
In Control of Your Own Devices

Creating and Visualizing Relationships
Between Devices in Home Automation

Master's Thesis in Computer Science

Martin Eskerud

May 15, 2015
Halden, Norway



Abstract

This thesis is a continuation of the home automation graph interface concept developed in the Controlling the Home project [47].

Home Automation is the discipline of enhancing devices in the home through technology. One of the directions for home automation is to create automatic behavior between devices. Existing applications facilitate connecting devices and creating simple logic in the home, such as homeBLOX [42], openHAB ¹ and homeMaestro [30].

After presenting these prototypes to users in a future workshop, a prototype is developed through two iterations of testing, including an expert evaluation. During the development process, a non-text-based programming language is integrated into the interface. The prototype facilitates remote control functionality, and design choices for visualizing a device's state and its utilities are explored. Furthermore, sensors visualization is integrated into the interface.

Finally, the prototype is evaluated by eight users in the target audience, which is young adults with little to no prior programming experience. The users are able to complete simple end user programming tasks by using the prototype.

The work in this thesis provides a new direction for the graph interface developed in Controlling the Home. The introduction of EUP to the interface is well received and tested to be understandable by users who possesses no prior programming experience.

Keywords Home Automation, Relationships, Sensors, Graph Interface

¹<http://www.openhab.org/>

Acknowledgments

A special thank you to Harald Holone for being my mentor through the duration of this project. I enjoyed every second of it.

Thanks to Anders and Sofie for keeping me company through the year, it would not have been possible without you. Special thanks to Anders for getting me started with Photoshop and lending me your tablet for testing and to Per for playing pool with me in the autumn. Another thanks to my family for being patient with my overall absence this year.

A last thanks to all the testers involved in the thesis, you made this possible.

Contents

Abstract	ii
Acknowledgments	iii
List of Figures (optional)	xi
List of Tables (optional)	xiii
1 Introduction	1
1.1 History of Home Automation	1
1.2 Home Automation Software	2
1.3 Research Question	5
1.4 Background and Motivation	7
1.5 Visualizing a Home Through a Graph Interface	7
2 Related Work	10
2.1 Relationships Between Devices	10
2.2 Interface Concepts and Theory	15
2.3 End User Programming	17
3 Method	21
3.1 Idea Generation	21
3.2 Evaluations	22
3.3 The Prototype	27
3.4 Summary	28
4 Prototype	29
4.1 Future Workshop	29
4.2 First Iteration Development	42
4.3 First User Test	51
4.4 Results from the User Test	54
4.5 Considerations and Other Observations	67
4.6 Second Iteration	69
5 Results	77
5.1 Preface and Analysis	77
5.2 Results from User Evaluation and Testing	81

6 Discussion	97
6.1 Facilitating Relationship Functionality in a Graph Interface	98
6.2 End User Programming	104
6.3 Sensors and Utilities	113
6.4 States of Devices	117
6.5 Summary	119
7 Conclusion	121
7.1 Concluding the Research Questions	121
7.2 Future Work	123
7.3 Reflection	125
Bibliography	132
Index (optional)	133
A Form of Consent - Future Workshop	134
B First User Test	136
C Final User Test	139
C.1 Tags Produced from Open Coding	148

List of Figures

- 1.1 The world’s first automatic dishwasher. The inventor, Josephine Cochrane, is depicted in the top left corner. 2
- 1.2 The micasaverde interface. Many buttons and menus. 3
- 1.3 Each circle is a ”room node”. Here, kitchen and hallway are selected. These rooms have sub-nodes visualizing the kitchen’s and hallway’s devices. 7
- 1.4 Controlling the Home visualizing a device with more than one utility. In this case, a stove with multiple hotplates. 9

- 2.1 Ninja Sphere interconnecting phone, TV and location. 11
- 2.2 Screen capture from openHAB tutorial on creating a scene of lights. 11
- 2.3 The Jigsaw Editor, in use and explained view. 12
- 2.4 The homeBLOX wizard. A two step creation of a relationship between devices. . . 12
- 2.5 Screen capture from the HomeMaestro promotion video. 13
- 2.6 Two connections visualized in Controlling the Home. The light is both connected to the TV, and the Fireplace. 13
- 2.7 Left: The beacons, hubs planted in your house. Right: The Airfy Beacon App, here shown selecting connections between devices. 14
- 2.8 From the Athom Kickstarter page, explaining how users can create automatic behaviour by using the Athom system. 14
- 2.9 The navigation pattern of a three-layer radial menu. 16
- 2.10 Screenshot from the Yahoo! Pipes application showing two ”sources” connected by pipes. The output is shown in the bottom of the screen. 18
- 2.11 A part of a ”googly eyes” program made in scratch, where the cat’s eyes follow your mouse cursor. Window explanation, left to right: Output, palette, block editor. 19
- 2.12 The red drawings shows the steps to make a the balloon animate upwards. Image from the StageCast introduction. The ”code” or rule maker as it is called in Stagecast is hidden inside the yellow outline around the balloon. . 20

- 3.1 The System Usability Scale grading curve. 68 (C) is considered average. . . 27
- 3.2 A screen from the high-fidelity prototype. The user has used two if-blocks to create logic on the light device. 28

4.1	Inconsistencies in the Controlling the Home prototype. The red circles are devices which should behave similarly, but the pair in the kitchen node has double blue lines, while the hallway nodes have one blue line and a thick grey outline. The grey outline is intended to be 'off', but this was not clear to users. The green pair has only a blue line as they have child nodes of their own similar to rooms, but the users did not understand why they had no border.	32
4.2	A new suggestion for the room node, the border would contain an image of the room, while the middle could be used for an icon or other functionality.	33
4.3	A discussion around an idea between the participants.	33
4.4	The pictures presented to the users of the Controlling the Home prototype. Here, there are relationships between devices in the light room, which are invisible when the room is closed.	35
4.5	This was thought to be the "advanced view". To the left, the user can actually program directly into the interface. Two thirds of the screen is used to allow users to stay connected to the old view, not losing their sense of position in the interface.	40
4.6	The "simple" editor. Here users can use pre-defined structure blocks to create logic between devices.	41
4.7	A side-scroll bar showing scenes and active devices. This was thought to be a small addition to the prototype, which could increase the overview of the application.	41
4.8	Left: reverse fish eye effect. Right: No distortion to the image.	43
4.9	The "simple" editor drafted during the future workshop. Here users can use pre-defined structure blocks to create logic between devices.	44
4.10	An image of the relationship editor during development.	45
4.11	The left figure shows the various parts the user can use in order to create a relationship. An example of a finished relationship is shown in the right figure.	47
4.12	An off-canvas layout used in Facebooks Messenger Application. Red: What used to be on top is now slid to the side, making way for the contact list on the right.	48
4.13	A comparison between the implementation, and the inspiration for the relationship overview menu.	48
4.14	Inconsistencies in the Controlling the Home prototype. The red circles are devices which should behave similarly, but the pair in the kitchen node has double blue lines, while the hallway nodes have one blue line and a thick grey outline. The grey outline is intended to be 'off', but this was not clear to users. The green pair has only a blue line as they have child nodes of their own similar to rooms, but the users did not understand why they had no border.	49
4.15	Bottom left: The "knob" for dimming. Top right: A device or room node with children illustrated as small circles.	50
4.16	"If the fireplace is 6pm?" This statement is confusing, and not intuitive.	51
4.17	The setup for the user test. I would sit next to the tester, in order to aid during training tasks and to observe their interactions with the interface.	52

4.18	Green: The device nodes for lights in two rooms. Dragging one on top of the other folds out the Configure Connection menu. The relationship between them is visualized by a relationship arrow. Red: The device nodes which are possible to be used when defining the relationship. Orange: The device nodes used in the relationship. There is no feedback to the user explaining which icon corresponds to which device as they are identical. Blue: The Toolbox	57
4.19	My switcher in comparison to the default android switcher.	57
4.20	User # 2 drew what she thought would be a more logical way of defining relationships. This would later be implemented, the changes and comparison to the old prototype is shown in Figure 4.24. This solution also uses a clock as a device node.	61
4.21	Child node indicators on both room- and device nodes.	62
4.22	A visual representation of the Power Adjustment Gesture. Having briefly pressed light node in the middle, without releasing their finger, run it outwards away from the node pressed. The power of the device will be adjusted based on the finger's angle to the light node.	63
4.23	Wheel menu concept	64
4.24	Left: The old relationship block used a device node to serve both as "action" and as a representation of itself in the programming logic. Right: The red arrows show how each purpose of the device node has been redistributed in order to make it easier to use, and more intuitive. The new relationship block (right) is not fit to scale.	70
4.25	Left: Stove during the first iteration, mistakenly a binary device node. Middle: The first iteration's floating value node. Right: The new device node powered on, now also changed to a floating value device node.	72
4.26	An excerpt of the application view. The red circles show where the user can press in order to "select" the adjacent node or device block. Presses not within the red zones are ignored.	73
4.27	An example of what happens if the user presses somewhere in the invision-App prototype tool if there is no picture mapped to that area.	75
4.28	The new touch zones for the prototype.	75
5.1	The age demographic data from The Gathering 2015.	78
5.2	The setup for the final evaluation. During the test, a camera was filming over the shoulder of the evaluator to capture the test tablet's screen.	80
5.3	The transition screens between low and high-fidelity prototype. The transition is to open the relationship editor, and to open a different room node. Top: relationship editor closed. Bottom: first screen of the high-fidelity prototype, where the relationship editor is open.	83
5.4	The complete solution to training task 3. None of the users completed this, instead they stopped after one "on-on"-if block, and one "off-off" if block.	88
5.5	The operator in the 'then clause' is highlighted in red. Melinda identified this as redundant, as all tasks would use the equals operator, and suggested that it should be static instead. A proof of concept for this is found in Figure 6.7 as "third iteration".	88
5.6	The movement sensor icon.	92

5.7	Radial interface example	92
5.8	Questionnaire results.	96
6.1	The two concepts emerging from the future workshop.	99
6.2	An off-canvas layout used in Facebooks Messenger Application. Red: What used to be on top is now slid to the side, making way for the contact list on the right.	100
6.3	An "cockpit style interface". Not scalable, lots of buttons and menus. . . .	101
6.4	Menu showing scenes and active devices. This was thought to be a small addition to the prototype, which could increase the overview of the application. This mockup would become the overview menu shown in Figure 6.5.	101
6.5	The overview menu, an off-canvas menu. Here implemented in the first iteration low-fidelity prototype	102
6.6	The home screen of the application here shows a blueprint of a flat as a background. This could help the interface feel less abstract.	103
6.7	The evolution of the if block.	105
6.8	The toolbox housing logic blocks. Picture from the second iteration prototype.	107
6.9	Left: A common timer structure from the second iteration prototype. A timer relationship always requires a clock -, equals -, and a timer block. Right: Mock up of new timer block. Here the clock and operator is fixed, the user only selects the time. If not for spacing, it could also be more fixed and read "when the time is...".	108
6.10	The different blocks in the Scratch software. Figure from Maloney et al. [37].	109
6.11	The logic block selection menu in Scratch.	109
6.12	A pseudo code representation of the work flow in the prototype. Left: How it is currently. Right: A simpler version which the prototype does not facilitate.	111
6.13	The complete solution to training task 3. None of the users completed this, instead they stopped after one "on-on"-if block, and one "off-off" if block. This also highlights the problem of ambiguous device node icons.	112
6.14	A suggestion for the new sensor visualization following Jemma and Calvin's opinions.	114
6.15	Device states and sensors. Left to right: movement sensor off, movement sensor on, other device on.	114
6.16	The evolution of the device state visualization.	114
6.17	Child node indicators on both room- and device nodes.	115
6.18	Radial menus from several games. Top left to right: Dishonored, Tal-lowmere, League of Legends, Zelda and Dota2.	116
6.19	Inconsistencies in the Controlling the Home prototype. The red circles are devices which should behave similarly, but the pair in the kitchen node has double blue lines, while the hallway nodes have one blue line and a thick grey outline. The grey outline is intended to be 'off', but this was not clear to users. The green pair has only a blue line as they have child nodes of their own similar to rooms, but the users did not understand why they had no border.	117

6.20 The evolution of the floating value device node. Left to right: Controlling the Home, first iteration design after future workshop where the node is 'off' at 50%, then 'off' and 'on' at 50% of the second iteration design tested in the final evaluation. 118

List of Tables

- 4.1 An overview of the evaluators in the first test. 29
- 4.2 Overview of testers involved in the first user test. 52
- 4.3 The data from the expert evaluation’s questionnaire. 67

- 5.1 Overview of testers involved in the first user test. The table is ordered based on the order of evaluators testing. 79
- 5.2 Tags selected through selective coding. 81
- 5.3 The full scores from the SUS questionnaire used during final evaluation for all users. 96

Chapter 1

Introduction

”You are disoriented. Blackness swims toward you like a school of eels who have just seen something that eels like a lot.”

Douglas Adams

The field of home automation is going through exciting development. In recent years, the technological revolution has skyrocketed the rate of new inventions within home automation. Leading up to the twenty first century, there has been a steady increase in development within the home automation field leading up the technological gold age which we are currently experiencing.

1.1 History of Home Automation

To see why home automation is in its current state, we can look back and see the impacts and changes which have influenced the home automation domain.

The 1800s were the starting age of modern home automation. In 1850, Joel Houghton submitted a patent for ”Improvement in machines for washing table furniture” [22]. However, this device was hand powered, not automatic. In 1886, Josephine Cochrane invented the first automatic dishwasher after growing tired of seeing her servants dropping and breaking her china. She swore there had to be a better way of doing dishes. Her invention was later unveiled at the Chicago World Fair in 1893, alongside many other home automation inventions [10, p. 1910]. The dishwasher as it stood in the World Fair is depicted in Figure 1.1 below.

In the 1900s, electricity became more abundant. This allowed more automatic home devices to emerge. Virtually every home task which was previously done by hand now had some kind of home appliance which would aid users. In the 1950s, home automation in America began becoming the standard. Appliances such as dishwashers and electric stoves were no longer reserved to restaurants, hotels and the elite. In this decade, several now essential devices were introduced to the commercial market. Consumers could now buy stoves, and washing machines. Most of the devices which emerged at this time was simply electrical or gas powered ways of doing previously manual tasks. The microwave however, was something completely new. The microwave was invented on accident by Percy Spencer, who was developing radar equipment for the US Navy. Several colleagues



Figure 1.1: The world's first automatic dishwasher. The inventor, Josephine Cochrane, is depicted in the top left corner.

had noticed food melting when working nearby the radar equipment, but Spencer was the first to investigate why [9]. In 1945, he filed the patent for "Method of treating footstuffs" [48].

The next big change in home automation would be the introduction of the microprocessor.

In the early 1970s, the microprocessor immediately began to change and shape the world, as well as the home and kitchen. During the next decades, technological advancements would introduce the home to timed devices such as microwaves and ovens, as well as coffee makers with drip control and programmable portion sizes. GE claims to have started the trend of modern home automation ¹. At the same time, more companies were experimenting with combining microprocessors and home appliances.

Today, in the 21st. century, the groundwork of technologies has been laid out. There is a broad platform of sensors, wireless technologies, mobile devices and smart devices which can all be combined. Currently we are seeing that these technologies are improving. Everything is becoming smaller, cheaper and more accurate. Users have begun experimenting and tinkering with the technology being brought forth by corporations. We can observe this development directly in the growth of start up companies on sites such as Kickstarter ². The current products which I looked towards will be detailed in Section 2.1.

1.2 Home Automation Software

In today's world, consumer grade hardware as well as software tools for home automation are available which allow the user to do virtually anything. Kickstarter and similar online communities where people interested in technology visit are filled with new improvements to existing home automation, as well as new products. Today, even our cutlery can connect other devices [29].

With the increasing presence of the Internet of Things (IoT) as summarized by Xia et al.[55], there is a lot of power in the consumer's hands to create their own solution, or to customize existing ones to fit their needs.

¹http://www.geappliances.com/build-with-ge/downloads/kitchens_of_tomorrow_today.pdf

²www.kickstarter.com

We can observe from the prototypes of smart homes, home automation and new devices connecting to the IoT, that the devices are in abundance, while the systems controlling them are not. Miorandi et al. recognizes "device heterogeneity" as one of the key features an IoT system needs to consider. This is observable in the market today, and the software, both front- and back-end are considerably underdeveloped compared to the hardware available [39]. This supports the findings of Bandyopadhyay and Sen who concluded that the fragmentation of standards and protocols in hardware for home automation could hamper and become a problem in IoT becoming popular [3]. Therefore, we need to find better ways of controlling and presenting information from the hardware.

In 2014, I was part of a project looking at new solutions for home automation. In this research, we found that users expressed dismay with some of the current solutions to home automation, where the interfaces are "cockpit-like" [47]. Due to the prevalence of cockpit interfaces, we saw potential in pursuing something different. An example of this cockpit-like interface is shown in Figure 1.2, where Micasaverde's ³ interface is illustrated.



Figure 1.2: The micasaverde interface. Many buttons and menus.

While commercial solutions such as the Micasaverde are available to consumers, I have previously mentioned that users too can create entirely new solutions all by themselves.

One such user was Chris Cicchitelli, who in 2008 created CastleOS ⁴. CastleOS is a complete home automation system. The system makes use of several different sensor types, which allow the user to create complex rules and connections between devices. The user interface is hidden, and is usable through voice only. Movement sensors disable lights in rooms which are not used, and resemble the "ambient intelligence" used by Garate et al [19]. Their interfaces utilizes holograms of avatars in order to humanize the natural language interface. By using a voice controlled interface, the user is not overloaded by information, but can access additional information when necessary from a manual, the users hands and eyes are also free while interacting with an auditory interface. The downside of a voice controlled interface is that it can be inaccurate and slow [46, p. 331-332].

Both voice and gesture operated interfaces are currently popular solutions, but also

³<http://getvera.com/>

⁴www.castleos.com

more simplified interfaces relying on direct manipulation of objects and items exist. In a previous project "Controlling the Home", we found that users responded positively to an abstract direct manipulation interface for representing the home [47]. This is further detailed in Section 1.4. A feature of this interface was to allow the user to connect different devices, creating a relationship between them.

These relationships provide the user with the ability to customize their home.

Connecting devices in home automation is not a new concept. In 2008 the Microsoft sponsored research team HomeMaestro presented a simple way of connecting two or more devices. Their system would listen to the devices in the home. Creating a relationship was easy, switch one item off, then another on or off while recording the relationship. This would allow the user to create chains of on/off behaviour between devices [30]. This thesis builds upon Controlling the Home, which conceptualized the ability to create the same functionality as HomeMaestro. This was done exclusively through the mobile touch interface, without the need for tangible interaction with devices [47]. No back-end solution to the devices was developed, and will also not be considered in this thesis.

When introducing home automation to a home, there is a necessity for the user to still recognize their home. Making technology "homey" is important, as alienating the home to the user is undesirable [52]. There are however different schools of thought concerning how the home needs to be slightly alienated from the user in order to allow new technologies to emerge. This should be done carefully while allowing the user to experience the benefits of new technology, and be based in testing and research deeming it necessary [52][5].

There is a general consensus within the home automation domain that the systems are generally more approached by tech savvy individuals [7, 38, 52, 42]. These are the type of users who generally prefer command language interaction [46]. There are however exceptions to this rule, and it is not a bold claim to say that even tech savvy users use both direct and menu manipulation. One of the 2000s biggest names within home automation is Bill Gates. Mr. Gates keeps most of the details about his extensively automated home secret, but there are a few things known about his home automation system:

- Guests are given pins with RFID chips, allowing the temperature, humidity, lighting and music to change to their preference, in their current location ⁵.
- Lights are dimmed and adjusted based on your location in the house.
- A media center allows users to select art, music and entertainment on any screen in the home⁶.

These attributes are facts about the system which have been observed by guests, and do not detail the possible closed software which could be running behind.

Looking towards systems such as Mr. Gates' is useful, as it can clearly show which way he envisions home automation to go. Sadly, there are a lot of missing details, but suggests a direction - more sensors and to allow interaction when necessary.

With this history and current situation established, it is time to describe the research topics for this thesis.

⁵<http://www.connectedtvnews.com/first-connected-home-bill-gates-residence/>

⁶<http://seattle.curbed.com/archives/2013/01/bill-gates-medina-mansion.php>

1.3 Research Question

The research question of this Master Thesis is:

How can relationships between household devices be visualized by using a graph interface for the purpose of home automation? Specifically,

1. How can a graph based interface for home automation integrate relationships between devices?
2. How can a design for a non-text-based programming language for a home automation interface on a mobile device be designed?
3. How can device utilities and sensors be visualized in a graph based home automation interface?
4. How can the state(s) of devices be visualized in a graph based home automation interface?

While sensors, device states and utilities do not directly impact relationships, their presence in the interface impacts relationship visualization. Sensors should be usable as a piece in a relationship due to their common presence in real world home automation systems [7]. The interface should also facilitate altering device states and utilities through relationships, and when altered, their new state should be clear to the user.

The next section will describe the background for the project. The motivation behind using a graph based interface as seen in the "Controlling the Home" project is also detailed. Relationships, and how to create them will be explored in Section 2.1.

1.4 Background and Motivation

In 2014, I was part of a project to create a graph interface for home automation through participatory design [47]. During this project, we created a prototype which was later used in testing the interface concept. This prototype was called "Controlling the Home". The prototype used a graph interface to visualize a home and the devices inside. This graph style interface will serve as a foundation for the prototype which will be developed for this project in order to answer the research question.

1.5 Visualizing a Home Through a Graph Interface

The interface of Controlling the Home featured two main concepts. In Figure 1.3 below, we can see how the application looked when two room nodes are open. A room node is an abstraction of a room, illustrated as a node. This node in turn has its own child nodes which are the devices in the room.

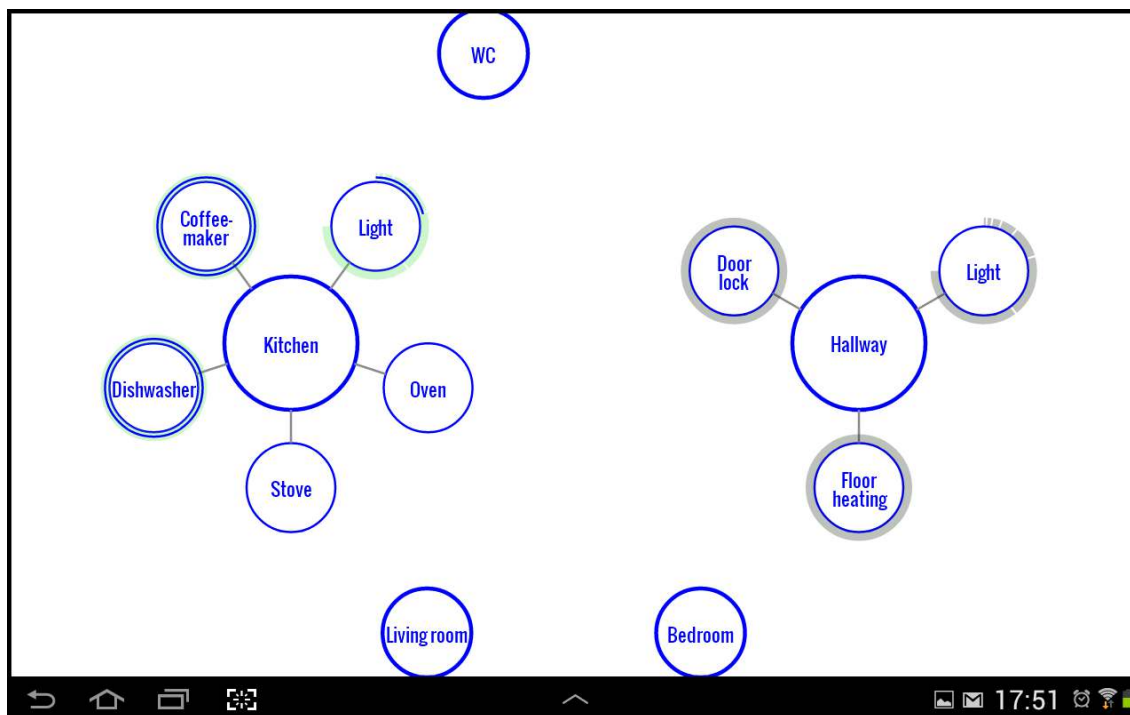


Figure 1.3: Each circle is a "room node". Here, kitchen and hallway are selected. These rooms have sub-nodes visualizing the kitchen's and hallway's devices.

These nodes in this interface means the interaction style of this prototype was "direct manipulation". The nodes could be selected and unselected by pulling them towards the middle- or the edge of the screen, respectively.

1.5.1 The graph concept

This interface featured two main concepts. The user was able to navigate a home represented as nodes, and they were able to create "rules" in order to control the devices in the home.

1.5.2 Rule and relationship functionality

From testing, we concluded that the prototype was able to illustrate a home as a graph, but during testing we also gathered feedback regarding the rule and connection functionality. As the project's goal was to establish the effectiveness of visualizing a home by using a graph, we did focus on testing or improving the rule functionality throughout the project.

This functionality looked promising, but was underdeveloped, it needed refinement in order to be part of a complete home automation system. Controlling the Home listed several concepts and ideas as potential new areas for future work. Each research question in Section 1.5.3 below is detailed in how it ties into future work described in Controlling the Home.

1.5.3 How this thesis ties into Controlling the Home

Each part of the research questions presented in Section 1.3 has ties into the previous work and experiences from Controlling the Home. The following list presents each research question, followed by their relation to Controlling the Home:

- How can a graph based interface for home automation integrate relationships between devices?

In the Controlling the Home prototype, connections between devices were illustrated by drawing an arrow between the two devices which were connected. However, if the user would create many relationships, the system would become difficult to maneuver and the interface would not provide the same overview of the home. In addition, only connections between the rooms currently being shown would appear. Overall, it did not provide the user with a sense of control over the active relationships between devices in the home.

- How can a design for a non-text-based programming language for a home automation interface on a mobile device be designed?

Relationships were implemented into Controlling the Home in order for there to be more to do than simply remote controlling devices. In this thesis, I want to develop this functionality into a non-text-based programming language integrated into the graph interface.

- How can device utilities and sensors be visualized in a graph based home automation interface?

In Controlling the Home, the goal was to see how a graph based interface worked for visualizing a home in the context of home automation. In order to focus on this, other functionality was omitted. One of the things which Controlling the Home did not visualize was sensors. Sensors are cheap, useful and expected to be in a modern automated home [12]. Sensors are commonly used as "triggers", where a sensor reading can for instance switch the lights off, if there are no one in the room.

Device utilities refer to devices with more than one action. Such as a stove top as shown in Figure 1.4, or a washer.

The solution developed in Controlling the Home was very consistent with the graph design. Several devices would benefit from having an improved visualization of their

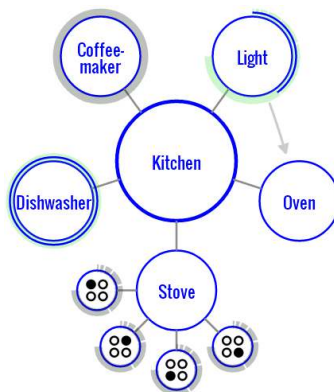


Figure 1.4: Controlling the Home visualizing a device with more than one utility. In this case, a stove with multiple hotplates.

utilities. I want to improve this visualization, which will also impact a device's states in regards to relationships between devices.

- How can the state(s) of devices be visualized in a graph based home automation interface?

In this thesis, I will divide device states into two categories; binary device nodes, and floating value device nodes. A binary device node has two states, 'on' or 'off'. Floating value device nodes have a range of power, such as a light or a speaker. These visualizations were present in *Controlling the Home*, but need to be consistent with the new designs proposed in this thesis in order to interlace optimally with relationships.

When addressing these issues, I will attempt to integrate new functionality, rather than just implement it. I will be open to rethinking several ideas from *Controlling the Home*, but I wish for the core graph interface to remain as it was the main product from *Controlling the Home*.

1.5.4 Terminology

To describe interaction between two devices, I will refer to the device causing an action in another device as the trigger device. The device which the trigger interacts with will be referred to as the receiving device. The connection created within the interface between devices will be referred to as a relationship.

Chapter 2

Related Work

”I’m not a psychopath, Anderson. I’m a high-functioning sociopath. Do your research.”

Sherlock Holmes

In the first part of this section, I will look at existing home automation software facilitating connecting devices. The ”Controlling the Home” prototype will also be considered here, but only for its functionality in connecting devices together.

In the second part of this section, overall concepts for interface design which can be influence potential changes to interface will be detailed.

2.1 Relationships Between Devices

Devices in a home can be connected in more ways than one. The location of a device can make it connected to something such as a radio in the shower, it can be connected physically to another device such as a TV and a DVD player or it can be connected digitally through software. For this thesis, I will refer to any devices with either of the above connections as a relationship.

Some devices can change its utility and purpose when it is connected to another device. A video game console is useless without a screen or projector, while a hair dryer has less synergy with other devices. Relationships between devices can sometimes be hidden as well, if we consider a stove and a TV, there appears to be no direct relationship between them. However, if we use the TV to display a warning or a small image of the items on the stove we can create new utility by connecting devices through software.

The Kickstarter project ”Ninja Sphere”¹ aims to interconnect devices and create serve as a complete solution for home automation. This concept is visualized in Figure 2.1 below.

Ninja Sphere allows for simple ”if this then that” rules based on presence, location, time and energy usage. Rietzler et al. described this kind of rule behaviour to create automation scenarios which would likely create a set of possibly interfering but also hard to maintain rules[42]. This feedback supports our previous findings, as their users experienced problems when several rules were engaged simultaneously with little overview [47].

¹www.ninjablocks.com

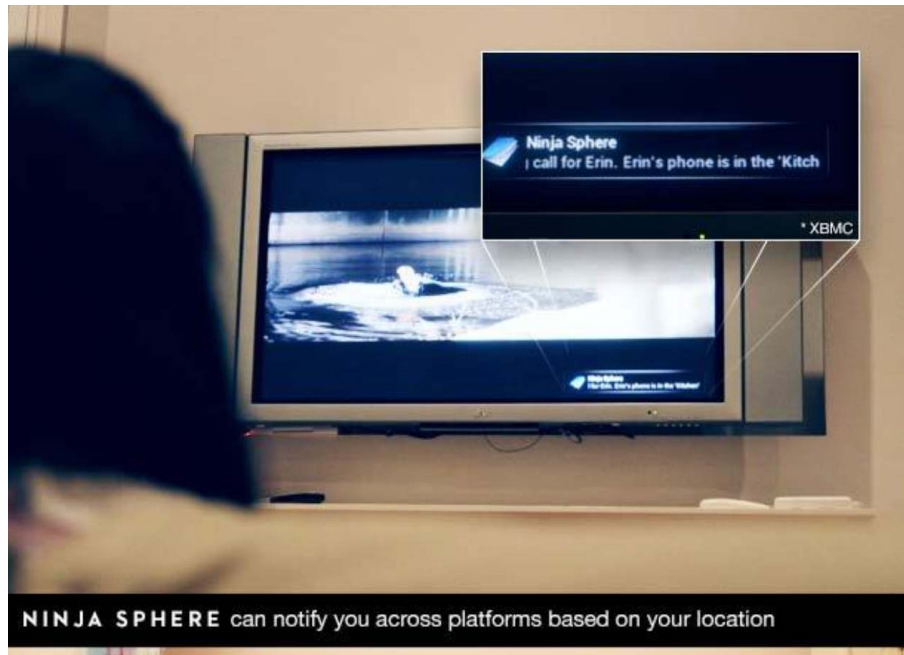


Figure 2.1: Ninja Sphere interconnecting phone, TV and location.

openHAB ² is an open source home automation project utilizing Raspberry Pi ³. In one of their promotional videos ⁴, openHAB shows the system's ability to "bind" devices together, and allow them to be controlled together remotely. OpenHAB refers to this as a scene, which is depicted below in Figure 2.2.

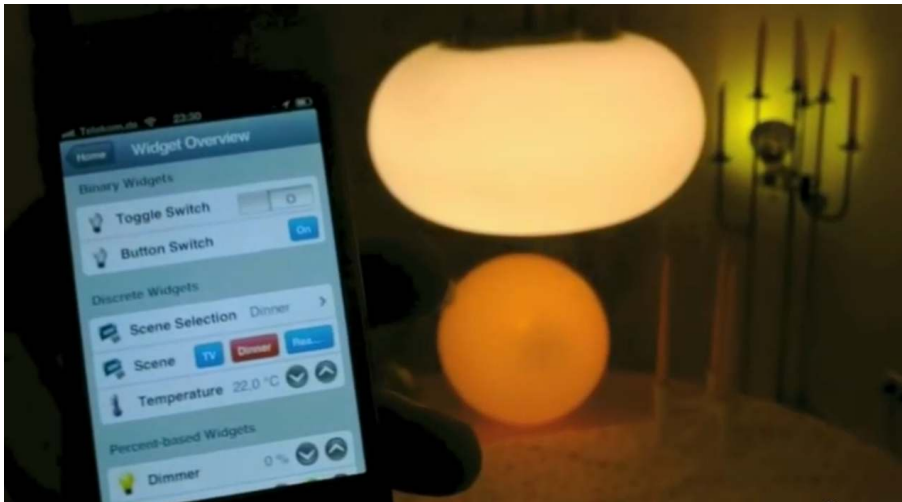


Figure 2.2: Screen capture from openHAB tutorial on creating a scene of lights.

In the video, we can see how openHAB allows the user to customize and bind a group of lamps into a scene. In Figure 2.2 the scene now selected is "dinner", other available

²www.openhab.org

³<http://www.raspberrypi.org/>

⁴<https://www.youtube.com/watch?v=Q4.LkXIRBwc>

scenes are TV and reading. openHAB provides a framework for the users, but everything is fully customizable and open if the user is interested and tech savvy enough to customize it as they wish.

In 2013, homeBLOX created their prototype highlighting the use for block or module based structure in home automation. Home automation solutions need to be flexible in order to accommodate the wide range of possible device combinations in the homes of the world [42]. Their tangible interface solution was influenced by the older prototype "Jigsaw Editor" from 2003, shown in Figure 2.3.



Figure 2.3: The Jigsaw Editor, in use and explained view.

This tangible user interface was only a prototype, but shows that the concept of a direct manipulation user interfaces for home automation has been around for atleast a decade. The Jigsaw Editor used colors and animations in order to captivate the user, elements from this tangible interface was carried on into the homeBLOX interface, as shown in Figure 2.4.



Figure 2.4: The homeBLOX wizard. A two step creation of a relationship between devices.

In the homeBLOX interface, the same three-part creation of relationships is present as we observe in the Jigsaw Editor. First, we choose the trigger device, then we select the target device and then define what happens between them. This is the same technique used by HomeMaestro [30], where the user selected the action on the device by physically interacting and recording the desired behaviour.



Figure 2.5: Screen capture from the HomeMaestro promotion video.

This solution utilizes tangible interaction with objects, rather than menu selection or direct manipulation in order for the user to create relationships [46] [26]. Figure 2.5 does not show the interaction with tangible objects, but the options are derived from real world interactions with devices.

In our prototype "Controlling the Home" [47], the different actions were not considered, as only the action "on" and "off" were supported. The visualization of the connection was made by drawing an arrow between devices visualized as circular nodes.

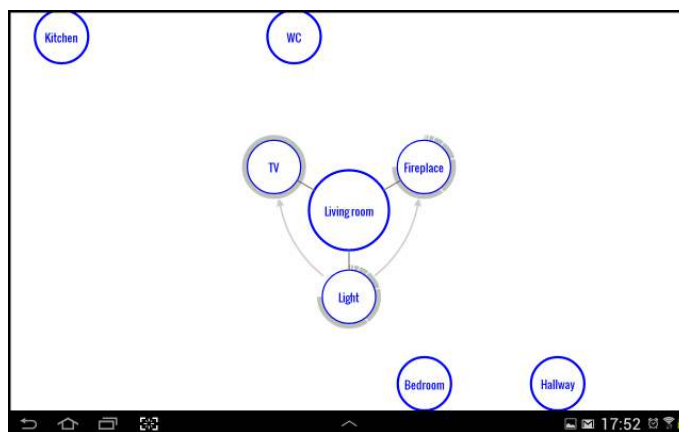


Figure 2.6: Two connections visualized in Controlling the Home. The light is both connected to the TV, and the Fireplace.

Figure 2.6 shows the prototype home automation system Controlling the Home. Here, a single room node is open, and the light in the room is connected to two other devices simultaneously.

All of the aforementioned solutions involve some degree of end user programming (EUP), which will be detailed further in Section 2.3. Another open product is the Airfy's Beacon. At its simplest, the Beacon, which uses several WiFi sensors throughout the home, can switch off devices in rooms where nobody is present. These beacons as well as the iPhone app is shown in Figure 2.7. As with most new projects, Airfy's Beacon is open source and can be configured and therefore accommodates and encourages end user programming.



Figure 2.7: Left: The beacons, hubs planted in your house. Right: The Airfy Beacon App, here shown selecting connections between devices.

Another kickstarter project currently being developed is the Athom⁵. The athom visually bears resemblance to the Ninja Sphere mentioned above. The Athom also uses "if this then that" style interaction for creating rules. This is explained in Figure 2.8.

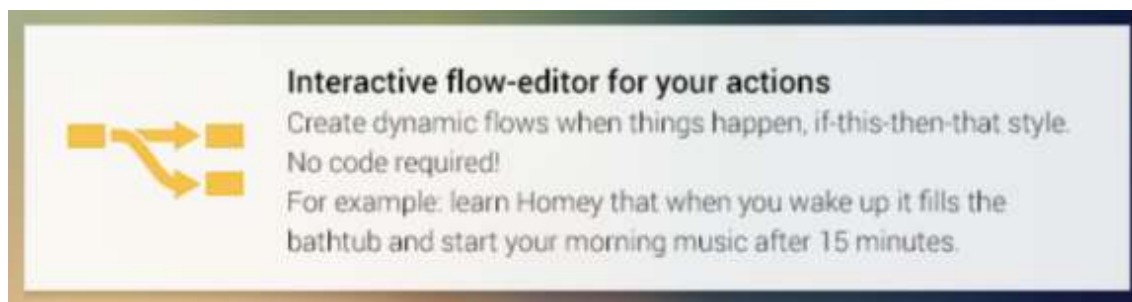


Figure 2.8: From the Athom Kickstarter page, explaining how users can create automatic behaviour by using the Athom system.

As the Athom is still being developing in time of writing, there is missing information about the product. As shown in Figure 2.8, Athom will allow for if-this-then-that logic. The Athom refers to their relationship editor as a "flow-editor".

⁵<https://www.kickstarter.com/projects/athom/homey-the-living-room-talk-to-your-home?ref=discovery>

2.1.1 Relationships Between Devices in Summary

Looking at the different solutions for visualizing relationships between devices, there are common factors in how this is currently practiced.

All of the home automation solutions presented an interface where the user needed to choose both the first and the second device to be connected together. The different software solutions all had different methods of choosing these devices however.

Having selected the target devices, the user then needs to select an action on the initial device, which will trigger an action in the second device. We can also observe that there none of the software presented in Section 2.1 had a common name for two devices being connected. Therefore, for the remainder of this thesis I will again refer to two devices having a connection as a "relationship" between devices, the words connection, rule, scene and binding are also applicable, but I feel these terms are narrow and not as descriptive.

Some of the home automation systems described in Section 2.1 explicitly state the concept behind their relationship functionality as "if this then that", showing that this is a common way of structuring EUP applications for home automation. The next section will detail other concepts which will be used in creating a concept for visualizing relationships between devices.

2.2 Interface Concepts and Theory

When looking back at the previously mentioned applications in Section 2.1 we can see that there are parallels to be drawn, for instance their common factor of all incorporating end user programming.

As part of answering the research questions, a graph interface will be developed. This interface will build on the design from Controlling the Home project [47], but will introduce new elements in order to visualize relationships.

2.2.1 Graph Interfaces

Interfaces using graphs are abundant in today's world. Users are interacting daily using graph interfaces, perhaps without realizing it. Underground maps for trains [49, pp. 69-89], and folder structure in operating systems both use graphs and networks to illustrate information to the user with great success. Lately, web applications allow users to create "friend maps" of social networks. These networks are classic representations of networks, and are understandable and shared by users with little to no technical background.

Freire and Rodríguez's paper "Preserving the mental map in interactive graph interfaces" describes the importance of avoiding information overload in graph interfaces. They also emphasize the importance of animations and feedback to the user when navigating levels in the graph, in order to ensure the user experiences the interface as predictable, not changing the view drastically between interactions and being able to tell what changes are happening to the interface [16].

2.2.2 Tutorials in Mobile Interfaces

Tutorials in mobile interfaces is sometimes necessary. Making good tutorials and help interfaces is a challenge however, in addition, users are prone to not seek help in digital

interfaces. Novick and Ward [40] found that the most common causes for users being deterred from seeking help in the application were:

”Users cite difficulties in navigating the help systems, particularly difficulties in finding useful search terms, and disappointment in the level of explanation found.”

Therefore, the help implementation needs to be well thought out to provide the help necessary. Another concern is adding too much help, and making it intrusive. Inbar et.al looked at different ways of implementing help interfaces on mobile, and found that the least intrusive help implementation for mobile devices was to use an icon, which the user can interact to get help. They explore ”pushed” help, as in tips which are not requested by the user, but instead prompted to the user [25]. The alternative to this is ”pulled” help, which users request themselves.

When to use pulled or pushed help is explored by Jepsen et al. who through user testing determine that the only factor to consider when deciding on how to display help is the ”perceived criticality” of the action which the user needs help with [27].

2.2.3 Radial Interfaces

Radial interfaces are selection menus where the user starts off in the middle, and navigates towards the outer edges to select an option or item. An example of a radial menu interaction in four steps is shown in Figure 2.9.

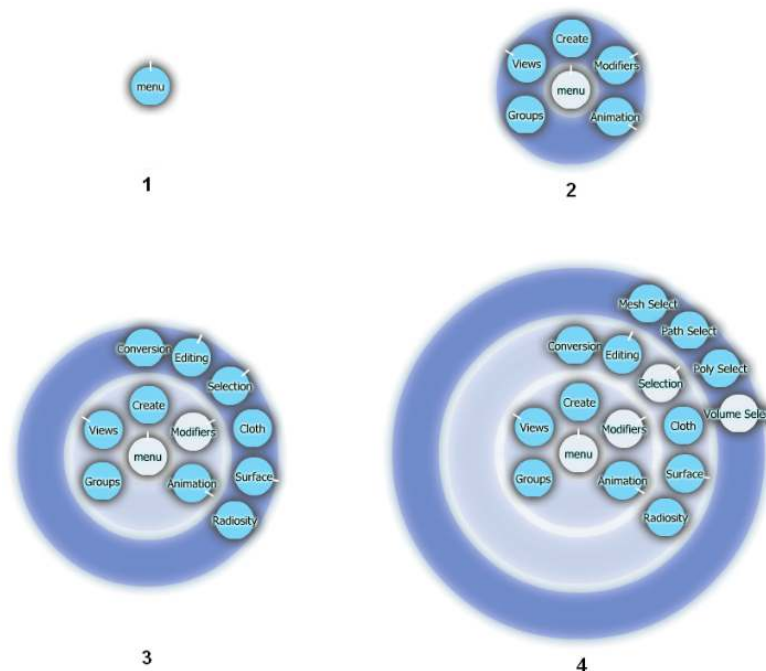


Figure 2.9: The navigation pattern of a three-layer radial menu.

The radial interface is common in video games, as radial interfaces provide a smaller average distance to menu items, which is beneficial in interactions which should be quick [32,

44]. One of the key benefits of a radial interface is that it takes less space than a comparable list menus. Space is precious in this concept interface. One of the weaknesses of the radial interface is visible if the labels on each item is long [44]. By using icons, this problem is voided, if the icons are understