. If someone wants to do a “or” link, then this should be a special “feature” in expert mode. Admittedly, we have been dealing with the problem “and / or” since the beginning of the work, but we have not really considered it yet. We really wanted to tackle this problem during the next process iteration.

Last but not least, we also received positive feedback about the methods which we introduced to the participants. For example, B3 has told us that he was fascinated by how we conduct research at the university or *“researching such things, or thinking about it. No matter how useful or marketable that is, I think it opens the eyes to other things. So alone the approach I find very interesting. Even if I do not want to use it that way. Nevertheless, I would never have come up with the idea of doing this. My respect for developing such thoughts.”*

To summarize, in this evaluation we learned that our controller was too reminiscent of an app for conventional smartphones. This was presumably because our controller had a touchscreen and some participants associated it with a smartphone. The fact that the controller works directly with the objects was a very interesting approach for all participants. For us, that means that we should design the next prototype in such a way that it does not remind the users about apps, but that we still work directly with objects.

4.6 Final Prototype

The experiences we had with the evaluation of the mid-fi prototype and the discussions with the participants made a strong impact on the final prototype that we introduce in the following.

After considering that we gained sufficient experience and got enough feedback from other people through the iterative design process, we decided to render the lessons-learned into a final prototype for this thesis. We deliberately do not use the term *hi-fi prototype*, because the definition of a hi-fi prototype would be that the hi-fi prototype is very similar to a final product (functions and materials). But in this thesis we have no *final product*; rather we built a *research prototype* which is addressed at research questions and so to answer our research questions (i.e., it is similar to the functions, but we did not spend many resources on the materials and form factor). Simply put, this final prototype should help us, thanks to its interactive possibilities, to answer our research questions.

Building on the feedback from the participants from previous evaluations (both lo-fi and mid-fi prototypes), we redesigned our controller. We also designed makeshift sensors and actuators so users can better test our controller interactively with our sensors and actuators. Thus, we were also able to observe reactions of the participants in response to the system's interactivity.

4.6.1 Prototype Development

In the final versions of this prototype we wanted to incorporate all the collected experience and the valuable feedback from the participants, which we gained from the previous evaluation (see section 4.5.2). To aid the readers' understanding, we recapitulate the important aspects that moved us to fundamentally changing the previous prototype.

- *The controller should not be too similar of an app on a smartphone.*
- *The controller should still interact directly with the objects.*
- *The "simple" procedure (i.e. simple linking of sensors and actuators) should be "as far as possible" maintained in this way, as this is perceived as very intuitive.*
- *All sensors and actuators should be linked with "and" by default and for "or" a separate option should be offered.*

In the previous evaluation, participants told us that our prototype is functionally similar to a smartphone, and they asked why we spent all the effort creating a "standalone device" instead of solving it as an app. This gave us the idea to let the display disappear. We already used the opportunity during the previous evaluation to discuss with the people what they would think if we offered the controller as a "stick" that does not contain a display. After most of those participants liked this idea, our further developments were based on this idea. We decided to redesign the next prototype, after the prototype should no longer have a display. In doing so, we considered an idea that we originally had after the design workshop, namely that we implement our controller as a kind of "stick". In *Play the Record* our modules had buttons (see Figure

4.3). Instead of using the buttons, it seemed to us as a possible idea that the user could hold a stick in his hand and so the user shows the system which modules the user wants to use while the user holds the stick next to a module. The reason why we did not opt for the stick was that at that time we had the problem that the user could not set numerical values with the prototype. As a result, we wanted to design a controller that allows the user to set numbers. Now we have also redesigned the actuators and sensors, and indeed the user can directly control the numerical values on these modules. In other words, with a temperature sensor, for example, the user has the option of setting the temperature directly on the module when this sensor is to be triggered. This is comparable to a thermostat, where we can set the desired room temperature directly on the appliance (how long the heater (actuator) should heat). Furthermore, we now offer the user a set of configurations or actions in the shape of tokens with associated extra functions. For example, to configure times such as *duration* or *delay*, we offer the user a newly developed time module. For convenience, the reader can find an overview of all new modules and functions in Table 4.7 below.

Once again, we want to explain interacting with the new prototype using a simple example, since it includes all the important aspects of programming the modules. To begin with, the user holds a stick in his hand. We called this stick “magic wand”, because we visualized the procedure in such a way that the user metaphorically “enchants” the actuators and sensors, as a fact that they can communicate with each other. For example, if the user wants to turn on a lamp when a push button is pressed, the user holds the *magic wand* next to the pushbutton and then goes to the lamp and holds the *magic wand* next to the lamp. Now the two things would be connected to each other, hence, if someone pushes the pushbutton, the lamp would go on. This process could also be executed with the previous controller (mid-fi prototype with touchscreen), where the process was so intuitive that no display was actually necessary. If the user now, for examples, wants the lamp to light for 10 minutes after the user presses the pushbutton, the user additionally uses our time module. For this, the user holds the wand first to the pushbutton, then to the lamp, and now the user sets a duration of 10 minutes on the time module. Once the user has set the time, the user scans the time module with the magic wand and has completed the process.

As mentioned in the previous section, we had problems in explaining the “and” and “or” logic to non-technically experienced users. In doing so, we considered that we would simply link all modules in a program with “and”, this means that they all belong together. The “or” shortcut is now used by the user, when the user is creating a *new program* because the *new program* is indirectly a “or” linking. Before that, this action was called *new event*, but since we thought it sounded too technical, we renamed the action to *add program*. The advantage now is that the user is no longer confronted with the terms “and” or “or” and now uses this logic unconsciously.

We offer several features to the user and all these features which we have developed are separately presented under section 4.6.2.

At the beginning, we explored different housings for our sensors and actuators using pen and paper, creating sketches. In doing so, we emphasized that all sensors and

actuators should look as similar as possible. On the one hand, the modules should have a certain recognition value but on the other hand, once a casing design is set, we only have to make small adjustments for each actuator or sensor. We actually gave our controller a shape of a wand, because the user has to hold a thick tube in his hand, since inside the tube there were all electronic components including the battery.

As a next step, we have a 3D model on the computer. For modeling, we used SketchUp²⁹ because it has low entry barriers and we could model objects very quickly and easily. The housing of the modules (see Figure 4.14) had a rectangular shape with round corners, in addition, two different sizes were designed for the sensors. Depending on the module, the top (cover) consisted of push buttons or a small display, but all models had a hole for our RGB Led. On the side of the housing, a toggle switch was mounted to turn it on and off. In addition, there were holes on the side of the housing, either as access for charging the battery or for the sensor component.

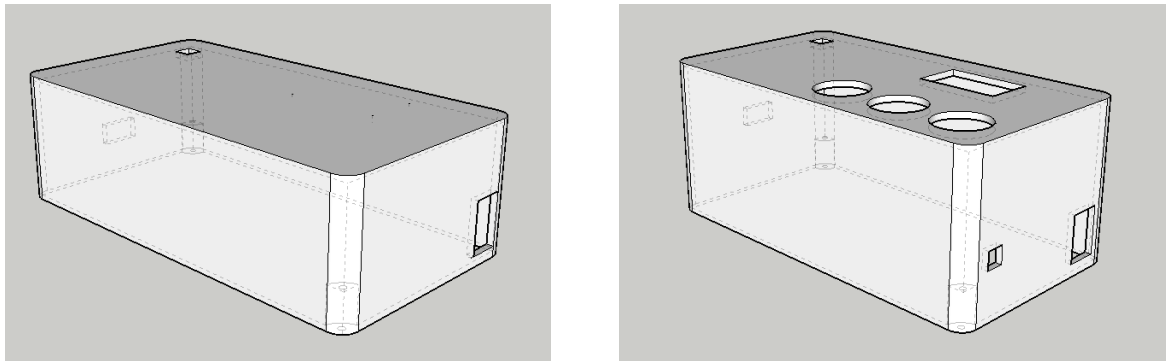


Figure 4.14 - 3D modeled casings for the buzzer (left) and light sensor (right)

For the bottom housing we used a wooden plate, which was screwed on with two screws. For the screws, two special screw holes were modeled on the housing (see Figure 4.14 where we can see the screw holes on both models, top left and bottom right).

The controller was modularly designed with several parts including a base part, where at the bottom there was access to charge the battery, and a toggle switch to turn the controller on and off. The middle housing consisted of two "extension parts". In the top part were four LEDs inside the housing (see Figure 4.15) to assist the user with an additional visual feedback. Furthermore, a flat surface was prepared in the upper part, wherein the antenna was located for the RFID reader. Thus, the user could simply touch the casing from the module with the flat surface from the controller to establish/"program" a connection. Since we did not have much space inside the controller, we had to use a battery based on *Lithium Ion* chemistry instead of a power bank as a power source. Since the Lithium Polymer battery had only 3.7V output voltage, but we needed 5V for our Arduino and RFID Reader, we used a booster that *boosted* the output voltage to 5V.

²⁹ <https://www.sketchup.com> (retrieved 05.12.2017)

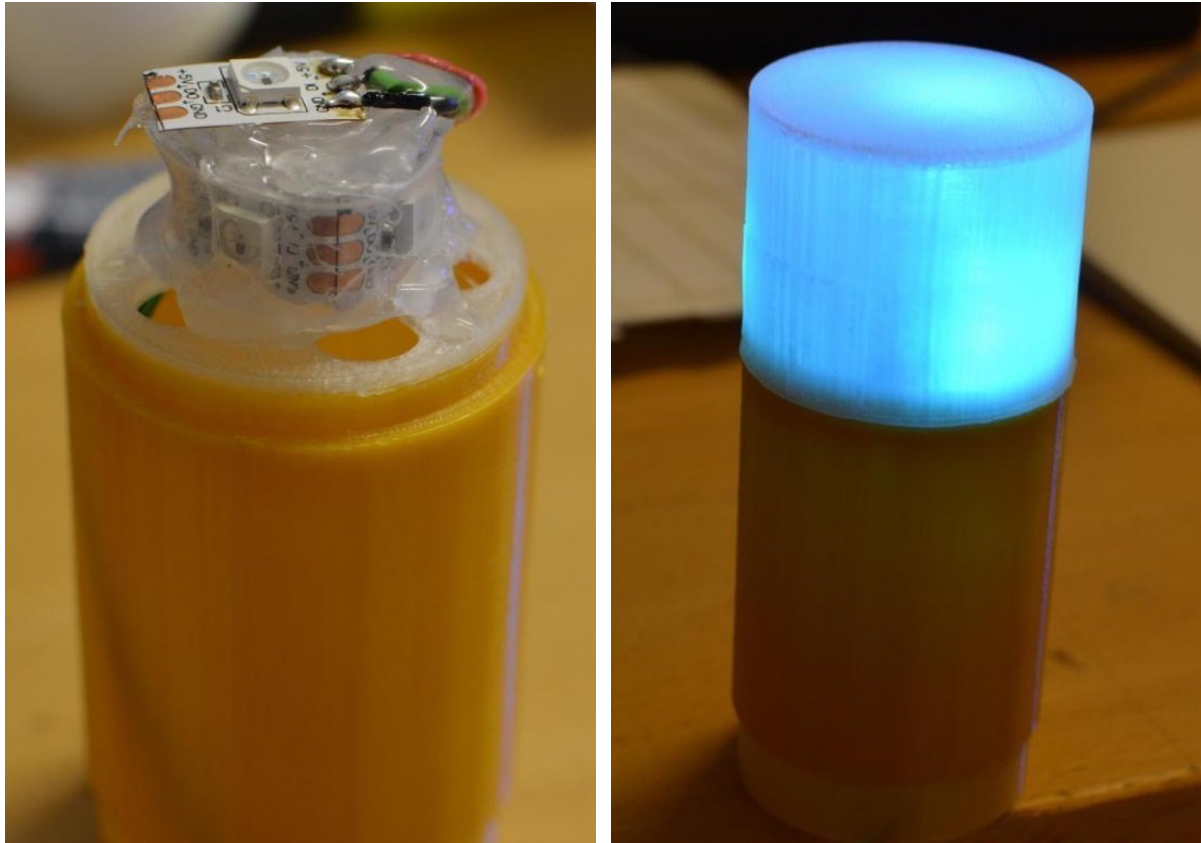


Figure 4.15 – Work in progress: the top part of the controller, the left picture shows how the wiring looks inside and on the right where the top part was melted together.

After modeling, the parts were printed with a 3D printer and after that the housings were equipped with components and the technical components were wired inside. The following technical components were used:

- Arduino Nano
- Adafruit NeoPixel RGB LED Strip
- Light-dependent resistor (LDR)
- Ultrasonic Distance Measuring Sensor
- Digital Temperature Sensor Probe (DS18B20)
- 128 x 64 Pixel 0,96 Zoll OLED (SSD1306)
- RFID-Transponder Key-Tag 125Khz
- 433MHz RF Transmitter
- nRF24L01+ 2.4GHz as RF module
- 5V Buzzer
- RFID Reader 125Khz
- 5V Output Powerbank as a power source
 - Lithium Ion Battery - 3.7V at 850mAh
 - 5V/1A Lithium Polymer Charger/Booster

In addition, we attached labels, which we made of wood, on the casing. Labeled were the functions of the buttons and a mark was placed where the RFID key tag was, so the

user knows where the user should hold the controller. The prepared holes on the housing for the RGB LED were filled with a matte piece of plastic because the light from the RGB LED was very bright and we softened it in this way. The actions and system controls were manufactured as round tokens using a 3D printer. Inside the token was an RFID key tag, just as in every case of the modules. The tokens, along with the *time module*, lay on a wooden board (we called it *Magic Board*, as a reference to the wand), where special recesses for the tokens and the module were cut. On the wooden board, the placeholders for the tokens and the module were labeled as groups with *Actuator and Sensor Operations*, *Timer Module* and *System Controls*.

After assembling all modules (see Figure 4.16) we started to program all Arduinos. Since we were using nRF24L01 RF-modules in our project, we used the *MySensors* framework, which serves to communicate between all the modules. Invisible to the user, we had a *gateway* that received all messages from all modules or sent them to the modules. The messages were forwarded to a computer through a serial port. The entire logic was programmed on the computer using Java³⁰. The computer could in turn communicate through the serial port with all modules. The *gateway* consisted of an *Arduino Mega*, because this had more static random-access memory (SRAM) than an *Arduino Nano* or *Uno* and so could better handle the message traffic. Nonetheless, in addition to the nRF24L01 module, the 433MHz RF transmitter was connected to this Arduino to control the RF sockets.



Figure 4.16 - Overview of all components of the final prototype. The orange casings represent the actuators, the green casings the sensors, and the yellow casings the system controls and actions, as well as the controller (magic wand). The white case represents the stopwatch, which was used in the evaluation to time the tasks.

³⁰ <http://www.oracle.com/technetwork/java/index.html> (retrieved 05.12.2017)

4.6.2 Implemented Features

In this section we want to briefly introduce all features of the final prototype. Table 4.7 presents all the actuators, sensors, actions and system controls we provide to the user.

As actuators (orange housings), we came up with a *Twitter token*, which allows the user to post sensor data on Twitter. The *buzzer* provides the user with acoustic feedback. With our *RGB lamp*, the user could set a specific color in which the lamp should light up. The *RF Socket* allowed us to control a wide variety of devices, such as turning a fan, a table lamp or a radio (basically everything that can be plugged into a standard power outlet).

For the sensors (green housings), we strictly distinguish between two types of switches, namely a *toggle switch* and a *pushbutton*. Thus, in the case of a *pushbutton*, the actuator is only active as long as the user holds down the pushbutton. We also offered classic sensors such as *temperature*, *light intensity* and for measuring a *distance*. All of these modules featured a display on top that allowed the user to read the current measured value from the sensor and set a *threshold* of their own when this module was triggered (i.e. tell the system that this module is active). In addition, the user could set whether to trigger when the measured value is *greater* or *less* than the threshold.

As actions (originally we called these *operations*), the user could by means of the *time* module *delay* actuators or activate them for a certain *duration* or set a specific *time-point* when the actuator should start.

The actions *time*, *on* and *off* we restricted only to use on actuators, as we believe it would make no sense to offer them also for sensors. The action *invert status* allowed the user to invert the state. For example, all actuators are *off* by default, this action allows the user to set the initial state of the actuator as active. *Invert status* can also be considered as a “logical complement”, i.e. the user can for example negate a pushbutton, which means that the pushbutton is active as not pressed. With the *On* or *Off* actions, the user can force a state on an actuator. For example, if the user wants to create/“program” an emergency stop button, the user can associate the actuators with the *Off* action. As soon as the switch is pressed, all actuators will go off. These actions are also used, if an actuator was activated in an other program, to override the state of the actuator and then to disable the actuator. The actions were initially implemented so that the action came after the module. Later, after the evaluation of Phase I (see 5.3), this has been changed so, that the action must be set before the module. It was initially so (i.e. in Phase I) that when a module was linked to a *time-action*, the RGB LED on the time module would turn yellow for a short duration, otherwise blue, meaning that the last module was not an actuator was. Finally, it was implemented so that the LED of the time module lights up in yellow until it is linked to an actuator, and then the LED went back to green.

With the system control *Add Program* the user can create a new program. As already described in 4.4.1, each program contains a number of sensors and actuators which are linked to each other, which are not influenced by other programs. *Reset All* discards everything the user has done, allowing the user to start from a clean system, and with *Last Undo*, the user can remove the last used module. Last but not least, we made a *stopwatch* for us as a module with which we stopped the times per task during the evaluation. The task duration was then automatically saved on the computer as a text

file. In addition to the duration, the modules which were used by the participant and a sheet number were also saved in the text file, so that we could compare the times with the questionnaire in retrospect.

As feedback for the user we either used the RGB LED or used an acoustic signal. Therefore, all modules and the controller were equipped with a *NeoPixel RGB LED* (see as example Figure 4.15). If the LED was *green*, the module (or the controller) was connected to the *gateway*, if it was *red* the module was not connected. The color *yellow* was used to tell the user that the module is *in use* in the program. The controller also informed the user with a short beep when a module was detected. Furthermore, the controller signaled an error with a “triple beep”, which means that the module is either unknown or not connected to the system. Since the tokens did not have a RGB LED, the top of the controller shone in yellow for a short period of time to tell the user that the token was recognized in the system (note: the acoustic feedback from the controller only means that the controller has detected the RFID token).

<i>Actuator</i>	<i>Sensor</i>	<i>Actions</i>	<i>System Controls</i>
Buzzer	Pushbutton	Invert Status	Add Program
RGB Lamp	Toggle Switch	On	Reset all
Twitter	Distance	Off	Undo Last
3x RF Sockets	Temperature	Time	(Stopwatch)
	Light		

Table 4.7 - All provided features which were developed

4.6.3 Troubleshooting

We do not want to spend too much space on the problems during the design process, but in every development phases people are faced with problems and (technical) obstacles and so we had ours. Here we would like to briefly mention which four problems significantly slowed down our development process.

1) The problem with the power bank we have already described above (i.e., it had an interference effect on the antenna). 2) With the power banks we yet had another problem. As it turned out, most power banks turn off automatically when there is not enough load on the voltage output. After our modules did not need as much load as we needed to recharge a smartphone, for example, we were forced to increase the load in the modules by installing additional “dummy” components. In the end, we installed three additional RGB LEDs, which we have lit up white at full power and which we then taped to make the white light invisible, so that the power bank does not turn off automatically. 3) When working with the *nRF24L01* modules we often had problems at the beginning, that messages were not sent to other modules or messages were not received. This was due to the fact that the full voltage is needed for the transmission, which is not always constant in the circuit. This problem was solved using a 100µF capacitor to bypass the power supply. 4) We also had problems with the temperature sensor, because at the beginning we were not able to use the temperature sensor and the small display (on the casing) at the same time. If one of the two things were connected separately, it worked without problems. We tried several temperature

sensors (for example LilyPad³¹), finally we were able to read both the display and the temperature with the *DS18B20*. Unfortunately, we were unable to display the measured temperature in “real time” on the display because reading the sensor data takes longer than one second. Thus, we were forced to set the sampling rate higher (in the end we set it to 5 seconds).

4.6.4 Evaluation and Insights

The final prototype was used to answer our research questions. Therefore, this evaluation was conducted with several participants and analyzed with approaches on the HCI discipline, as described in the methods section (see chapter 3). Hence, a separate chapter (see chapter 5) has been dedicated to this evaluation.

³¹ <https://www.sparkfun.com/products/8777> (retrieved 07.12.2017)

5 Findings

In this chapter we want to present the (final) evaluation of our final prototype. As already discussed in the methods, this thesis has an iterative design character where we have already carried out two evaluations (see section 4.4.3 and 4.5.2) for earlier prototypes. In these evaluations, we were able to gather some feedback from the participants, and considering this feedback finally took us to the development of the final prototype (see section 4.6). The evaluation of this final prototype is part of the iterative design process of this thesis, but in contrast to the previous evaluations, we do not specifically want to receive feedback from the participants to incorporate it in a next prototype, but in this iteration we draw our conclusions, which we will discuss in the following chapter 6. For this reason, we first want to prepare all acquired examination results in this chapter and then summarize them.

The evaluation of the final prototype was conducted in two phases, as already mentioned in the methods section. Phase I took place during the *open house day* of the faculty of informatics of the Vienna University of Technology and involved people with a strong interest in technology (prospective and first-semester informatics students). Phase II was an in-situ evaluation of TOP at the homes of senior participants who were not experienced with technology. In summary, a total of 30 people were involved in the final evaluation as participants (see in section 3.5). This evaluation provided a contrast between people with and without an affinity to technology.

We distinguish between qualitative and quantitative collected data. The former was obtained through the feedback of the participants, the second by completing a questionnaire and by taking the times the participants needed to solve specific tasks. The results are first examined with regard to phase I, then regarding phase II, and finally we reflect on the differences and similarities in the performances of Phase I and II.

5.1 General Procedure

The process took place roughly in four steps for both phases, which we will briefly describe here, in order to explain how we got to the findings. It should be noted that our intentions and research motivation in the context of home automation (smart home) applications were explained to the participants upfront.

Introduction: In the beginning, participants were told succinctly and without many technical terms our motivations for conducting this work, namely that we want to explore in an experimental way how people experience alternative approaches to programming home automation technology (commonly called smart home), for example, by using tangible computing devices.

Demonstration and Video: The participants were provided with a simple “live-demo” for programming elements with our prototype, so they could immediately get an idea of our intentions. Afterwards the participants were shown videos (see Findings Appendix for snapshots) where all important features of TOP were explained in a compact way. The motivation for showing videos was that they enabled us to ensure that all participants received the same “standardized” introduction to the final prototype.

Hands-on Activity: Afterwards, the participants were asked to try out TOP themselves. For this purpose, we devised tasks (see Table 5.3) for the participants, where we also took the time, how long they needed to solve a task. The participants were encouraged to say aloud what they think (*thinking aloud technique*) while trying our prototype (*topping*) and were asked if we can record them on audio. After the participants had finished trying out our prototype, a discussion was started with the participant. If they did not already mention this, we asked them, for example, how the participants liked the prototype, whether the participants would use it and what for, what they would improve in the design of the device, etc. Finally, participants were asked to complete our anonymous questionnaire.

Field Notes: While the participants were not in the demonstration room, we made field notes regarding what we thought was important (such as “what worked well”, “sources of error”, “learning curve”, “frustration”, “Controller's handling” etc.).

5.2 Evaluation Phase I

As stated in the methods section, in Phase I we had 27 participants who filled in questionnaires (M1-M27). To support the readers in understanding our sample at a glance (characteristics, computer skills of participants; see section 3.5), we present subsequently the ten questions of the survey followed by a color-coded table (Table 5.1) of the answers of the participants. Note, this should serve as a descriptive statistic of the sample. We do not aim at conducting inferential statistics and generalizations. Apart from their “age”, “gender” and their “highest degree”, we asked the participants to fill in the following, where they were invited to choose between “strongly agree” (5) and “strongly disagree” (1):

- Q1: *I have already heard a lot about smart homes*
- Q2: *I have already programmed a lot in a computer language (such as Java, c++, php or similar)*
- Q3: *Programming concepts like IF, ELSE, FOR, WHILE are very familiar to me*
- Q4: *The presented prototype is very intuitive in my opinion*
- Q5: *I would use a fully-developed system to the presented prototypes myself*
- Q6: *I would recommend a fully-developed system to the presented prototypes for people without technical reference (for example, some older people)*
- Q7: *I already own smart home products like the Philips HUE*

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7
M1	5	4	5	4	4	5	2
M2	4	4	5	4	4	4	2
M3	4	4	4	4	4	4	2
M4	5	5	5	4	5	5	1
M5	4	1	4	5	4	3	1
M6	5	4	5	4	3	5	4
M7	5	4	5	2	4	4	1
M8	5	5	5	4	4	3	1
M9	4	4	5	4	4	4	2
M10	3	5	4	4	4	3	1
M11	3	5	4	4	4	3	1
M12	4	4	5	3	4	3	2
M13	4	4	5	5	4	5	1
M14	5	3	5	5	5	5	1
M15	5	3	5	4	4	5	2
M16	5	4	5	5	4	5	1
M17	3	2	4	4	4	5	1
M18	5	4	5	4	4	4	4
M19	4	4	4	4	4	3	1
M20	5	5	5	4	3	5	5
M21	4	1	4	4	4	2	1
M22	3	4	5	4	4	3	2
M23	5	2	5	5	4	5	1
M24	3	4	5	4	4	5	4
M25	5	4	5	4	3	5	1
M26	4	4	5	3	1	4	1
M27	5	4	5	4	3	3	2

Table 5.1 - Overview of the participants and their answers to the survey as a color-coded table, where dark green means strongly agree (coded as 5) and white denotes strongly disagree (coded as 1). Note: the first seven participants in the table have tested the prototype on their own (M1-M7), the remaining participants was presented the prototype, but they did not try it out (M8-M27).

Scale	Q1	Q2	Q3	Q4	Q5	Q6	Q7
1	0	2	0	0	1	0	15
2	0	2	0	1	0	1	8
3	5	2	0	2	4	8	0
4	9	16	7	19	20	6	3
5	13	5	20	5	2	12	1

Table 5.2 - How many times a scale has been selected with the respective question

Table 5.2 represent the distribution of ratings per question, hence we see for example that Q1 was answered 13 times with “strongly agree” and Q2 twice with “strongly disagree”. Figure 5.1 Illustrates Table 5.2 as a bar chart.

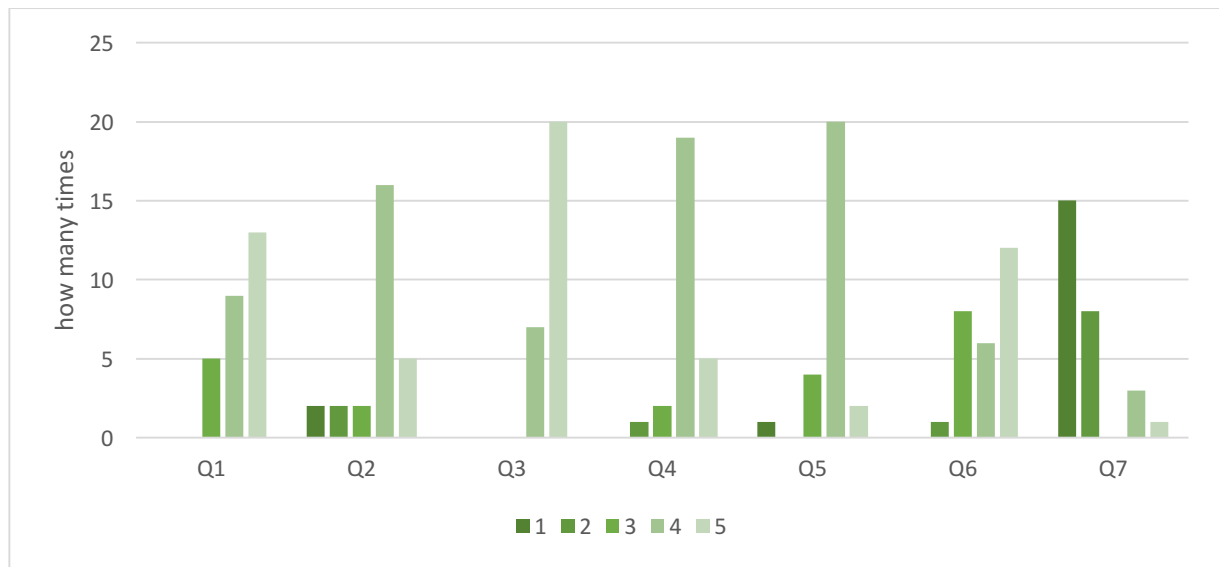


Figure 5.1 - Illustrates the distribution in Table 5.2- how often a scale has been selected per question

As mentioned above, we designed a set of tasks to be solved by the participants. Table 5.3 gives an overview of the tasks in "pseudo code". With the first task we wanted to test run a simple linkage ("and") and our concept of how to use sensors. The second task showed how to work with "and" and "or" operations respectively with "add program". The third and fourth tasks demonstrated how to work with times. In addition, the fourth task included how to work with "invert status". The last and fifth task explained how to use the "on" and "off" tokens. If a participant made a mistake, she or he could correct the error with "undo last", so they did not have to start over from the beginning. The participants were also familiarized with "reset all", in case of they wanted to start again from a "clean" system or status.

#	TASK	PSEUDOCODE ³²
1	Post the current temperature on Twitter when it is above 23 degrees	<pre> if(temperature > 23) twitter(temperature); </pre>
2	Use with the toggle switch the buzzer or with the pushbutton the lamp	<pre> if(sensorSwitch.isActive() == true) buzzer.activate(); else buzzer.deactivate(); if(sensorPushbutton.isActive() == true) lamp.activate(); else lamp.deactivate(); </pre>

³² Program code which serves to visualize a paradigm.

3	If it is dark in the room, turn on the lamp for 6 seconds	<pre> if(brightness < 0.20){ lamp.activate(); last = millis(); while(millis() - last < 6000); //wait lamp.deactivate(); } </pre>
4	Use the pushbutton to turn the buzzer on after 4 seconds for a duration of 4 seconds, in addition, the lamp should go on when the pushbutton is not pressed	<pre> if(sensorPushbutton.isActive() == true){ last = millis(); while(millis() - last < 4000); //wait buzzer.activate(); last = millis(); while(millis() - last < 4000); //wait buzzer.deactivate(); } if(!(sensorPushbutton.isActive())) lamp.activate(); else lamp.deactivate(); </pre>
5	Use the toggle switch to turn on the fan and use the pushbutton as an "emergency stop" (i.e., when the pushbutton is pressed, the fan should go off (immediately), even if the toggle switch is on)	<pre> fan.activate(); while(sensorSwitch.isActive()){ if(sensorPushbutton.isActive()) break; } fan.deactivate(); </pre>
6	Free interactions with no instructions (optional ³³): Use any two sensors and actuators, then remove each (last) used module separately	

Table 5.3 –This table presents all the tasks assigned to the participants. Since we are talking about programming in this project and TOP should indeed replace the actual programming, we additionally want to illustrate the tasks in the form of a pseudocode exemplifying how this task is to be understood in the context of a programming language.

The open house day of the faculty of informatics of TU Wien provides interested students with the opportunity to "look behind the scenes" of their university.

³³ This task was not taken into account in the evaluations and statistics.



Figure 5.2 - Snapshot from the room where TOP was presented to the participants.

At this occasion, we prepared a demo of TOP (see Figure 5.2). Interested students were given detailed information and instructions on the concept. They could also voluntarily fill in our survey (see also appendix) and use the prototype.

A total of 27 people were introduced to the TOP prototype in Phase I and provided feedback in discussions and by filling in the survey. 7 people voluntarily tested TOP and solved our five tasks that we designed for this lab study (see Table 5.3).

Below is a summary of what we were able to observe and what was provided by the participants as feedback. Figure 5.3 shows simple descriptive statistics which we generated from the timings we took during the fulfillment of the tasks and Figure 5.4 shows the average times as Box-Whisker-Plots. In order to facilitate taking the times and to allow us to focus on the participants, we built a special *stopwatch* module, which enabled us to stop the time the participant needed to solve a task automatically. Once the time was taken, a text file with the identifier of the questionnaire was stored on the computer (note: we could enter the identifier of the questionnaire on the module for matching times and questionnaire responses).

Subsequently, we grouped our qualitative observations by the different tasks.

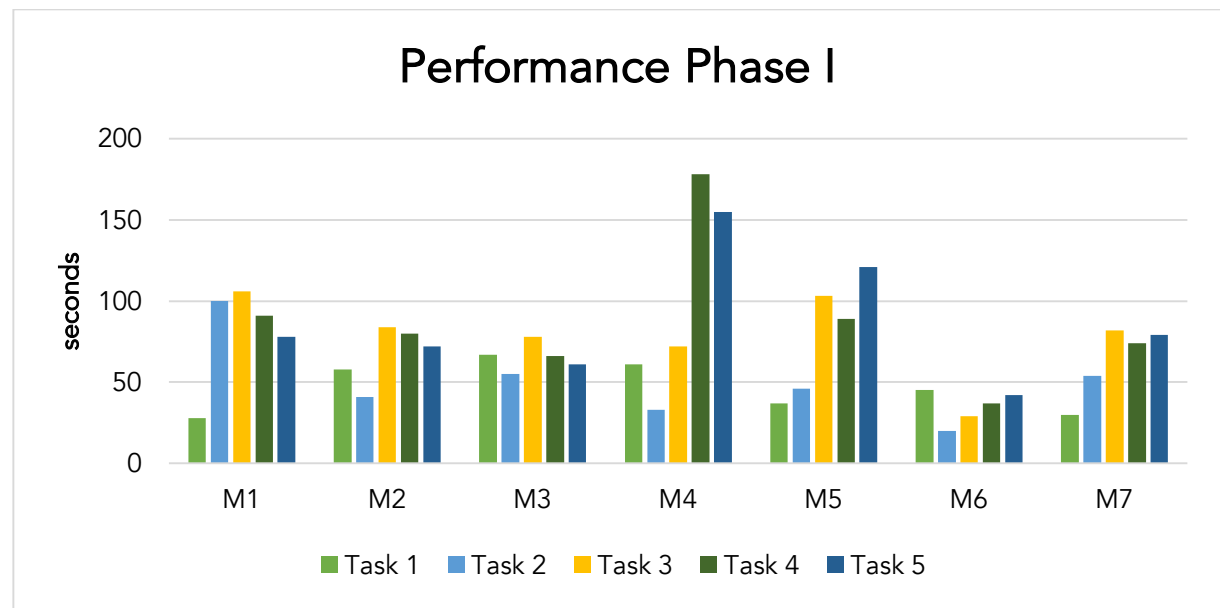


Figure 5.3 - Descriptive statistics of performances in seconds from Phase I, how long each participant needed to solve a task.

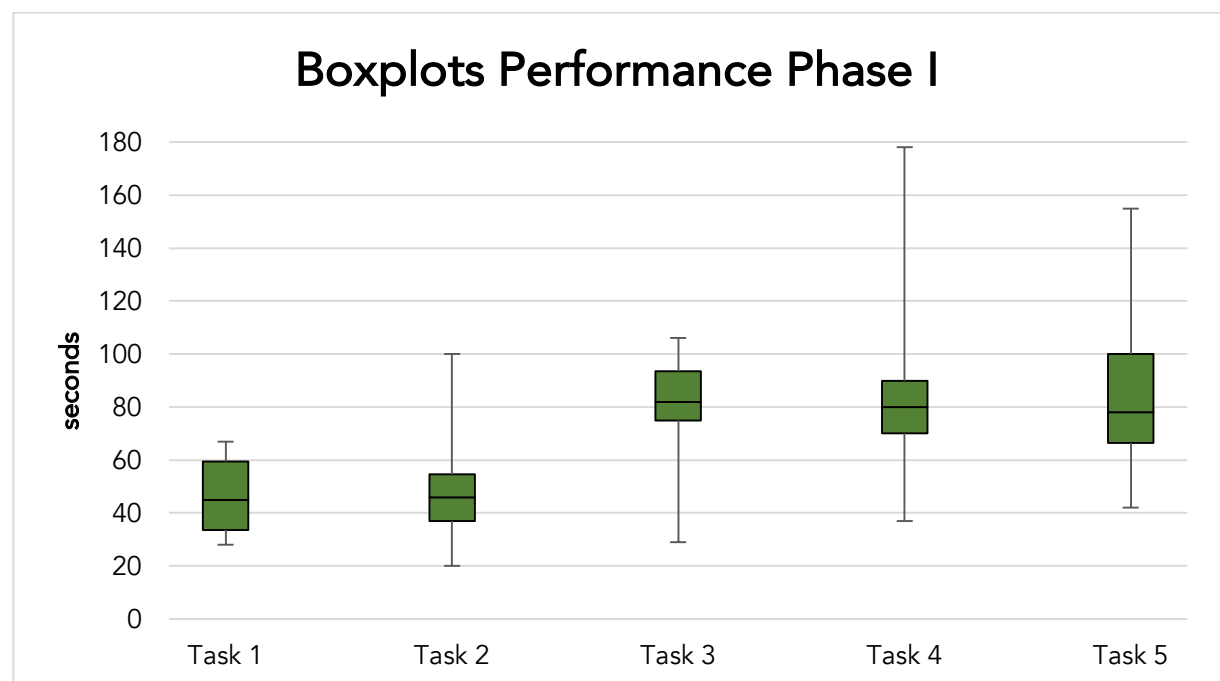


Figure 5.4 - Box-Whisker-Plots of the average performances in seconds per task from Phase I

As evident from this data, the participants were able to solve the different tasks in reasonable times. Overall, the TOP system was described as intuitive and easy to use by most of the participants. M2, for example, mentioned that she finds the concept great for her grandparents, as *"I find it extremely intuitive, because I know that, we have half a smart home at home and my grandparents are so overwhelmed with it, thus the concept is great, if you are not a smart home expert"* and M1 said that she finds *"the project really cool and interesting. Quite simple and offers possibilities to design your own home."* In addition to a lot of positive expressions of amazement like *"wow"* or *"nice"*, the participants have praised our prototype, for example, M2 said that this is *"a*

great project, with lots of potential to give non-technology affine people control in smart home environments.” M4 kindly informed us that he thinks “the concept is very interesting, I must say! Actually, this is perfect to really give people a way to configure their own devices without first learning Java or something similar” and M5 told us, that she appreciates the privacy in the project and finds the approach “very exciting to develop smart home products where data stays local and does not converge or is store centrally and then is being further processed.”

Nevertheless, and to little surprise, we did not receive positive feedback only. Rather, the participants provided us also with constructive feedback for improving the prototype system. This feedback was also in line with our own observations during the lab study.

In the beginning of the test, almost everyone asked where she or he was supposed to hold the magic wand (controller). The people were slightly irritated, if the “scan” (that is, the controller has recognized the module) did not work as anticipated. However, after we prompted them to keep the wand below the LED and informed them where the antenna surface of the device was, they had no more trouble after that. Maybe we should have provided an additional indication, for example, on the magic board or some other location, where exactly the participants should hold the magic wand.

We go on to report observations and feedback grouped by task.

Task 1 (post on twitter): With regard to the order in operating the sensor module, almost all participants were unsure whether to set first the values (e.g., a temperature value) and then scan the sensor module (with the controller) or if they could scan the sensor directly (without presetting a value beforehand). This was due to the fact that some participants (M3, M5, M6) were unsure whether the sensor values, which they sets at the module, are saved immediately into the system (in the background) or not until they scanned the sensor module. Indeed it does not matter, because we configure only the sensor when it should trigger (i.e. the sensor module sends “active” to the system). Some participants (M1, M2, M5, M6, M7) have asked at the beginning, whether this sequence is important, for instance, whether we first have to select the sensor or the actuator. However, most of them proceeded according to the sequence as provided in the written task instructions, namely first the sensor and then the actuator. In fact, for the sensors and actuators, the order was actually not prescribed.

Especially with such a non-technically complex implementation as “post sensor values on Twitter”, we could see from all participants a positive expression of amazement. People immediately realized with this task or rather with the twitter-post that the prototype really worked and could also communicate via the Internet.

Task 2 (turn on/off lamp): The difference between “and” and “or” operations could not be understood immediately for the most participants (M1, M2, M3, M5). The participants were unsure whether “Add Program” is a logical “or” representation as we had pre-defined it. M1 and M5 were unsure whether, according to “Add Program”, the pre-existing links would persist. Admittedly, the examples were designed in such a way

that we wanted to examine if the participants understood that "Add Program" is a representation for "or".

One participant (M7) thought the state of the sensor, when he is scanning with the controller, was important because he thought he was programming the sensor with the state, that is that the system remembers the state of the sensor as it was scanned. For example, if a user links a turned-off toggle switch with a lamp, then the lamp would be lit when the toggle switch is off.

Even though not everyone understood the system right away, M5 told us that she thinks *"it takes a bit to get in touch, like a game, that you understand the order and know what's going on"* and M7 told us he *"finds it an interesting design choice to withhold the terms 'and' and 'or' in other terms."*

Task 3 (lamp duration) and Task 4(duration, delay and invert status): Almost everyone (M1, M2, M3, M5, M6, M7) wanted to use the time module first and then select the actuator. A person (M2) also said that it would be the best for them if the sequence would not matter. It must be noted that there must be an order. Either the action ("invert status", "on", "off", "time-parameter") has to be set after the actuator or before the actuator. Without a syntax, the system would not know who to assign the action to. However, many participants (M1, M2, M3, M5, M7) found it more intuitive that the action should be selected first, and then the actuator. Thus in general the order should be like this as they suggested: sensor-action-actuator.

To set the delay and duration functions was confusing at first for M1, M4, M5 and M7. They thought that they could directly set the delay and duration function and both set values are applied to the actuator simultaneously. In fact, they had to scan each delay and duration function individually, hence they had to scan twice. Nevertheless, the remaining participants (M2, M3, M6) understood that we have to scan each time-parameter individually, including delay and duration. Also, it was not always clear when scanning the delay of 4 seconds, whether they should scan the duration immediately afterwards, or they had to scan the last actor again, and then scan the duration. Indeed, it did not matter, but some participants (M2, M5, M6, M7) felt unsafe. Other participants (M3, M5) were again not sure whether the actuator stores both values (duration, delay), meaning the values are nested. Therefore, they thought that a separate program was needed for each delay and duration action.

In addition, when the participants were asked to set the delay and duration to an actuator, some participants (M2, M3, M6, M7) wanted to set the duration first and then the delay. But if we first set the duration and then the delay, then the duration is first executed and then the delay. They thought the order did not matter, and that always delay and then duration is executed.

Task 5 (emergency switch): During the last task, we were able to observe whether the participants could deal with TOP alone. Two participants, for example, asked what the "on" and "off" tokens were for. Others complained that the tokens were not mentioned by us and as a result, for example, M5 meant that *"you just have to know that the 'on' and 'off' tokens overwrite the source devices"*, hence the participant was missing an explanation. It should also be noted here, however, that the tokens "on" and "off" were not explained by us, because we wanted to test in this way, whether the participants

themselves came up with ideas for how to use one of these tokens ("on" or "off"). Interestingly, the task was solved differently by the participants (M2, M4, M7) - the solution (they interpreted the linking as a logical *contradiction*) was not quite as intended by us, but was definitely a correct approach. However, this also showed us after the participants had made up and implemented a couple of use cases, that the participants understood how to apply the logical connectives in TOP. For instance, if the participant did not want to join the modules with "and", the participant used "Add Program" instead.

When we saw that the participants did not know how to use the "on" and "off" tokens, we explained what the functions of these tokens were. As a result, all participants have been able to make use of "on" or "off" and solved the last task.

Participant M4 noted that he still found potential for improvements with regard to "ordinary people" that don't know about technology, because he meant that "on" and "off" is very difficult to understand if one has to understand the internal logic of the technique to know why one needs "on" or "off". Again, most people had the opinion that we should first select the action ("on" or "off" token) and then the actuator (i.e. sensor – action – actuator). Then again, another participant (M4) said that he would find it funny if we offered more logical tokens, such as "XOR", so that people could get even closer to programming. Due to, he claimed that this prototype invites us to try out such logical links in a very interactive way.

On the whole, it can be said that most of the participants enjoyed working with TOP, some were even grateful that they were "allowed" to test TOP. Many participants saw our concept as an interesting approach for (older) people with little technical background, which is why we also went through a second phase, where we wanted to test TOP with people who are not so technophilic. The handling with the controller was easy and intuitive for all participants, and therefore all participants, as soon as they had the controller in hand, knew how to link modules together. The initial comprehension problems were quickly eliminated by the participants after they had worked with TOP themselves or solved our tasks (learning by doing). Of course, there is room for improvement in some places, which we will then take a closer look at in the discussion.

5.3 Evaluation Phase II

Minor changes (see below) were made to the TOP prototype before its deployment to Phase II. In this phase another three participants were exposed to the prototype and the five tasks, as already introduced in the method section (see 3.5). This time, however, the study took place in the participants' homes (see as example Figure 5.5), and these participants had little experience with computers in contrast to the people involved in Phase I (see also description above).

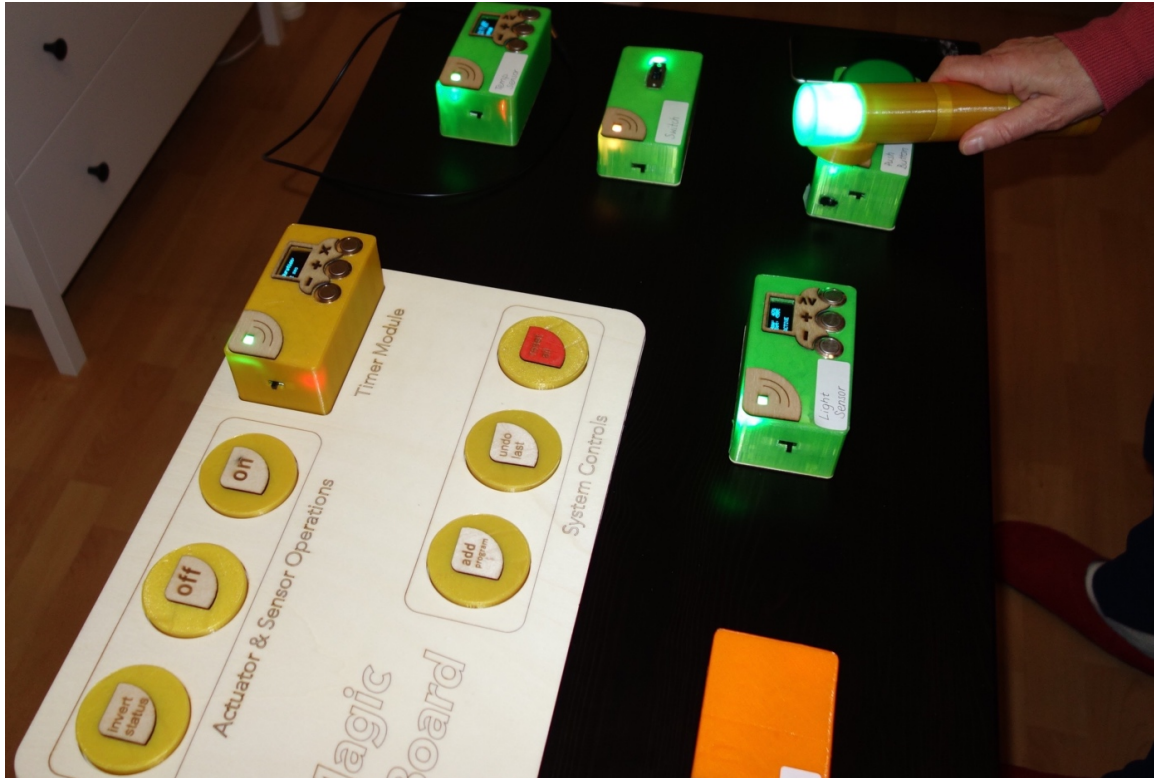


Figure 5.5 - Snapshot of the in-situ evaluation where a participant tries out the prototype.

In the prototype, the following things have changed in Phase II. First, after receiving feedback from many participants of Phase I that we should use the action before the module (instead after the module), we have applied this correction at this phase. For this purpose, the yellow LED on the time module has also been adapted so that it lights up yellow until an actuator has been linked to the time module (after that, the LED goes back to green). Secondly, the participants received more detailed instructions and demonstrations by us next to the videos, because in this phase more explanation was needed and the participants could ask at every step, in case of they did not understand the prototype. Also, we spent more time at this phase with the participants compared to Phase I, where our time was limited (only as long as the open house day was). And thirdly, after we thought that the "on" and "off" tokens were not so easy to understand and we also feared that the participants would probably not know how to solve the problem. That's why we decided that we should redefine the fifth task, as some participants in Phase I have already tried to solve. The fifth task was therefore as follows:

- The fan should only start when the toggle switch is on and the pushbutton is not pressed.

Besides these minor changes in the task or in its wording, the participants were given the same tasks (c.f. Table 5.3) as we did in Phase I therefore we could compare the performances of both phases.

We presented the prototype to three people, who were not experienced with technology and who volunteered to solve our tasks. Again, below is a summary of what we were able to observe and what was provided as feedback by the participants. Figure

5.6 shows again simple descriptive statistics which we generated from the times we took during the fulfillment of the tasks (average times per task as Box-Whisker-Plots).

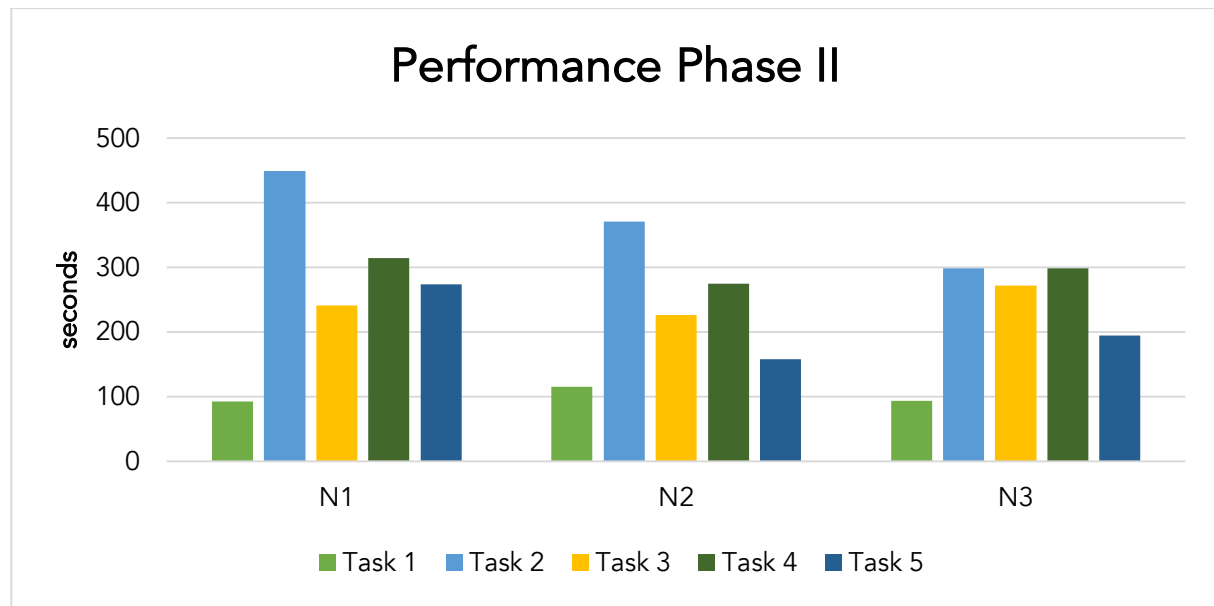


Figure 5.6 - Descriptive statistics of performances (in seconds) from Phase II, how long each participant needed to solve a task.

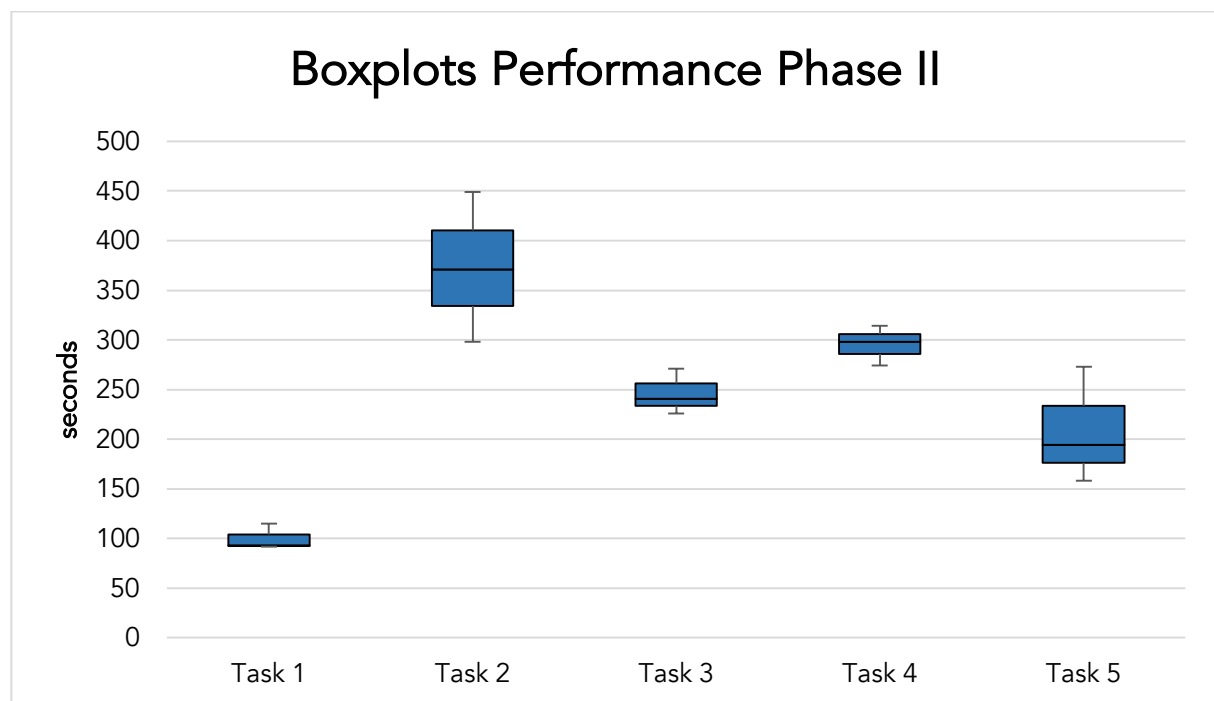


Figure 5.7 - Box-Whisker-Plots of the average performances in seconds per task from Phase II

General feedback: As can be seen in the Table 5.3, all participants were able to solve the tasks we set up in an acceptable time. Even for the non-technology-oriented participants, the operation with the controller was intuitive. For instance, the controller reminded N3 of a "mouse pointer on the computer, with 'which I can select things with'" and N1 commented that she finds "the stick for older people already good, because the young people can configure their home automation with a smartphone,

but I would probably be overwhelmed with a smartphone". N2 told us that "I did not know that you could switch the light on and off with a smartphone, but honestly, I would not even think of using the lights with a smartphone. With such a stick I could already imagine controlling my lights at home." N3 was also enthusiastic that "there are many possibilities how I can link sensor and actuators together." He wanted to tell us that the system is not so rigid and we can actually combine any switch combined with other sensors to activate an actuator. N1 also told us that "if I can really turn on or off with a switch all the lights in the living room, then that's really very convenient and I can imagine using such a system at home."

Practical considerations about everyday use of the participants: The participants also asked questions about how to imagine the whole TOP system in their house. That is, where in reality the *Magic Board* would be in the house, for example, N1 asked if this would be like a "remote control at the table for the TV", or if the *Magic Board* "should always be visible in the living room." N1 was unsure "what would happen in case of a power failure", if we then have to reprogram everything again, or "once I've programmed something, I do not need the stick, but do I still need the *Magic Board*?" Or whether the modules communicate with each other or with a computer in the background. In the end, they also asked if they had to install TOP themselves at their home or if it was already pre-installed in the house. Because otherwise they also had to know how to integrate TOP into their house.

Hands-on mentality of the participants: We have noted with all participants right at the beginning, before we presented our tasks, that they were more interested in trying things out than in listening to theoretical explanations; for example, the participants started to take the temperature probe in their hand to see if "something" at the display changes. They also held the light sensor module next to the window (daylight) to see if the brightness value increased. Compared to Phase I with the young informatics students, most participants were a little more reluctant to try out how TOP works, but at this stage the participants were very impressed with what we had developed in our research. However, we also noticed in this phase that the participants were more afraid to do something wrong, for example, when holding the controller to one of the module. In some instances, they did not receive immediate feedback (because they held the wand too far away from the RFID token) and this irritated them more easily than the younger students of Phase I. Only when the participants moved the controller on the housing a bit down the connection worked. We were also able to observe in two participants (N1, N3) that the participants did not pay attention to the acoustic signal, and although there was no acoustic feedback, the participants moved the controller from one module to the other, because they believed the controller recognized the module. After that, the participants did not understand why the link did not work, even though the participant did everything right. This observation showed us that the participants did not pay enough attention to the feedback in the beginning.

Logical operations and advanced function: The senior participants also had more difficulties in understanding the logical operations compared to the younger participants of Phase I, that is "and", "or" or "invert status". This came too little

surprise, because for people without a technical background such things are completely new and not commonplace. For this reason we spent a lot more time with the participants in making the logical linking understandable (that's why the second task at Figure 5.6 had such a long time to complete).

Another observation relates to the advanced function of "reset all". At the beginning of each task, participants were asked to reset the system. *N1* was confused to use "reset all" because the participant thought it would turn off all lights (actuators). After we have clarified *N1*, which function "reset all" has, the participant asks, if the participant then with "off" can turn off all lights (actuators) and turn on the actuators with "on". Afterwards, the participant also knew why this token (reset all) is also marked in red.

We go on to report further insights grouped by task.

Task 1 (post on twitter): The first task was solved by all participants without rough problems. After telling the participant that the temperature sensor works like a kind of "digital thermostat", which is often mounted on the wall next to the light switch, the participants were able to create an appropriate mental model of the TOP system and tried to adjust the temperature sensor on their own. It should be noted here that the participants have complained that the prototype was labeled and programmed in "English" and this greatly impeded the understanding because they did not precisely understand the meaning of every function. This was not complained about by all participants in task 1, but in the following tasks each participant complained about the use of English (technical) terms. Some participants (*N1*, *N2*) were unsure whether they have to write something on Twitter themselves, or if the post was made by our system automatically. Nevertheless, we noticed that the participants were impressed by the fact that they programmed a computer system independently which posted the current temperature on Twitter. For example, *N1* said "wow, so I do not have to do anything and it will automatically be posted on twitter."

Task 2 (turn on/off lamp): We acknowledge that we had to provided more extensive training to the participants of Phase II compared to those of Phase I with respect to logical operations. We quickly realized that the participants lacked the "logical knowledge". Nevertheless, we wanted the participants to reach the solution through independent testing and therefore we explained to them the principle of computer logics in up to ten minutes. All participants finally solved the second task by first connecting the toggle switch to the buzzer and then (without "Add Program") the pushbutton with the lamp. They thought that once we have connected a sensor to an actuator, and then select another sensor that will automatically create a new program. After informing participants that they need to tell the system when to start a new program, they used the token and understood that "Add Program" must be between the two programs, for example, participant *N1* has finally understood this token so that "'Add Program' completes the last task (linking), which is then saved and now I can do something new." On the contrary, *N3* wanted to solve this task and therefore *N3* first made the first link (switch and buzzer), and then wanted to reset everything and then make the second link (push button and lamp). In this way, however, the task was addressed in a wrong way, because only the last link would have worked and not the

additional switch and buzzer. Hence, N3 spent some time on the wrong approach before successfully completing the task.

N1 asked if it was important to scan the toggle switch when it was on. Participants N1, N2 and N3 thought that the state of the sensor was important. For instance, N1 meant that *"the switch was on when I scanned the buzzer. Now I turn the switch off again, and I go with the controller to the pushbutton and then with the controller to the lamp"* and wanted to solve this task in this way. Also in Phase I we had participants who thought that TOP works this way in order to demonstrate to the system which state from the sensor we want to use.

N3 wanted to know if he could delete a particular module or if only "undo last" was available. Again, N1 thought that if he scanned with the controller again a module which he had already scanned, the selection will be deleted. That means, that we can with a toggle-scan select or remove a module.

Task 3 (lamp duration) and Task4 (duration, delay and invert status): In the tasks with the time module, only one participant (N2) scanned the lamp first and as a next step he wanted to take care of the timer module. N3 was unsure whether the time module was working with the light sensor or the lamp. We told the participant that only actuators could be linked with the time module (because it makes for us only sense in this way). In response which he asked how he should be able to know which links are possible and also that there might be a sensor where it makes sense to link it to the time module. Overall, there were nevertheless no big problems with how to use the time module. Most of the participants (N1, N2) set delay and duration, and wanted to scan than the time module. The correct procedure have would be that we only scan what we see on the display. N2 asked, *"why does the time module not remember both values and put the action on the lamp?"* The participants said that this was quite tedious and asked how they should know that this should be done this way. N1 has also suggested that we should offer both ways of interactions, namely that either the user scans each "delay" and "duration" individually or the user only scans the time module once.

All participants also mentioned that everything was in English on the display of the timer module and they had to ask each time what the term exactly meant.

For the "light sensor" was not always clear what the displayed value (percentages) meant. Only after the participants started to try out (e.g. to cover the photoresistor) they understood the percentage.

Fortunately, we also noticed that people understood "Add Program". For example N2 said, *"so I want to do the new link in addition, which means I need a new program, which I start with this 'add program'"*.

Task 5 (invert status): Finally, we have to admit that "Invert Status" was the toughest challenge for the participants because the participants did not know how such a behavior is expressed in logic. However, N1 and N2, for example, once again asked what the individual tokens on the Magic Board meant, came to the conclusion that "Invert Status" is the right action for this task. Hence N2 also mentioned that, *"if you know, which meaning the individual tokens have, then I would do much easier with the tasks"* and N1 concluded that *"I totally forgot that 'Invert Status' is there. If I were to do*

the tasks 10x, then I would know that there is something like inverting. Or I should have noted during your explanations what the tokens do, I could not remember everything.” Even N3 could not remember all the functions and told us that he would have liked to have a manual, where he can look up how to set special actions. He told us, he had a camera that also has a lot of features and “I cannot always remember, for example, how I set the time for a ‘time exposure’, because the function is pretty hidden. But the camera offers a help menu where all features are sorted. There is also a button next to the display for calling up the help menu, so it's very accessible. In the help menu, I can then search for ‘time exposure’ and then it explains in a few sentences, how this works. I find it convenient, especially for settings that are used very rarely, I can always look up quickly, how to do it.”

To sum up, the participants told us that they were generally enthusiastic about TOP, hence they could select objects simply by pointing with the controller and connecting them together. They shared with us that for simple tasks such as connecting a light switch to multiple lights in the living room, they found that the controller worked great. However, they also said that they find it considerably harder to use actions like “Invert Status”, and as a general remark, they did not want the lights to be controlled automatically. For the participants, it would be enough if they could manually switch the lights on and off with a toggle switch on the wall, and to determine which lights this should be, they would use this controller.

In the beginning there were still problems of understanding what “Add program” means, but in the end all participants understood the principle of it. However, each participant has also told us that they, on the one hand would have to practice a bit with our system in order to find out how to use TOP, and on the other hand, that one had to be interested in something like using home automation at home. The participants emphasized that they would rather program their devices with a “remote control” (such like our controller) at home than with a smartphone.

5.4 Comparison of Phase I and Phase II

In this section, we want to contrast both phases. Figure 5.8 shows a descriptive statistic of the average time of all participants per task, split each in technology-affine (Phase I) and non-technology-affine (Phase II). For better illustration, we offer the comparison as a column chart instead of box plots.

We conclude this comparison by reporting the general interest that both groups of participants showed for TOP and tangible programming.

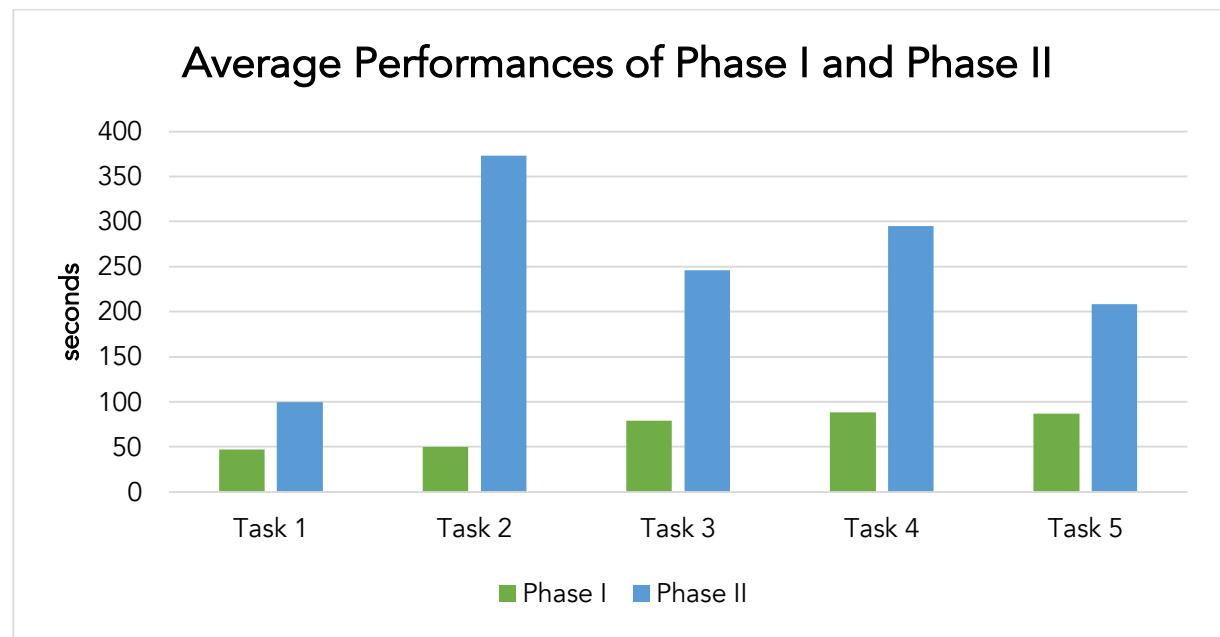


Figure 5.8 - Descriptive statistics of average performances (in seconds) from Phase I and II

Table 5.4 summarizes common problems or questions of the participants in Phase I and Phase II. It shows using color-coding how often this problem occurred or was addressed accumulated across participants. (Note, this is the maximum count of all questions that occurred during the training of the participants before they went on to independently solve their tasks)

Problem Description	Phase I (technology-affine (n=7))	Phase II (non-technology-affine (n=3))
Needed an additional indication on the modules (because the controller did not always recognize a module properly)	5	3
Order in operating the sensor module (e.g. first set the values and then scan?)	6	3
Order, whether first select a sensor or an actuator	4	2
First selected the "action" and then the module	5	2
Did not understand the difference between "and / or" operations with TOP	4	3
Did not understand how "invert status" works	0	3
Unsure, if with "Add Program" the pre-existing links would persist	2	3
State of the sensor was important when scanning with the controller	1	3
Wanted to apply "delay" and "duration" to an actuator at the same time	3	3

Order with "duration" and "delay" (to the actuator)	4	2
Did not pay attention to the acoustic feedback from the controller	0	2
Autonomously set a threshold on the sensor module	1	1

Table 5.4 – Frequency of how often an issue was addressed by participants during the training. Dark red is very common and pale red rather rare.

General interest and user needs in TOP: The main interests of the participants with respect to tangible programming was in their opinion the following tasks in approaching them with TOP. They found the approach exciting in general, how we (in a playful way) can link sensors and actuators together. Most of them thought that our project had a lot of potential to give non-technology affine people control in home automation environments without first learning a programming language. Some participants thought that they could well imagine using TOP as a supplement to existing home automation applications. For example, they suggested "inventing" a combination of TOP and IFTTT. We were also told that they could imagine using the controller to turn their devices on or off. For example, to support people that want to turn off a certain light with the "off" token. Some thought it was interesting for visualizing programming in a tangible way and told us to make the "programming tokens" available (for example, "xor"). Another point of interest was that the whole (sensor) data was stored centrally and not in a "foreign" cloud. In the end, many participants told us that they see great potential in developing this concept as a toy to teach children their first programming experience.

In summary, all participants in both phases have found the handling with the controller to be intuitive. All participants found the controller as an exciting approach to connecting sensors and actuators together. Additionally, all attendees really liked the fact that the entire prototype was interactive and therefore the participants could try the prototype in a "fun way". This interactive behavior helped them to understand the system. The participants were also amazed by the direct responsiveness of the functions, such as the fact that the participant could directly read the measured sensor value on the display or that the sensor value could be published on the Internet (via twitter). They appreciated that they *themselves* had different possibilities to connect sensors and actuators and the function behind them felt like "magic". That is why many participants (M1, M2, M5, M6, M7, N1, N2, N3) were interested in how we implemented parts of the prototype technically and were amazed that this way (i.e. using Arduinos, using 3D printers, etc.) is being researched. The participants (M1, M4, M7, N1, N3) also liked the design of our prototypes, because the housings of the sensors, actuators and the controller had their own colors. Also, they appreciated that the RGB LEDs had a smooth transition during the color change or that we have provided a self-developed RGB lamp as an actuator. Especially in Phase II, the participants were impressed by the design of the wood panel and did not want to believe that we made it ourselves. All these little things contribute to a good usability, because as Aristoteles said, the whole is more than the sum of its parts.

All tasks could be solved by the participants independently, besides sometimes the procedure was not very clear to them (for example in which order actions must be done). Once the participants had tried the system, the process was clear, and thanks to the interactive system, it was possible for the participants to independently try out how some features worked. The biggest differences were in the understanding of logical links, where mainly the technical knowledge was missing. Also, Phase II participants were not immediately familiar with all the features, such as "invert status" and "reset all", which was self-explanatory for Phase I participants. For the "on" and "off" tokens, we got an interesting input because a Phase II participant (N1) thought these tokens were used to turn actuators on or off. For example, if we want to turn on (or turn off) a particular light in the living room, we use one of these tokens to directly control the light without using a sensor.

Nevertheless, in both phases we could observe that after a few applications the participants became more and more familiar with our system, that is, the participants were involved in the beginning of a steep learning curve. Furthermore, we observed that none of the participants of Phase I and II used "undo last". For example, in Phase II, participants wanted a feature that would allow them to remove a specific module that they (unintentionally) scanned. Whereupon the participants in Phase I simply reset everything and started from the beginning, without mentioning that they only want to remove a specific module.

It was also interesting to observe that the participants of Phase II did not pay attention to the acoustic signal during scanning, although in Phase I all participants understood that a module was only recognized after a signal tone. Nevertheless, we were able to observe that the participants of both phases paid attention to the RGB LEDs on the modules (i.e. that one module is in use and the LED lights in yellow).

6 Discussion

This master thesis concludes with this chapter where we want to discuss our results and all the insights we gained during the design process. In this thesis, the main motivation was to explore how people with different skill levels in relation to computers, tablets, smartphones or who have no programming experience can benefit from IoT technology. Moreover, we were interested in researching alternatives to IoT paradigms available on the market or in academia to combine sensors and actuator (i.e., to program sequences of actions and reactions). To respond to this motivation, a number of research questions have concerned us during the entire design process. Therefore we want to conclude our findings by referring to the questions and subquestions (see section 1.2) in the following sections.

6.1 Brief Summary

In order to answer our research questions, we decided to implement an iterative design process in which we finally built three prototypes. The first prototype was to introduce our concept to the participant, the other two prototypes were based on the feedback from the participants during the evaluations. The process started with a literature and market research, followed by a design workshop. In this workshop we learned about the idea that the programming of smart objects should be done without conventional computers or mobile phones/tablets, but through physical interaction with the objects. As a response, we developed a lo-fi prototype, where the actions were realized as physically tangible objects, and these objects had to be plugged in or out of a controller depending on their function. The controller had a display that informed the user about their activities. During an evaluation, it was revealed that the operation was cumbersome and that we should use the display more to guide the user instead of just showing the current state. After that, a mid-fi prototype was developed, where all available actions were shown on the display. The user was able to see which actions were available by holding the controller next to an object. Furthermore, the display has also given the user instructions what should be done next in response to what the user just had done. During the evaluation, we found out that this prototype was reminiscent of an app for conventional smartphones, and we were told that the controller working directly with the objects was an exciting approach. At the final prototyping stage, we developed a controller without a display, which could link sensors and actuators together by holding the controller in each case next to the modules. For this we built our own sensors and actuators as modules. Numerical values for the sensors or for the time module were set directly on the module (as in the case of a thermostat, for example) and tokens (physical objects, with which the users interact in order to gain or manipulate a digital information) were produced for the actions. The final evaluation took place in two phases, in Phase I with young students (technically experienced

prospective students of informatics) and in Phase II with older participants (not technically experienced).

6.2 Reflections on the Findings

To push the research forward, we have used the “research through design” method (e.g.,(Zimmerman, Forlizzi et al. 2007)). With the help of this approach, we explored our problems through an iterative design process, and using a (self-made) prototype, we were able to generate knowledge through observations. Drawing on the generated knowledge through the evaluation, but also on literature search and market research, we were able to build a final prototype. Therefore the design process ended with a final prototype that was followed by a final evaluation.

We contextualized our prototype in the setting of home automation, where our prototype was designed to explore the possibilities of empowering people to control their own devices in their homes. The home automation context was chosen by us because we felt that this was the easiest way to explain our project to the people respectively participants. As already mentioned in the introduction, our target groups besides people without computer experience are also artists, makers, researchers or technology enthusiasts. That is, we classify this latter part of the target groups as people that have experience with technology (as opposed to the first fraction of the target group). Thus, we distinguish between users who have experience with technology and which are not so experienced. This is also the reason why in our final evaluation, we split the evaluation into two phases in order to be able to explore which user experiences are enabled by our prototype.

In the findings section, we found that both technically experienced and non-experienced persons could work well with our prototype. As already mentioned in the Findings, the handling with the controller was easy and intuitive for all participants, and therefore all participants, as soon as they had the controller in their hands, knew how to link modules together. Additionally, all attendees “really” liked the fact that the entire prototype was interactive and therefore the participants could try out the prototype in a “fun way”. This interactive behavior helped them to understand the system.

Initial comprehension problems were quickly eliminated by the participants after they had worked with TOP on their own or solved our tasks (“learning by doing”). Nevertheless, and to little surprise, most comprehension problems occurred in understanding the logical operations, where the older people lacked the technical background knowledge, but also the young students did not immediately understand the difference between “and” and “or” tokens. This occurrence is probably due to the fact that our second task was designed explicitly to check the logical understanding or whether the participants can distinguish “and” and “or” with our system. If the word “program” would have been used in the task text, the results would most likely have been better. For example, if we set the task instruction-text as follows: “in the first program, link sensor1 to the actuator1, and in the second program, connect sensor2 to the actuator2”. Therefore the participants would not be concerned with asking themselves what “and” or “or” meant in this context. This wording would probably also

be more intuitive, because in everyday life, we do not think about how to link something in the household with logic operators. It is more obvious to say that we want, for instance, to connect the switch to the lamp and the light sensor with the roller blind. In general, we also observed that participants from both phases had a different understanding of how our system worked in the "background", because the participants had different background knowledge. This was particularly noticeable in the order of the selections they made. For example, it was self-explanatory for Phase I participants that the selection order of sensors and actuators in a program does not matter because these modules are all linked together. Admittedly, some Phase I participants also asked if they were required to first select the sensor, but after a clarification the order was clear to the participants. By contrast, Phase II participants thought that the sequence was defined by first selecting one sensor and then one or more actuators, then these actuators and the sensor would form a "program". However, when the participants subsequently select a sensor again, the participants thought that this would start a "new program" because they did not understand (at the beginning) that this sensor would still be linked to the previous modules. For the Phase I participants, it was clear that the (last) sensor would be added to the current program. This is probably due to the fact that we (technicians) know how such a behavior is handled programmatically. This is also the reason why in the statistics with the performances, the participants of Phase II have higher times in the second task, because these participants needed more attempts to finally solve the task (which they then have independently managed). Nevertheless, the evaluations have shown us that not all participants think this way and next time we could implement this case and test it again with a few participants. However, in the graph comparing the two phases we can see that the older participants initially needed more time to complete our tasks, but the longer the participants worked with our prototype, the more the participants understood how our system works. That is, at the beginning there was a steep learning curve for the participants, once this was overcome the participants could work with our system.

The participants also told us that in everyday life they will hardly be confronted with such complex tasks. They strongly appreciated the fact that they could link sensors with actuators in a simple way and thought that this was enough for their own purposes.

We observed with the time module that the participants in both phases felt unsafe with respect to the sequence. As we could observe, participants had a different approach, how to link the time module to an actuator. It was therefore quite difficult to find a solution that everybody would understand immediately. We think that the approach with the action between the sensor and the actuator is the most intuitive, since also most people wanted to proceed that way. But as a few people mentioned during the testing, such cases are rather training topics or are solved by the trial and error principle, where the people themselves then come to the conclusion that it either works or they must have done something wrong.

Also, a participant has informed us that the participant still saw potential for improvements for non-experienced people who will use the "on" and "off" tokens, because he meant that "on" and "off" is very difficult to understand if one has to understand the internal logic of the technique to know why one needs "on" or "off". This participant was not so wrong. But well, we told him before, the "on" and "off"

tokens came at the end of the development, so we have three “Actions” tokens and three “System Controls” tokens. The tokens would understand the people, if they would deal more time with our prototype. Then they would face problems where they would find these tokens useful. Instead of the two tokens, we could have offered other tokens, which might also not have been self-explanatory.

Finally, as a hypothetical question, we can ask ourselves now, must the people be able to understand within 3-5min? This could be accomplished using fewer tokens, but it would come at the price of having less features. As an alternative, we could offer “advance tokens”, which are reserved for special cases, but these tokens are not so self-explanatory, but require more experience in dealing with TOP (compare in computer programs: quick / easy vs. expert mode).

Finally, we can say that we can answer our research question *“can we support end-users of different skill levels in programming the IoT using tangible and smart objects?”* with “yes”. In response to the questions *“are the end-users interested in our approach to this challenge in principal?”* we can say definitely yes, the participants appreciated to work directly with the objects in the physical world instead of using an app to choose the desired device in a virtual list. To a certain level we could empower people to program their own devices, but to little surprise we had to accepted that people are generally more interested in simple links of sensors and actuators than in complex-nested programming. The reason for this is, that programming with tangible reaches quite fast its boundaries (see also below), because of the scalability: every function must be represented with a token and the users have to remember the exact procedure and what meaning the token have (we have seen this behavior during the evaluation, specially with the older participants, where they could not remember all meanings of the tokens). Also, involving too many tokens the system can reach physical clutter (cf. section 6.5). Nevertheless, we believe that with our playful interaction approach programming can extent or complement textual or visual programming, because the users do not have to deal with programming languages, which are often difficult to learn.

We have already addressed our research question *“what kind of IoT applications or needs would the end-users be interested in?”* in the Findings (see section 5.4), where the participants told us, what they were interested in. We also summarized their interests in this section. To reflect on these interests briefly, we think that our project has a lot of potential to give non-technology people the ability to control their home automation environments, as our approach does not exist in this form on the market. Instead, always a computer/smartphone is needed and some people do not know how to deal with such apps. We think, it is more intuitive to work directly with the objects, where the user can choose a real object on their own instead from a virtual list. Beyond home automation this approach could also be used for visualizing programming in a tangible way and it could support kids to gain their first programming experience or also experience about IoT in a fun way, because as Lechelt, Rogers et al. (2016) highlighted, IoT is arriving and surrounding us, but teachers in schools do not teach about novel technology like IoT in the classroom (Lechelt, Rogers et al. 2016). We think that our approach could bring computational thinking to children and with our approach we could motivate the teachers to use IoT technology at school. As already mentioned in the introduction (see chapter 1), manufactures have great power through their IoT

products, because often the smart devices are cloud based and they can be “disconnected” them from the server, and as a consequence the users cannot use the devices anymore. In our system the devices are local and the user has the “power” over them as there is no possibility to control the devices from outside (internet). Last but not least, we think that our approach could be very interesting in combination with a service like “IFTTT” for complex usages.

We go on to once more related our approach to previous work.

6.3 Comparison with the Previous Work

With respect to the previous prototype (which was developed prior to this thesis), the interaction has been fundamentally changed. Before that, the modules also had a button, allowing the user to select the desired sensors or actuators. The interaction with the button has been replaced by our controller, with which additional actions can be also selected. The reason why we have replaced the button is that on the one hand we want to hold in our hands a “virtual pointer” with which we can select elements, but on the other hand the technique should be hidden and not immediately apparent to the user - so the system itself should remain invisible, which actually corresponds to the definition of ubiquitous computing. Only by means of a controller does the “selection” (i.e. that it is technically possible) again become visible. Furthermore, our new concept should also illustrate that the devices do not need an additional button to select it. For example, with a light switch mounted on the wall, we do not want to have an extra button to select the light switch.

We have tried to solve the “and” and “or” problem so that these terms are not used directly if possible, since we have already thought in advance that these terms will be difficult to understand for non-technical people. In the previous prototype, we defined the logical link in a program so that all sensors are linked with “or” and all actuators with “and”. At that time, we thought that it was “intuitive” for the users that if a sensor detected something (motion, for example), it would trigger all selected actuators. During the design process, however, we were informed by the participants that this link was rather confusing for the participants. That's why we decided that we want to link all sensors and actuators together in a program with “and” and link each new program with “or”. This approach was understandable to the young students, unfortunately we have to admit that we had less success with this approach with older participants. Nonetheless, during the study, we observed how they would deal with this issue and thus we learned what approaches we could implement in the next iteration (see Future Directions). An open topic before this thesis was whether we need additional feedback in the form of RGB LEDs or is our feedback to the user sufficient. All participants noticed and were also aware that the RGB LED color changed during the procedure and therefore understood when they chose a module or it was in use. We gave our controller additional acoustic feedback, which was not always heard by all participants, but we can still say it was necessary, since most participants understood that only after an acoustic signal the module was recognized by the system. Nevertheless, if the case occurred that a module was not detected by the controller (i.e. no beep) and the participant continued to work anyway and at the end wondered why their linking did

not work, all participants have noticed that in one module (which was not detected), the RGB LED color has been incorrectly lit. Therefore, we believe that a combination of the two feedbacks has produced the best results. Finally, the challenge was how to enter numerical values in our system (for example, to specify a temperature value when the sensor module should trigger). After the controller lost its display during the design process, we decided to enable numeric input directly on the modules if the module needed an input (for example, a pushbutton does not require an input).

Another difference to the procedure or programming is, that the previous prototype is based on the recording of sequences of actions and reactions using the modules. In this thesis, we use "Add Program" to tell the system which sensors and actuators to add to a program and link them together with "and". In contrast, in the previous prototype we had a "Record" button (and complementarily a "Play" button) that started the recording procedure. During recording, the user could select the desired sensors and actuators by pressing the designated button on the module (note: some sensors like the "distance sensor" did not have a button, for example, the user just had to hold his hand in front of the sensor, so the system detected a change in the measured sensor values and this sensor was then used in the program). Once the user has started the record state and has selected a sensor or actuator, the user had a certain cycle time in which the user could specify which modules the user would like to add to the program. The cycle time was stopped after the user did not select any new sensors or actuators after 10 more seconds, so the program was "completed" and stored in the system, after which a new program was automatically started by the system. The downside was that the user had to select certain modules in a certain amount of time. Thus, if the modules were far away from each other and the user does not get fast enough from one module to the other module, the next module will be saved in a new program (since the old program has expired). In the current prototype of this thesis we do not offer a recording or play function, whereas, only the sensors and actuators can be selected in this thesis, the order will not be recorded and there is no certain time to select the desired modules. Hence, with this thesis, we wanted to show a "new" basic concept for the selection of sensors and actuators and therefore decided to start with the simplest version (instead of offering a record-mode, which would be technically easy to implement by simply offering a "Play" and a "Record" token, and maybe to bypass the cycle time a "Stop" token for stopping the recording). We think that there are many different approaches of selecting, which sensors and actuators are linked together, and they all need to be evaluated in the future in order to ultimately find "the best one". We hope that this work will provide a good starting point for further research in this direction. In the following section, we briefly recap what we have learnt with respect to current prevalent paradigms in supporting end-users to program the IoT.

6.4 Prevalent Paradigms in Supporting End-Users to Program the IoT

We researched prevalent paradigms in supporting users to program the IoT in the course of the literature review in order to later use this knowledge to inform our own design. In this section, we briefly recap what approaches are currently being used to

configure the IoT. In the end of this section, we eventually discuss how the ideas that we proposed as part of this thesis differ or correspond to the related work and what they add to the literature.

The technologies of the Internet of Things are becoming smaller and even more ubiquitous, and as IoT becomes more and more integrated into our surroundings, our environment is constantly being connected to the world. By integrating IoT with the environment, IoT aims to make our lives more livable, easier, and more enjoyable, but on the other hand, IoT is also facing challenges. Challenges of how to configure, program or even to use the IoT, where people often have an interest in using such technology, but they are often confronted with entry barriers because of the lack of technical experience (e.g. using a smartphone). This has led us to engage with the market and the research community to explore which paradigms already exist for configuring or installing IoT applications.

We found in the research that for programming or controlling IoT devices often apps are needed on the smartphone. To program more sophisticated IoT applications and without restriction from the manufacturer, developers can use higher-level programming languages, but this requires knowledge about programming. Therefore a number of products support developers with visual programming languages, but also domain specific or scripted languages. Although smart devices can also be controlled via voice control, we have found that “voice control” must also be programmed before it can be used by the user, allowing the user to control their devices. Our approach as proposed in this thesis is significantly different from other researched solutions, and our programming of IoT devices requires physical interaction. We connected the act of programming with the physical world, that means, the user does not need a smartphone or a computer, where they have a list of their IoT devices on the display to select the desired “thing”. In contrast, in our approach, the user does not need a *virtual* list; instead the user directly selects the “thing” directly in the physical world. We believe that our approach can make programming easier because the user does not have to remember the correct syntax, as with a programming language (both higher and visually), and the user does not have to interpret complex algorithms. Therefore, the user does not necessarily have to be able to program to benefit from IoT, as is unfortunately common in the marketplace. We think that our approach spares people from learning a programming language. Nevertheless, we also believe that our approach will allow users to gain initial programming experience because the results have shown that it visualizes/embodies programming routines in a tangible way.

6.5 Boundaries of Tangible Programming and Computing

During our design process, we used to develop our prototype towards a kind of a “smartphone” due to the use of a display on the controller, as the participants reported. After trying to represent the functions on the display with tokens, we came across a few limitations. Of course, tangible programming (TP) has its boundaries, but nonetheless we view these limits as challenges we have tried to push with our system, and we believe that those boundaries could also be pushed further in future scientific work.

To address our question *"where are the boundaries of employing tangibles and similar concepts for programming the IoT?"* we can claim, that the most obvious problem is the scalability of our TOP system. Therefore the limits of TP are encountered relatively quickly, because every function that normally appears on a display is now represented by a "token". This means that the more functions or complexity we want to represent with our system, the more space (tokens) is needed. This can quickly become unmanageable and then might lead to the user having to deal with finding the right token or function, which on the long run can be annoying for the user, because the user has to move a lot with our controller. Another limitation is that each token as an action represents only one function at a time. To be fair, we could have wrapped the "delay" and "duration" actions in the time module instead of offering each one individually. Thus, Bellotti, Back et al. (2002) argued that using TP for complex commands would require much more effort rather than the user executing the command in a console or terminal. Another aspect is, that on a display, we have the ability to zoom or create multiple subareas (tabs), which allows us to map many "functions" on a surface. In contrast, our tokens have a fixed size. As a result, our space with physical models is indeed limited, that is, at some point we reach a stage where we have no way to add new tokens. As Ullmer and Ishii (2000) already mentioned about this challenge, is that "such systems require a careful balance between physical and graphical expression to avoid physical clutter." (Ullmer and Ishii 2000, p.10) Furthermore, such tokens can also be lost, which limits the user in their operation and this token must be restored with much effort. While it is possible to demonstrate a behavior to the system with TP and thus program sensors and actuators, there is no way to see what is currently or has already been programmed, such as a history, unless we offer a display. Also, the tokens are not mutable, they are rigid and additional feedback to the user can usually only be communicated in the form of an RGB LED, an acoustic signal or by means of haptics (vibrate, for example). The problem here is that the user needs to know the meaning of the feedback (if not spoken words), unlike on a display, where the user can read the feedback in the form of a text message. Also not to be underestimated is "user fatigue", because for example with our controller a lot of movement is necessary, which can be a burden especially for older people. If the controller works directly with the objects (such as a table lamp, fan, etc.), objects may be located in hard-to-reach places, such as the light on the ceiling, which is difficult to reach with the controller. However, this could be remedied quickly by having a token act as a representative of the ceiling light at an accessible location instead of having to touch the light directly on the ceiling with the controller. The disadvantage of this approach, however, would be that we no longer work *directly* with the ceiling light and this can give a completely different programming experience.

Therefore, for question *"to what extent can IoT applications be configured with tangibles in contrast to conventional approaches, e.g., using an app?"*, we can say that TP is very well suited for "simple" sensor and actuator links, but the configuration with additional complex functions can confuse the user since the user must first understand how the functions are applied in a system and the user must also remember the procedure, which can be problematic if the user does not use the system on a daily basis. Too many tokens can confuse the user and make the whole system clutter (because each function is represented by a token). At this point, we could assist the user

by, for example, installing a display on the “Magic Board” that would give the user additional instructions and guide the user. In our prototype we limited the actions to standard functions (like “invert status” or “reset all”). This is also one of the reasons why the participants of the evaluation suggested to create a combination of TOP and an app like “IFTTT”, because in this way we can shift the “complex” linking and configuration to the app. Also the older participants have told us, they are more interested in simple configurations. Therefore, we can claim that the complex configuration is more likely to be used by technology enthusiasts, and we can expect these users to have that experience with smartphones.

6.6 Final Reflections on the Importance of Evaluations (User Feedback)

During this master thesis, we learned how important evaluations can be, because by discussing and interacting with the participants, the participants gave us insights that we would not even have anticipated. Furthermore, we learned how important it was for us to develop an (interactive) prototype instead of describing the participants our project with words. By using an interactive prototype, all participants could imagine more easily what our ideas and motivations were. We think that the use of such prototypes is extremely important, because people make different experiences with technology during their life and thus people will imagine different things and other things than intend if we “only” describe our concepts.

The evaluations also showed us that users have different viewpoints and this can increase the risk that we will fail with our prototype design. We go that far to speak of “failure”, because we observed in the investigations with the participants that they felt insecure and did not criticize the prototype openly. Only, after we encouraged them to freely interact with the prototype, and they made first positive experiences with “interactivity”, they found the courage to openly address their concerns. In this context, it should also be noted that the participants were watched by us all the time (and also recorded on audio on their approval) and perhaps for this reason did not dare to ask or tell what was incomprehensible.

In line with this concern, it was also important to us to work with experts in the HCI area during the evaluation, because they could give us valuable tips and solutions to push the design process forward without the “fears” of the “amateurs” about providing direct and open critique.

6.7 Future Directions

This master thesis constitutes explorative research drawing on our own designs and on a relatively small number of participants. Even though we used some quantitative data during the evaluation, it is not our intention to infer generalizations from our work nor do we claim that TOP will necessarily be a success when applied by just any arbitrary user. Rather, we suggest that the reader might evaluate the insights of this thesis (including the design proposals) as a qualitative and explorative research contribution.

In the remainder of this section, we describe some aspects that might be worth researching to continue the research agenda around TOP as initiated by this thesis.

For future work, we would consider it important to provide the user with additional feedback and information during the “programming” process. For example, this could be accomplished by mounting a display on the Magic Board that either informs the user what to do as a next step or that helps explaining how the user can perform a certain function.

Also, the “on” and “off” actions should be further optimized in our opinion, because it was not easy for us to make these self-explanatory to all participants, as these tokens were understood differently by many participants. This appeared to be a “true Wicked Problem” (Rittel and Webber 1973) to pick up the design jargon introduced in the Approach and Methods section. For example, a participant requested a feature that allows the user to simultaneously turn on or off all lights in the living room (i.e., all actuators) in which the participant would use either the “on” or “off” token with the controller. Hence, one first idea was to use these tokens to turn the whole system on or off.

Of course, a number of other special functions would be possible (for example, a token to permanently save a program, or a token to transmit the sensor data to a cloud service), but this would increase the space for the tokens and the complexity of the system quickly, and we should therefore carefully consider in future work which tokens are *really* needed.

Furthermore, it would be worth considering to design the “Add Program” token in a different way. We could instead offer two other tokens, a “Start” and an “End” token. With “Start” the user starts a new program and with “End” the user terminates the program. For example, if the user wishes to start a new program, the user first selects the “Start” token, then the user performs their desired linking, and finally selects the “End” token. Also, it would be possible to define, rather than using “Start” and “End” tokens, that a sensor starts a new program. That is, the user first selects the desired sensors, followed by the associated actuators. As soon as the user again selects a sensor, the system automatically starts a new program. However, this would require the user to follow an order, but nevertheless, we have already seen this approach with the older participants, and these examples would then have to be tested to see if the users find one of these approaches to be more intuitive.

As shown in the Findings comparing the two phases, where the most errors were made during testing, the following issues have the potential to be improved. The indication where the user should hold the controller should be less ambiguous, as it often irritated the participants when they held the controller next on the module and there was no response. Furthermore, we could investigate how we could make the logical “and” and “or” links more intuitive by trying different approaches. Possible approaches have already been suggested in this chapter. For the time module we could offer both options, namely that the user either scans individually per function (“delay”, “duration”) or both functions are recognized by the system immediately, since some participants thought that the time module is already considered as an “action” like a token, and not every function individually as an “action”.

As the evaluations have already shown us, the participants of both phases had different ways of thinking, which we have already discussed above. Where the Phase I

participants got on well with our prototype, we noticed that the older participants had a different approach to running our prototype. This was not negative at all, but we believe this should emphasize that the prototype could be tested especially with more older participants and could be tailored to their needs, as they gave interesting insights, as non-technicians think. Instead of older people, we could also test the prototype with children and maybe design the prototype as an educational benefit. On this subject, there was a study of Horn, Solovey et al. (2009) in which tangible and a graphical interface were compared at an exhibition at the Boston Museum of Science. Adults and children were examined, where it came out that especially children engaged in a TP exhibit, an effect that seemed to be especially strong for girls.

6.8 Conclusion

We have developed an alternative approach to programming sensors and actuators and related IoT applications by using tangible computing devices through an iterative, design-based, and user-centered approach. During this iterative design process we gained many interesting insights based on mostly positive as well as some negative user experiences of our 43 participants. Finally, an interactive prototype featuring a network of sensors and actuators including a tangible programming device and (RFID) tokens for configuring functions like delay or inverting has been created to empower people to control their own IoT devices according to their specific needs. We argue based on all evaluations and the design process that people appreciated working directly with the objects as implemented by our concept. With respect to “simple” applications our controller constitutes a good alternative to complement already existing products and approaches in the home automation market. Therefore, to a certain extent we could turn “passive consumers” into “active producers” in the context of home automation, because people could in an intuitive approach link their devices and solve tasks with interactive technology. They did not need a computer or an app as it is a common approach in commercial products, and they were not confronted with high entry barriers because of their lack of technical experience (in particular Phase 2 participants). However, we also found that tangible programming cannot be applied easily to complex tasks, as it is not possible to represent every imaginable function as a piece of hardware (token) or sequences of interactions with different pieces of hardware. This issue is also related to the target group, and depending on the target group we could offer (a multitude of) different tokens. We also found during the evaluation that the participants had different approaches in interacting with our prototype. There remain many different approaches to selecting and programming interactive objects, and we hope that these approaches can be evaluated in further work to find additional appropriate methods of linking IoT elements. One important challenge would be to further investigate how the numbers of tokens can be reduced in such a system.

Nevertheless, it can be concluded that our system is an applicable approach for linking sensors and actuators, both for people with little knowledge of computers and for technology enthusiasts and therefore we hope that this thesis will provide a good starting point for further research in the direction of programming IoT applications by using tangible computing devices.

List of Figures

FIGURE 1.1 A GRAPHIC OF CISCO IBSG (2011) WHICH SHOWS THE NUMBER OF CONNECTED DEVICES IN RELATION TO THE WORLD POPULATION. IN 2010 THERE WERE OVER 12.5 BILLION DEVICES CONNECTED TO THE INTERNET, WHILE THE WORLD POPULATION WAS ABOUT 6.8 BILLION. HENCE, THERE HAVE ALREADY BEEN MORE THAN ONE CONNECTED DEVICES PER PERSON SINCE ABOUT 2008.	1
FIGURE 1.2 WORLDWIDE INTEREST OVER TIME ON GOOGLE TRENDS FOR INTERNET OF THINGS FROM 2004 TO MAY 4, 2017 – NOTE: "NUMBERS REPRESENT SEARCH INTEREST RELATIVE TO THE HIGHEST POINT ON THE CHART FOR THE GIVEN REGION AND TIME. A VALUE OF 100 IS THE PEAK POPULARITY FOR THE TERM. A VALUE OF 50 MEANS THAT THE TERM IS HALF AS POPULAR. LIKewise A SCORE OF 0 MEANS THE TERM WAS LESS THAN 1% AS POPULAR AS THE PEAK." (GOOGLE 2006).....	2
FIGURE 2.1 – EXAMPLES OF COMMERCIAL IoT PRODUCTS: RIGHT PHILIPS HUE AND LEFT NEST THERMOSTAT	10
FIGURE 2.2 - EXAMPLES OF IoT PRODUCTS FOR EDUCATION: RIGHT LITTLEBITS AND LEFT .NET GADGETEER	12
FIGURE 2.3 – EXAMPLES OF VISUAL (WEB BROWSER) TOOLS FOR WIRING THE IoT: RIGHT NODE-RED AND LEFT MODKIT (MILLNER AND BAAFI 2011)	16
FIGURE 3.1 - ITERATIVE DESIGN PROCESS OF THIS THESIS WHICH LED TO THE FINAL PROTOTYPE AND CONCLUDING EVALUATION.....	24
FIGURE 3.2 – MAKERBOT REPLICATOR 3D-PRINTER FOR RAPID PROTOTYPING TO QUICKLY CREATE PHYSICAL OBJECTS AND WHICH WAS ALSO USED TO MANUFACTURE THE PROTOTYPE.....	30
FIGURE 3.3 – ARDUINO NANO WHICH WAS USED FOR THE INTERACTIVE PROTOTYPES.	32
FIGURE 4.1 – PRESENTATION OF THE ITERATIVE DESIGN PROCESS. AFTER CONCEPTION, A PROTOTYPE WILL BE BUILT WHICH WILL BE TESTED WITH PARTICIPANTS, THUS GAINING NEW INSIGHTS LEADING TO A NEW OR IMPROVED CONCEPT. AFTER THIS PROCESS HAS BEEN REPEATED SEVERAL TIMES, THE PROCESS ENDS WITH A FINALIZED EVALUATION FOLLOWED BY THE CONCLUSION (CF. FIGURE 3.1).	38
FIGURE 4.2 – CHRONOLOGICAL TIME LINE OF THIS THESIS WITH MAJOR DESIGN PHASES.	39
FIGURE 4.3 – ILLUSTRATION OF "PLAY THE RECORD" (INTERNAL/UNOFFICIAL PROTOTYPE NAME) WHICH ENABLED TO CONNECT SENSORS AND ACTUATORS TOGETHER IN A SIMPLIFIED FASHION. USING THE BUTTONS ON THE MODULES, THE USER WAS ABLE TO SELECT THE SENSORS AND ACTUATORS HE WANTED TO USE. SOME SENSOR MODULES (E.G. DISTANCE SENSOR) HAD NO EXTRA BUTTON, INSTEAD, THESE SENSORS WERE DETECTED BY THE SYSTEM AFTER AN EVENT WAS TRIGGERED. NOTE, THERE ARE FUNDAMENTAL DIFFERENCES BETWEEN "PLAY THE RECORD" AND THE DESIGN CONCEPTS AS IMPLEMENTED IN THIS MASTER THESIS.	40
FIGURE 4.4 - PICTURES FROM THE WORKSHOP. RIGHT THE WORKSPACE AND LEFT SOME COLLECTED IDEAS ON A WHITEBOARD	42
FIGURE 4.5 – EXPLANATION WITH ILLUSTRATIVE AGENTS: THE AGENTS REPRESENT SMALL PARTS OF A TECHNICAL SYSTEM. EACH AGENT HAS A SPECIAL ABILITY ("DUDETTE WITH STETHOSCOPE" = MICROPHONE, "SCREAMING DUDE" = MAKING NOISE, , "FLASHLIGHT DUDETTE" = LAMP, "DUDE WITH A SPYGLASS" = DETECTING MOVEMENT (FROM LEFT TO RIGHT)) (CC BY 3.0 AT DOODER 2017). THE CHALLENGE FOR THE WORKSHOP PARTICIPANTS WAS: HOW DO WE TEACH THE AGENT WHEN SOMETHING HAPPENS, THAT THEY RESPOND IN A CERTAIN WAY?	43
FIGURE 4.6 – SOME SKETCHES FOR THE FIRST PROTOTYPE. THE INITIAL IDEA WAS TO DESIGN THE PROTOTYPE AS A FORM OF A WRENCH.....	46
FIGURE 4.7 - THE FIRST LO-FI PROTOTYPE. THE "LOLLIPOP-LOOKING" WOODEN ELEMENTS ARE THE LINKS, THE RECTANGULAR WOODEN ELEMENT IS THE CONTROLLER AND THE SMALL PAPER CARDS REPRESENT THE DISPLAY. THE RED ARROWS SHOW WHERE THE LINKS COULD BE PUT IN. IN ADDITION, PLAY-DOH SHOULD REPRESENT A SENSOR OR AN ACTUATOR.	49

FIGURE 4.8 - SNAPSHOT WHERE A PARTICIPANT INTERACTS WITH THE PROTOTYPE.	50
FIGURE 4.9 - THE FIRST MI-FI PROTOTYPE WITH ALL ITS COMPONENTS	56
FIGURE 4.10 THE POWER BANK ACTED AS AN INTERFERENCE FACTOR ON THE ANTENNA, WHICH LED TO REDESIGN THE PROTOTYPE.	57
FIGURE 4.11 - REDESIGNED PROTOTYPE WHERE ALL COMPONENTS ARE SEPARATED	58
FIGURE 4.12 – SNAPSHOTS OF THE DISPLAY: RIGHT “PUSH BUTTON” IS RECOGNIZED – LEFT: “EXPERT MODUS” OF A LAMP	59
FIGURE 4.13 - IMPROVED FORM FACTOR OF THE PROTOTYPE BY USING THE ARDUINO NANO INSTEAD OF AN ARDUINO MEGA.	60
FIGURE 4.14 - 3D MODELED CASINGS FOR THE BUZZER (LEFT) AND LIGHT SENSOR (RIGHT)	65
FIGURE 4.15 – WORK IN PROGRESS: THE TOP PART OF THE CONTROLLER, THE LEFT PICTURE SHOWS HOW THE WIRING LOOKS INSIDE AND ON THE RIGHT WHERE THE TOP PART WAS MELTED TOGETHER.....	66
FIGURE 4.16 - OVERVIEW OF ALL COMPONENTS OF THE FINAL PROTOTYPE. THE ORANGE CASINGS REPRESENT THE ACTUATORS, THE GREEN CASINGS THE SENSORS, AND THE YELLOW CASINGS THE SYSTEM CONTROLS AND ACTIONS, AS WELL AS THE CONTROLLER (MAGIC WAND). THE WHITE CASE REPRESENTS THE STOPWATCH, WHICH WAS USED IN THE EVALUATION TO TIME THE TASKS.	67
FIGURE 5.1 - ILLUSTRATES THE DISTRIBUTION IN TABLE 5.2- HOW OFTEN A SCALE HAS BEEN SELECTED PER QUESTION	74
FIGURE 5.2 - SNAPSHOT FROM THE ROOM WHERE TOP WAS PRESENTED TO THE PARTICIPANTS.....	76
FIGURE 5.3 - DESCRIPTIVE STATISTICS OF PERFORMANCES IN SECONDS FROM PHASE I, HOW LONG EACH PARTICIPANT NEEDED TO SOLVE A TASK.	77
FIGURE 5.4 - BOX-WHISKER-PLOTS OF THE AVERAGE PERFORMANCES IN SECONDS PER TASK FROM PHASE I	77
FIGURE 5.5 - SNAPSHOT OF THE IN-SITU EVALUATION WHERE A PARTICIPANT TRIES OUT THE PROTOTYPE. ...	81
FIGURE 5.6 - DESCRIPTIVE STATISTICS OF PERFORMANCES (IN SECONDS) FROM PHASE II, HOW LONG EACH PARTICIPANT NEEDED TO SOLVE A TASK.	82
FIGURE 5.7 - BOX-WHISKER-PLOTS OF THE AVERAGE PERFORMANCES IN SECONDS PER TASK FROM PHASE II	82
FIGURE 5.8 - DESCRIPTIVE STATISTICS OF AVERAGE PERFORMANCES (IN SECONDS) FROM PHASE I AND II	87

List of Tables

TABLE 2.1 - LIST OF COMMERCIAL IOT PRODUCTS	11
TABLE 2.2 - LIST OF IOT RESEARCH PRODUCTS IN THE ACADEMIA AND FOR EDUCATION	14
TABLE 2.3 - LIST OF IOT AND TANGIBLE EUD APPLICATIONS	18
TABLE 2.4 – SUMMARY OF PREREQUISITES (BOTH KNOWLEDGE AND EQUIPMENT) FOR OPERATING MOST IOT DEVICES.	19
TABLE 3.1 – OVERVIEW OF ALL PHASES IN WHICH THE RESEARCH METHODS WERE APPLIED	34
TABLE 3.2 DEMOGRAPHIC DATA OF ALL INVOLVED PARTICIPANTS.....	36
TABLE 4.1 – OVERVIEW OF THE 25 PARTICIPANTS OF EACH PHASE IN THIS THESIS.	39
TABLE 4.2 – WORKSHOP-AGENDA	43
TABLE 4.3 - OVERVIEW OF ALL NECESSARY ACTIONS TO COVER AS MANY APPLICATIONS AS POSSIBLE	47
TABLE 4.4 - AGAIN AN OVERVIEW OF ALL THE REACTIONS WHICH WERE DEFINED	48
TABLE 4.5 - OVERVIEW WHICH ATTENDEES PARTICIPATED IN THE FIRST TEST RUN.	50
TABLE 4.6 - OVERVIEW WHICH ATTENDEES PARTICIPATED FOR THE SECOND EVALUATION.	61
TABLE 4.7 - ALL PROVIDED FEATURES WHICH WERE DEVELOPED	69
TABLE 5.1 - OVERVIEW OF THE PARTICIPANTS AND THEIR ANSWERS TO THE SURVEY AS A COLOR-CODED TABLE, WHERE DARK GREEN MEANS STRONGLY AGREE (CODED AS 5) AND WHITE DENOTES STRONGLY DISAGREE (CODED AS 1). NOTE: THE FIRST SEVEN PARTICIPANTS IN THE TABLE HAVE TESTED THE PROTOTYPE ON THEIR OWN (M1-M7), THE REMAINING PARTICIPANTS WAS PRESENTED THE PROTOTYPE, BUT THEY DID NOT TRY IT OUT (M8-M27).....	73
TABLE 5.2 - HOW MANY TIMES A SCALE HAS BEEN SELECTED WITH THE RESPECTIVE QUESTION	73
TABLE 5.3 –THIS TABLE PRESENTS ALL THE TASKS ASSIGNED TO THE PARTICIPANTS. SINCE WE ARE TALKING ABOUT PROGRAMMING IN THIS PROJECT AND TOP SHOULD INDEED REPLACE THE ACTUAL PROGRAMMING, WE ADDITIONALLY WANT TO ILLUSTRATE THE TASKS IN THE FORM OF A PSEUDOCODE EXEMPLIFYING HOW THIS TASK IS TO BE UNDERSTOOD IN THE CONTEXT OF A PROGRAMMING LANGUAGE.	75
TABLE 5.4 – FREQUENCY OF HOW OFTEN AN ISSUE WAS ADDRESSED BY PARTICIPANTS DURING THE TRAINING. DARK RED IS VERY COMMON AND PALE RED RATHER RARE.	88

Acronyms

DSL	Domain Specific Language
e.g.	exempli gratia
et al.	et alii
etc.	et cetera
EUD	End User Development
GPL	General Purpose Languages
GPS	Global Positioning System
HCI	Human Computer Interaction
i.e.	id est
IFTTT	IF This Then That
IoT	Internet of Things
LED	light-emitting diode
NFC	Near Field Communication
RFID	Radio-Frequency Identification
RGB	red green blue
RQ	research question
RTD	Research through Design
TEI	International Conference on Tangible, Embedded and Embodied Interaction
TOP	thingy oriented programming
TP	tangible programming
UCD	user-centered design
USP	unique selling point
WLAN	wireless local area network

Appendix

Design Workshop Appendix

Handout

Dear Participant

thank you so much for contributing - this is highly appreciated!

The goal of this workshop is to invent some interactions for the Thingy Oriented Programming paradigm (short: TOP; more details will be explained). To do so, we will join four activities.

#1 Activity and #2 Activity:

Relax! You are not supposed to actively do anything here ☺ Instead, you will be given a brief presentation about the Internet of Things (IoT) and Thingy Oriented Programming (TOP) to make sure that everyone talks about the same thing in the end.

#3 Activity:

Now we need your help! You will be given a number of sensors/actors. Please select those sensors and actors, which are most relevant/interesting for IoT appliances in your opinion. Be prepared to explain your selection.

#4 Activity:

This is the most important exercise. Please invent a concept for programming your selection of sensors and actors. For example, how can you program a buzzer to be triggered by a specific button? Your concept should be as versatile as possible (allowing to program powerful applications for different use cases), while at the same time, it should be very intuitive and simple (no computer skills should be required and no computers).

#5 Activity:

This is the final task. You will be given a number of mounts for sensors/actors for your inspiration. Please make a selection of the best options for mounting TOP-modules and/or invent your own mounts!

Brainstorming - Possible Tasks

Reminder (Wattage, Power)

Scenario: A device (e.g. water boiler, coffee machine, washing machine, etc.) has been switched on. Meanwhile, a person reads a book or checks their emails. The person is so engrossed in their thoughts that they totally forget that they recently switched on the water boiler and forgets to pour the water into the teapot.

Solution: As soon as the water boiler has boiled the water, a "smart thing" (actuator) flashes or sounds which is clearly visible to the person and reminds them that the water has been boiled. The smart thing must be placed correctly for the person to see or hear it. The trigger is located directly on the socket where a "smart thing" (sensor) is connected between the plug (from the water boiler) and the socket. This smart thing is able to measure the consumption and thus detects that something has been turned on. Once the consumption is at almost 0, the actuator is informed (this triggers a visual or acoustic signal).

Other examples are also possible, for example, that the TV and computer are still in standby mode (reminder to disconnect devices from the power supply). So you could configure a timer that after the television is turned off after 10 pm, 15 minutes later the TV and the hi-fi system will be disconnected from the power supply (until 5 pm the next day).

Implementation: the sensor is quite easy to program, since it must be coupled with only one actuator. After the consumption is almost 0, a signal is sent to the coupled actuator either immediately or with a delay.

Reminder for airing / Comfort in the room

Scenario: a person sits in the room and works intensively. By the constant exhalation the air in the room is "consumed". The humidity is hardly perceptible for the person. This results in too short aeration, which promotes mold formation in the home. Continuous airing is a waste of energy during the heating period, so it should be ventilated regularly at short intervals.

Solution: As soon as a critical relative humidity value is reached in the room, a "smart thing" will remind you to ventilate. The duration of the ventilation can either be time-dependent (e.g. at least 8min) or until the relative humidity is within the "normal range".

Implementation: a weather station (for example with Arduino modules) which measures the room temperature and the relative humidity. Once a critical value is reached, there is either an acoustic or visual signal.

Upcoming problems:

How to define the "critical area"? Default values?

How should the "smart thing" recognize that was already aired? Do you touch the smart thing and it starts a countdown?

Stimulation Rerouting

Scenario: Karl is deaf and expects today a package from the post office. Because of his disability, he would not notice that the postman rings at the door, so he needs a visual or haptic signal.

Solution: a smart thing maps the acoustic signal into a visual signal. For this purpose a smart thing (sensor) is set at the door. As soon as a knock or doorbell is registered, another smart thing (actuator) is signaled, which then either starts to flash or vibrate.

Counter (like an egg timer)

Scenario: Person would like to be reminded in x-min (e.g. pizza in the oven).

Solution: once the counter has counted down to 0, a smart thing flashes or sounds, which is clearly visible to the person and reminds them.

Upcoming problems:

How do I set the time?

Important criterion: setting the time must be simpler than with an egg timer, otherwise it will not be used.

Other short tasks

Scenario: A child wants to build an alarm system on their own, for example when the cat is under their bed.

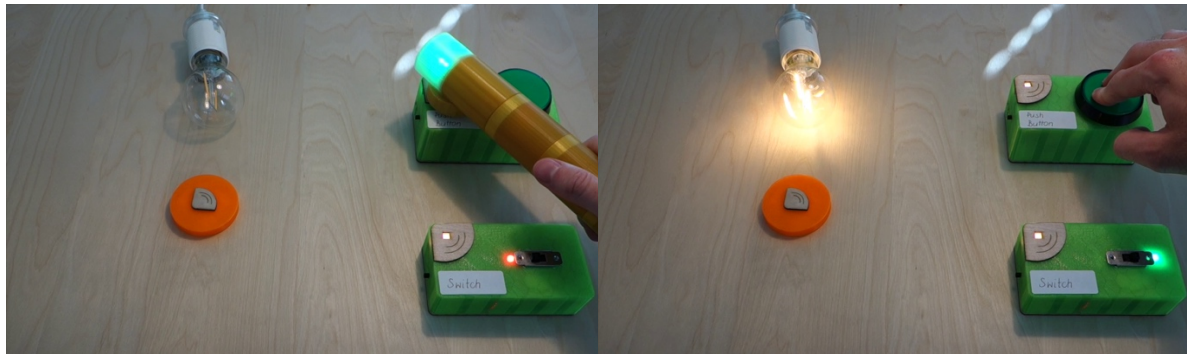
Scenario: A person would like to know if the toilet is occupied.

Scenario: If it is warm outside, a red LED should light, otherwise a blue one.

Scenario: An older person should be reminded that they should take their medication

Findings Appendix

Snapshots of the Introduction-Video



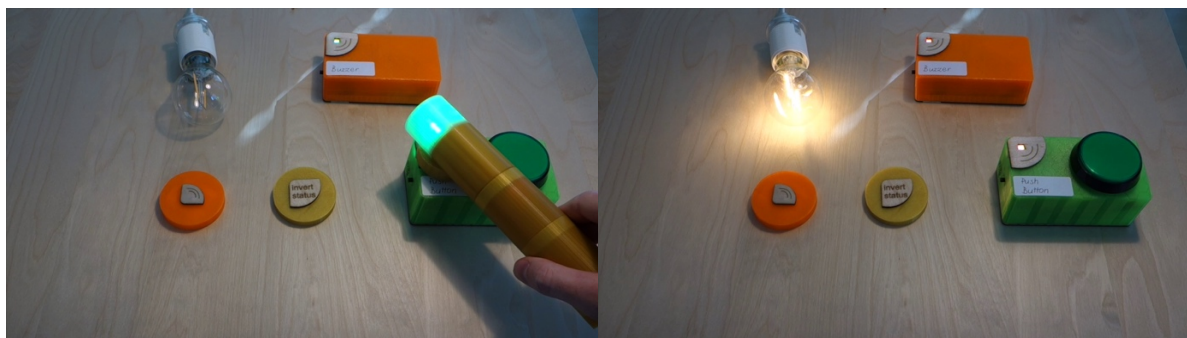
Linking switch and pushbutton with lamp (orange token)



Linking switch or pushbutton with lamp (orange token)



Inverting lamp



Inverting pushbutton



Setting duration and delay

Beginners' Day, 2. Oktober 2017



Feedbackbogen

zu TOP – Thingy Oriented Programming

von Daniel Dudo, BSc.

(Bogen Nummer: _____)

Liebe Beginner,

ich danke recht herzlich für Euer Feedback! – Dieses würde ich sehr gerne für meine Diplomarbeit verwenden, um den Prototypen weiter zu verbessern bzw. wissenschaftlich auszuwerten.

Mit der Abgabe dieses Fragebogens, gibst Du Dich damit einverstanden, dass ich Deine Daten in die Diplomarbeit und zugehörige wissenschaftliche Arbeiten einfließen lassen darf. Wir verwenden nur **anonymisierte Daten**, daher bitten wir Dich auch nicht um die Angabe Deines Namens. Wenn Du nicht einverstanden bist, gib diesen Bogen bitte einfach *leer* ab.

1. Alter: _____
















2. Geschlecht: _____

3. Welche Schulform hast Du besucht? (z.B. HTL): _____

Kannst du folgenden Aussagen zustimmen?

4. Ich habe schon viel über Smarthomes gehört:	 strongly agree agree neutral disagree strongly disagree
5. Ich habe schon viel programmiert in einer Computersprache wie etwa C/C++, Java, Php oder ähnliche Sprachen:	 strongly agree agree neutral disagree strongly disagree
6. Programmierkonzepte wie IF/ELSE, FOR, WHILE sind mir sehr vertraut:	 strongly agree agree neutral disagree strongly disagree
7. Der vorgestellte Prototype (TOP) ist meiner Meinung nach sehr intuitiv:	 strongly agree agree neutral disagree strongly disagree

BITTE WENDEN

8. Ich würde ein fertig-entwickeltes System/Produkt zu den vorgestellten Prototypen (TOP) selbst verwenden:	     strongly agree agree neutral disagree strongly disagree
9. Ich würde ein fertig-entwickeltes System/Produkt zu den vorgestellten Prototypen (TOP) für Menschen ohne Technik-Bezug empfehlen (z.B. manche ältere Menschen):	     strongly agree agree neutral disagree strongly disagree
10. Ich besitze bereits Smarthome-Produkte wie etwa die (Philips) HUE:	     strongly agree agree neutral disagree strongly disagree

11. Welche Smarthome Produkte besitzt Du bereits?: _____

12. Möchtest Du mir sonst noch was mitteilen?: _____

DANKE!

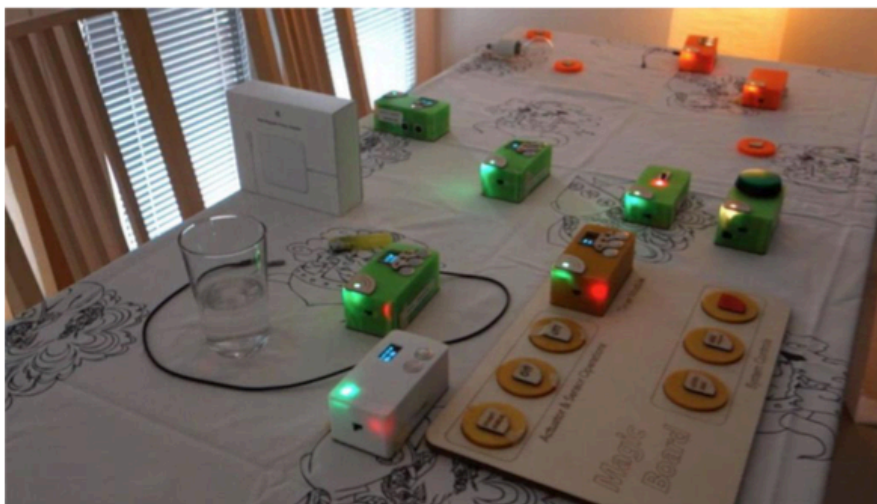
**Projekt: Thingy Oriented Programming**

Daniel Dudo
Diplomarbeit

Diese Arbeit geht der Frage nach, wie für das **Internet of Things** einfache Anwendungen möglichst schnell und einfach **prototypisch entwickelt** werden können.

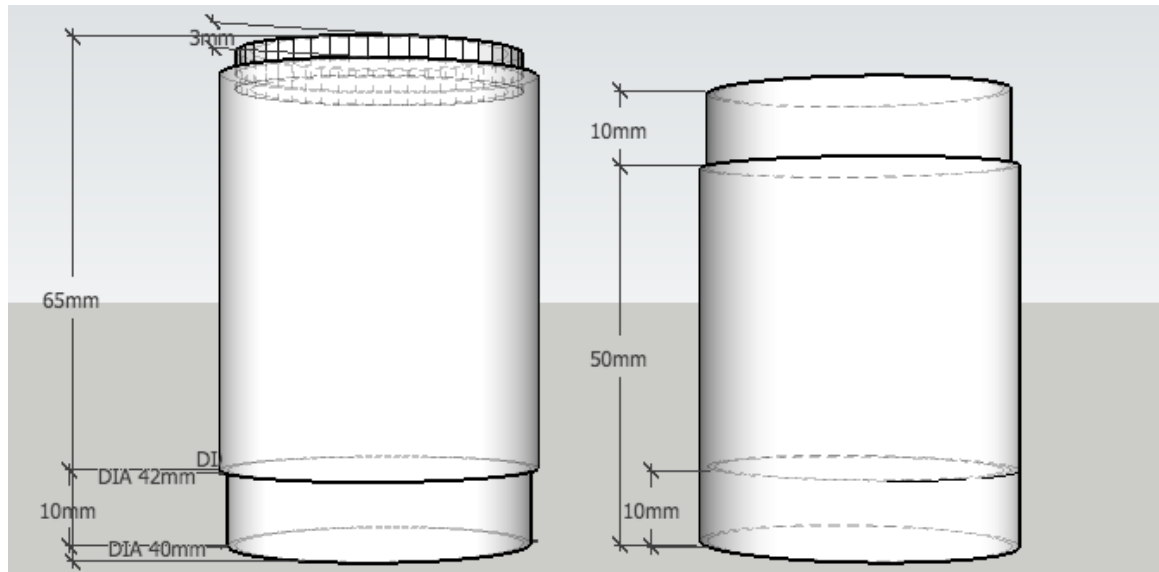
Wesentlicher Wert wird dabei auf den **Verzicht von Programmierkenntnissen** gelegt, um auch **Laien** die Möglichkeit zu geben, am **Entwicklungsprozess gestalterisch** teilzuhaben.

Bitte geben Sie Daniel **Feedback**, diese Präsentation ist **Teil seiner Diplomarbeit**.

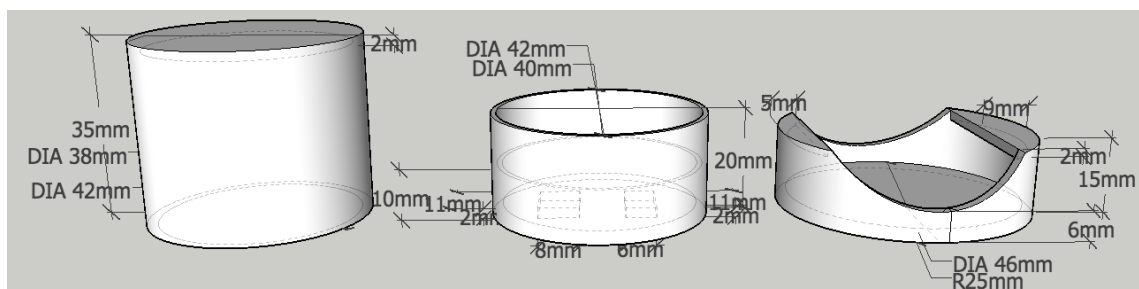


Institut für Gestaltungs- und Wirkungsforschung, HCI Group

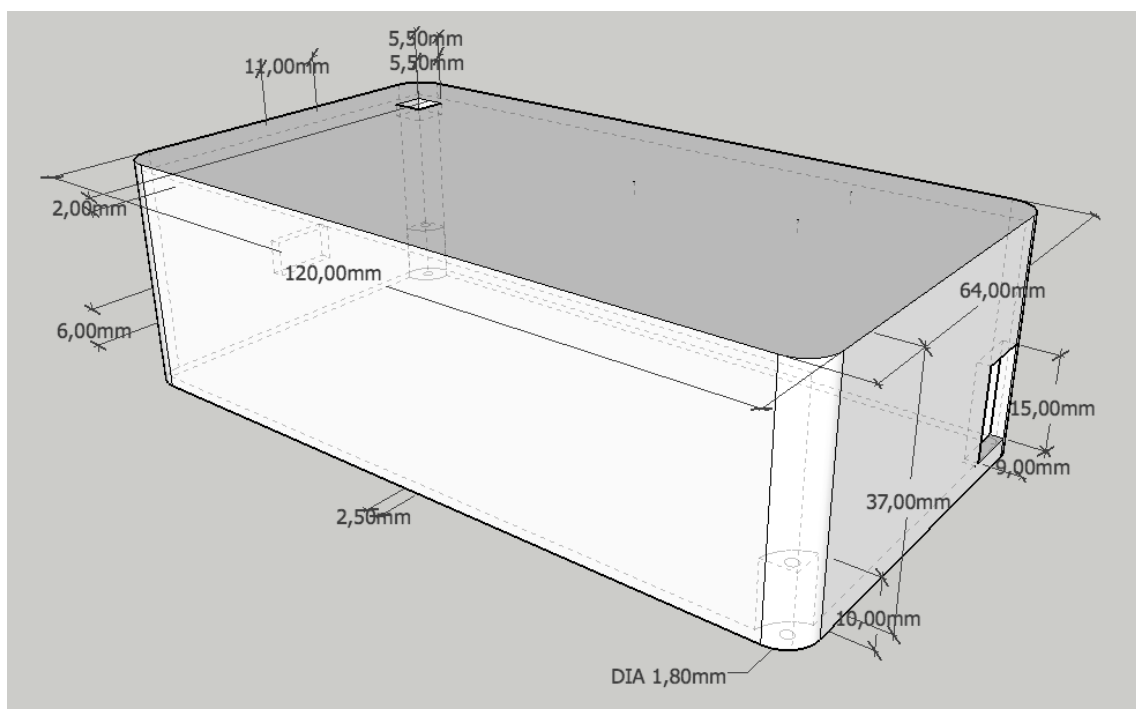
Models Appendix



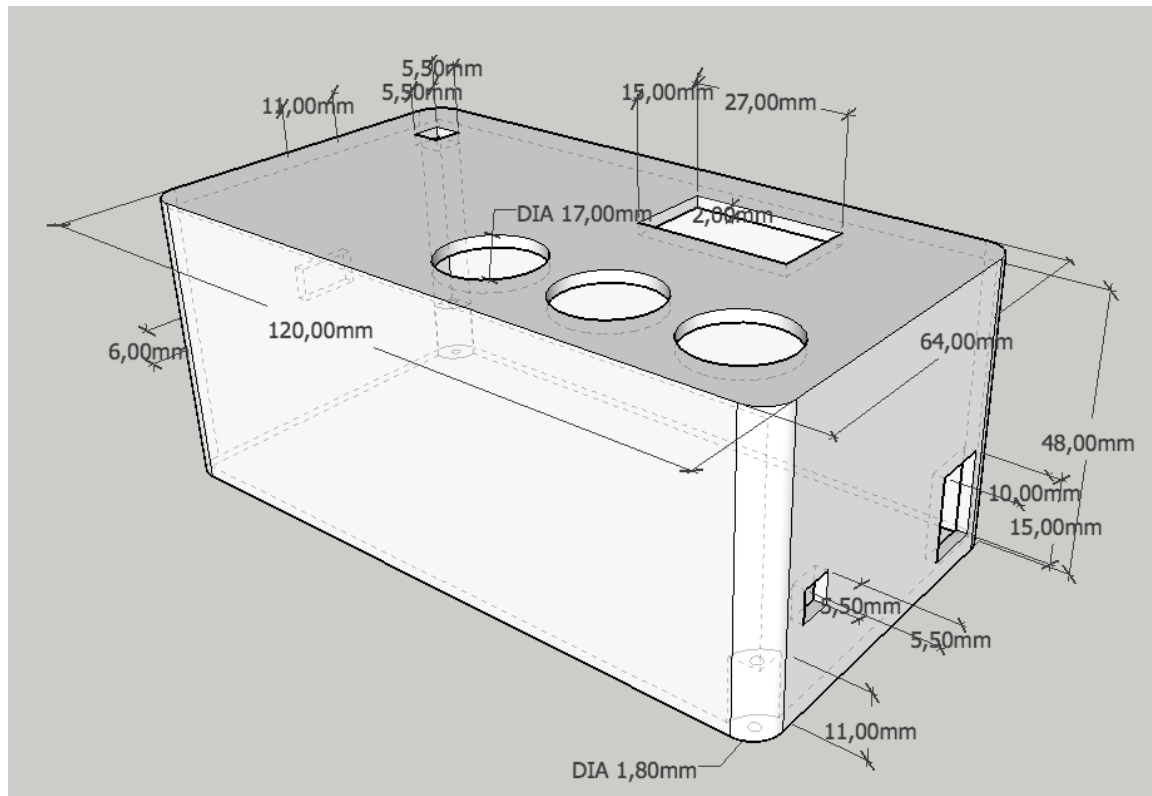
controller - attachment for light part (left); extension part (right)



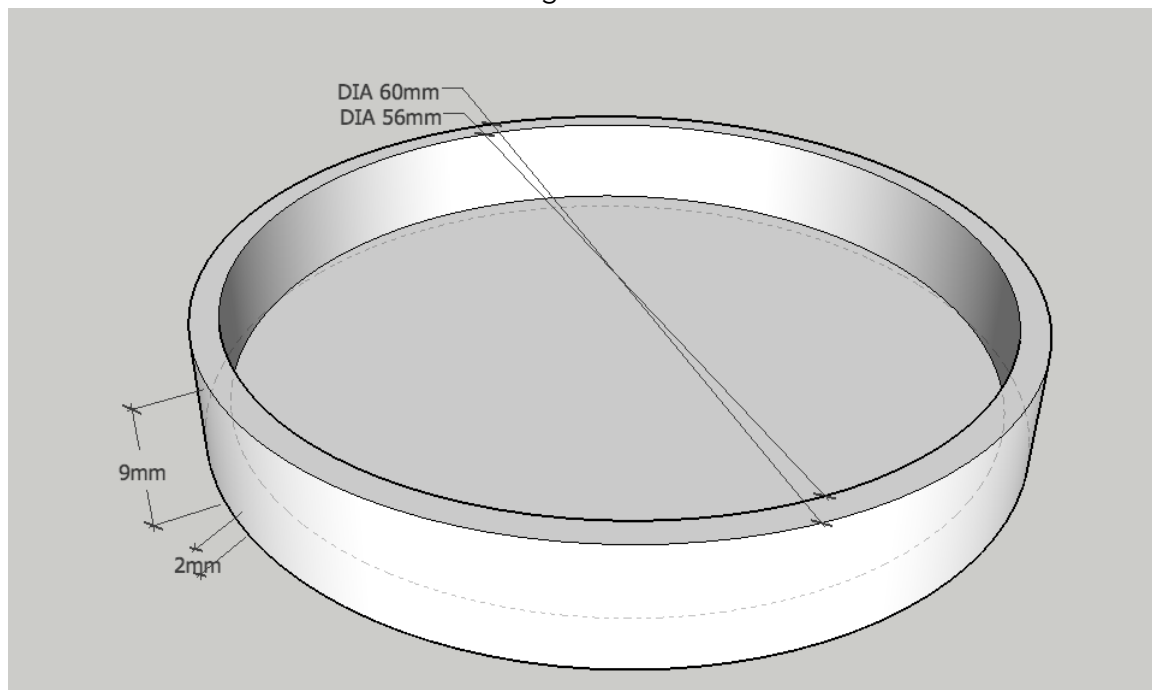
controller – light part (left); bottom part (middle); antenna part (right)



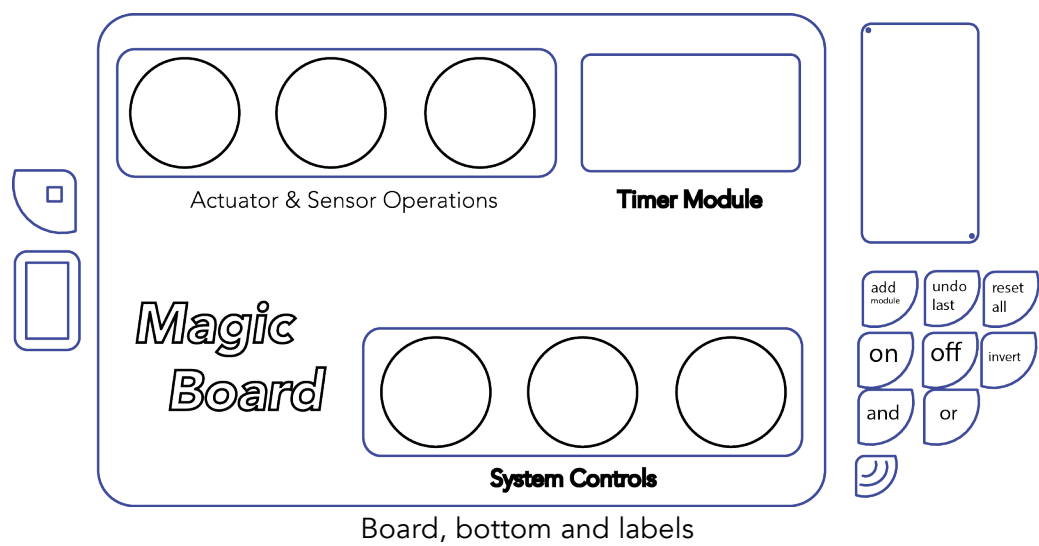
small boxes



large boxes



token



Constants Appendix

(Sub)types

```

/*
 * SubType Codes
 */
public static final int RESET = 133;
public static final int RFID_CODE = 99;
public static final int NEW_EVENT = 122;
public static final int EXPERT_REQ = 111;
public static final int BTN_1 = 150;
public static final int BTN_2 = 151;
public static final int BTN_3 = 152;
public static final int BTN_4 = 153;
public static final int NO_ACTUATOR = RECORDMODE;
public static final int SET_ACTUATOR = 34;
public static final int SPECIAL = 11;
/*
 * Subtypes from Mysensors.org
 */
public static final int V_DISTANCE = 13;
/*
 * Modes
 */
public static final int MODE = 111;
public static final int PLAYMODE = 55;
public static final int RECORDMODE = 33;
public static final int EVENTMODE = 44;
public static final int DEFAULTMODE = 77;

```

Device IDs

Range:

1 – 99: System and Actions
 100 – 199 Sensors
 200 – 254 Actuators

Name	ID
Unknown	-1
Controller (old)	1
Wand	2
Time module	10
Stopwatch	11
Reset Token	12
Undo Token	13

Add Program Token	14
Invert Status Token	15
Off Token	16
On Token	17
Special Token	25
Switch	101
Pushbutton	102
Light Sensor	103
Temp. Sensor	104
Distance Sensor	105
Motion Sensor	110
Remote Socket A	201
Remote Socket B	202
Remote Socket C	203
Buzzer	204
RGB Lamp	205
Twitter Token	206

Bibliography

Amazon (2015). "Amazon Echo is a hands-free speaker you control with your voice." Retrieved 10.02.2017, from www.amazon.de/echo.

Apple (2014). "HomeKit." Retrieved 02.03.2017, from <http://www.apple.com/ios/homekit/>.

Arduino (2005). "Open-source electronic prototyping platform enabling users to create interactive electronic objects." Retrieved 20.02.2017, from <https://www.arduino.cc/>.

Atzori, L., et al. (2010). "The internet of things: A survey." Computer networks **54**(15): 2787-2805.

Ballagas, R., et al. (2003). iStuff: a physical user interface toolkit for ubiquitous computing environments. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM.

Barricelli, B. R. and S. Valtolina (2015). Designing for end-user development in the internet of things. International Symposium on End User Development, Springer.

Bayazit, N. (2004). "Investigating design: A review of forty years of design research." Design issues **20**(1): 16-29.

Bdeir, A. (2009). Electronics as material: littleBits. Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, ACM.

Bellotti, V., et al. (2002). Making sense of sensing systems: five questions for designers and researchers. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM.

Blackstock, M. and R. Lea (2012). WoTKit: a lightweight toolkit for the web of things. Proceedings of the Third International Workshop on the Web of Things, ACM.

Blackwell, A. (2002). What is programming. 14th workshop of the Psychology of Programming Interest Group.

Blikstein, P., et al. (2016). "Project Bloks: designing a development platform for tangible programming for children." Position paper, retrieved online on: 06-30.

Blomberg, J. and M. Burrell (2009). "An ethnographic approach to design."

Bluetooth (1994). "Bluetooth Special Interest Group." Retrieved 12.07.2017, from <https://www.bluetooth.com/>

Brown, J. (2010). The world café: Shaping our futures through conversations that matter, ReadHowYouWant. com.

Bryman, A. (2012). Social Research Methods, Oxford University Press.

Bude, C. and A. Kervfors Bergstrand (2015). Internet of Things: Exploring and Securing a Future Concept.

Buxton, B. (2010). Sketching user experiences: getting the design right and the right design, Morgan Kaufmann.

Costabile, M. F., et al. (2006). End-user development: The software shaping workshop approach. End user development, Springer: 183-205.

Costabile, M. F., et al. (2008). End users as unwitting software developers. Proceedings of the 4th international workshop on End-user software engineering, ACM.

Da Xu, L., et al. (2014). "Internet of things in industries: A survey." IEEE Transactions on Industrial Informatics **10**(4): 2233-2243.

Dautriche, R., et al. (2013). End-user-development for smart homes: relevance and challenges. Proceedings of the Workshop " EUD for Supporting Sustainability in Maker Communities", 4th International Symposium on End-user Development (IS-EUD).

Dooder (2017, 2017). "freepik - graphic resources for everyone." Retrieved 28.12.2017, from www.freepik.com.

Drei (2016). "Internet of Things leicht gemacht: 3IoT Complete - Drei.at." Retrieved 20.02.2017, from https://www.drei.at/portal/media/pdf/business_1/internet-of-things-iot-complete-english.pdf.

ecobee (2007). "Most residential heating and cooling systems work with ecobee. Check Compatibility. Cool or warm your home from anywhere.". Retrieved 20.02.2017, from <https://www.ecobee.com/>.

Egger, F. N. (2000). Lo-Fi vs. Hi-Fi Prototyping: how real does the real thing have to be? OZCHI.

Evans, D. (2012). "The Internet of Things how the next evolution of the internet is changing everything (april 2011)." White Paper by Cisco Internet Business Solutions Group (IBSG).

Fallman, D. (2003). Design-oriented human-computer interaction. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM.

Fernandes, E., et al. (2016). Security analysis of emerging smart home applications. Security and Privacy (SP), 2016 IEEE Symposium on, IEEE.

FIBARO (2014). "FIBARO provides wireless home automation system. FIBARO's home intelligence products allows one easily convert any home into a true smart home.". Retrieved 10.03.2017, from <https://www.fibaro.com/>.

Fischer, G. (1998). Beyond" couch potatoes": From consumers to designers. Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific, IEEE.

García, C. G., et al. (2014). "Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios." Computer networks **64**: 143-158.

Gaver, W. (2012). What should we expect from research through design? Proceedings of the SIGCHI conference on human factors in computing systems, ACM.

Google. "Project Weave." Retrieved 05.03.2017, from <https://developers.google.com/weave/>.

Google (2006, 02.05.2017). "Trends." from <https://trends.google.com/trends/>.

Google (2016). "Google Home." Retrieved 11.12.2017, from <http://home.google.com/>.

Greenberg, S. and C. Fitchett (2001). Phidgets: easy development of physical interfaces through physical widgets. Proceedings of the 14th annual ACM symposium on User interface software and technology, ACM.

Güldenpfennig, F., et al. (2016). Toward Thingy Oriented Programming: Recording Marcos With Tangibles. Proceedings of the TEI'16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction, ACM.

Güldenpfennig, F., et al. (2016). "Making Space to Engage: An Open-Ended Exploration of Technology Design with Older Adults." International Journal of Mobile Human Computer Interaction (IJMHCI) **8**(2): 1-19.

Gupta, V. S. and P. Raspaile (2015). "Low cost standard Internet of Things." International Journal of Engineering Science & Advanced Technology **5**(2): 78-80.

Haller, S. (2010). "The things in the internet of things." Poster at the (IoT 2010). Tokyo, Japan, November **5**(8): 26-30.

Hanington, B. and B. Martin (2012). Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions, Rockport Publishers.

Hartmann, B., et al. (2005). d. tools: Visually prototyping physical UIs through statecharts. in Extended Abstracts of UIST 2005, Citeseer.

Höök, K., et al. (2015). Knowledge production in interaction design. Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, ACM.

Höök, K. and J. Löwgren (2012). "Strong concepts: Intermediate-level knowledge in interaction design research." ACM Transactions on Computer-Human Interaction (TOCHI) **19**(3): 23.

Horn, M. S., et al. (2009). Comparing the use of tangible and graphical programming languages for informal science education. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM.

Hornecker, E. (2008). Die Rückkehr des Sensorischen: Tangible Interfaces und Tangible Interaction, na.

Hussain, S. R., et al. (2017). Seamblue: Seamless bluetooth low energy connection migration for unmodified iot devices. Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN).

IBM (2014). "Node-RED - Flow-based programming for the Internet of Things." Retrieved 02.03.2017, from <https://nodered.org/>.

Jiuqiang, F. (2015). Teaching method research based on Arduino platform. Intelligent Systems Design and Engineering Applications (ISDEA), 2015 Sixth International Conference on, IEEE.

Joreg, M. W., Sebastian Gregor, Sebastian Oschatz (1998). "vvvv." Retrieved 09.02.2017, from <https://vvvv.org/>.

Kelion, L. (2017). "Garadget faces backlash after locking out irate user." Retrieved 02.05.2017, from <http://www.bbc.com/news/technology-39502256>.

Kim, S., et al. (2016). Acceptance of mobile technology by older adults: a preliminary study. Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services, ACM.

Kinsella, E. A. (2007). "Embodied reflection and the epistemology of reflective practice." Journal of Philosophy of Education **41**(3): 395-409.

Klemmer, S. R., et al. (2004). Papier-Mache: toolkit support for tangible input. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM.

Koomey, J., et al. (2011). "Implications of historical trends in the electrical efficiency of computing." IEEE Annals of the History of Computing **33**(3): 46-54.

Kovatsch, M., et al. (2012). Actinium: A restful runtime container for scriptable internet of things applications. Internet of Things (IOT), 2012 3rd International Conference on the, IEEE.

Kubitza, T. (2016). "Using Speech for End User Programming of Smart Environments in the Internet of Things." Retrieved 10.02.2017, from http://www.dgp.toronto.edu/dsli2016/papers/Kubitza_DSli2016.pdf.

LaCharite, N. (2016). "Alexa Skills Kit Fact Template: Step-by-Step Guide to Build a Fact Skill." Retrieved 21.09.2017, from <https://developer.amazon.com/de/blogs/alexa/post/Tx3DVGG0K0TPUGQ/updated-alexa-skills-kit-fact-template-step-by-step-guide-to-build-a-fact-skill>.

Laurel, B. (2003). Design research: Methods and perspectives, MIT press.

Lazar, J., et al. (2017). Research methods in human-computer interaction, Morgan Kaufmann.

Lechelt, Z., et al. (2016). ConnectUs: A New Toolkit for Teaching about the Internet of Things. Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, ACM.

Lee, I. and K. Lee (2015). "The Internet of Things (IoT): Applications, investments, and challenges for enterprises." Business Horizons **58**(4): 431-440.

Lewis, C. (1982). Using the "thinking-aloud" method in cognitive interface design, IBM TJ Watson Research Center.

Lieberman, H., et al. (2006). End-user development: An emerging paradigm. End user development, Springer: 1-8.

Liebherr (2016). "Liebherr collaborates with Microsoft on smart refrigerator prototype." Retrieved 12.02.2017, from <https://www.liebherr.com/en/rou/latest-news/news-press-releases/detail/liebherr-collaborates-with-microsoft-on-smart-refrigerator-prototype-news.html>.

Linden Tibbets, J. T. (2011). "IF This Then That." Retrieved 11.12.2017, from <https://ifttt.com/>.

Löwgren, J. (1995). Applying design methodology to software development. Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques, ACM.

Löwgren, J. (2007). "Interaction design, research practices and design research on the digital materials." Available at webzone.k3.mah.se/k3jolo. Accessed May.

Löwgren, J. and E. Stolterman (2004). Thoughtful interaction design: A design perspective on information technology, Mit Press.

Mareis, C. (2014). Design als Wissenskultur: Interferenzen zwischen Design-und Wissensdiskursen seit 1960, transcript Verlag.

Martin, F., et al. (2000). "MetaCricket: A designer's kit for making computational devices." IBM Systems Journal **39**(3.4): 795-815.

Mashal, I., et al. (2015). "Choices for interaction with things on Internet and underlying issues." Ad Hoc Networks **28**: 68-90.

Mayer, S., et al. (2014). "User interfaces for smart things--A generative approach with semantic interaction descriptions." ACM Transactions on Computer-Human Interaction (TOCHI) **21**(2): 12.

Microsoft (2015). "Windows 10 IoT Core ". Retrieved 11.02.2017, from <https://developer.microsoft.com/en-us/windows/iot>.

Millner, A. and E. Baafi (2011). Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. Proceedings of the 10th International Conference on Interaction Design and Children, ACM.

Nest (2010). "Nest Learning Thermostat." Retrieved 10.03.2017, from <https://nest.com/>.

NETATMO (2011). "Experience the comfort of a Smart Home: Smart Thermostat, Security Camera with Face Recognition, Weather Station." Retrieved 11.02.2017.

Norman, D. A. and S. W. Draper (1986). "User centered system design." New Perspectives on Human-Computer Interaction, L. Erlbaum Associates Inc., Hillsdale, NJ **3**.

Philips (2012). "Philips Hue." Retrieved 10.03.2017, from meethue.com.

Popper, K. (2005). The logic of scientific discovery, Routledge.

Popper, K. R. (1999). All life is problem solving, Psychology Press.

Poynor, R. (1995). "The Hand That Rocks the Cradle: Gillian Crampton Smith is making the Royal College of Art's Computer Related Design program a multimedia powerhouse." ID-New York-Design Publications- **42**: 60-60.

Raffle, H. S., et al. (2004). Topobo: a constructive assembly system with kinetic memory. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM.

Rittel, H. W. and M. M. Webber (1973). "2.3 planning problems are wicked." Polity 4: 155-169.

Rogers, Y. (2011). "Interaction design gone wild: striving for wild theory." Interactions 18(4): 58-62.

Rogers, Y., et al. (2011). Interaction design: beyond human-computer interaction, John Wiley & Sons.

Rose, K., et al. (2015). "The internet of things: An overview." The Internet Society (ISOC): 1-50.

SAMLabs (2014). "Build and program your awesome inventions with SAM, a collection of wireless blocks and a cool app for exploring your creativity.". Retrieved 09.02.2017, from <https://www.samlabs.com/>.

Samsung (2012). "Smart Things." Retrieved 02.03.2017, from <https://www.smarthings.com/>.

Schön, D. A. (1984). The reflective practitioner: How professionals think in action, Basic books.

Schön, D. A. (1995). "Knowing-in-action: The new scholarship requires a new epistemology." Change: The Magazine of Higher Learning 27(6): 27-34.

Schweikardt, E. (2011). Modular robotics studio. Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction, ACM.

Serbanati, A., et al. (2011). Building blocks of the internet of things: State of the art and beyond. Deploying RFID-Challenges, Solutions, and Open Issues, InTech.

Shneiderman, B. (2003). Leonardo's laptop: human needs and the new computing technologies, Mit Press.

SIG, B. (2017). "Mesh Model Bluetooth® Specification." Retrieved 21.07.2017, from https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=429634

Simpson, A. K., et al. (2017). Securing vulnerable home iot devices with an in-hub security manager. Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on, IEEE.

Starter, T. (1996). "Human-powered wearable computing." IBM Systems Journal 35(3.4): 618-629.

Stolterman, E. and M. Wiberg (2010). "Concept-driven interaction design research." Human-Computer Interaction **25**(2): 95-118.

Tan, L. and N. Wang (2010). Future internet: The internet of things. Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, IEEE.

Tan, S. and J. Brown (2005). "The world café in Singapore: Creating a learning culture through dialogue." The Journal of Applied Behavioral Science **41**(1): 83-90.

Thomas, D. R. (2006). "A general inductive approach for analyzing qualitative evaluation data." American journal of evaluation **27**(2): 237-246.

Tovey, M., et al. (2003). "Sketching, concept development and automotive design." Design studies **24**(2): 135-153.

Tzeremes, V. (2016). End User Software Product Line Support for Smart Spaces, George Mason University.

Ullmer, B. and H. Ishii (2000). "Emerging frameworks for tangible user interfaces." IBM systems journal **39**(3.4): 915-931.

Ur, B., et al. (2014). Practical trigger-action programming in the smart home. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM.

Vaquero, L. M. and L. Roderio-Merino (2014). "Finding your way in the fog: Towards a comprehensive definition of fog computing." ACM SIGCOMM Computer Communication Review **44**(5): 27-32.

Vera. "Vera Smart Home Controller." from <http://getvera.com/controllers/vera3/>.

Vermesan, O. and P. Friess (2013). Internet of things: converging technologies for smart environments and integrated ecosystems, River Publishers.

Villar, N., et al. (2007). "The VoodooIO gaming kit: a real-time adaptable gaming controller." Computers in Entertainment (CIE) **5**(3): 7.

Villar, N., et al. (2012). .NET gadgeteer: a platform for custom devices. International Conference on Pervasive Computing, Springer.

Walport, M. (2014). "Internet of things: making the most of the second digital revolution." Retrieved 10.03.2017, from <https://www.gov.uk/government/publications/internet-of-things-blackett-review>.

Weiser, M. (1991). "The computer for the 21st century." Scientific american **265**(3): 94-104.

Weiser, M. and J. S. Brown (1997). The coming age of calm technology. Beyond calculation, Springer: 75-85.

Wemo, B. (2011). "WEMO That - Home Automation Made Easy." Retrieved 09.02.2017.

Wolf, T. V., et al. (2006). Dispelling design as the black art of CHI. Proceedings of the SIGCHI conference on Human Factors in computing systems, ACM.

Zimmerman, J., et al. (2007). Research through design as a method for interaction design research in HCI. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM.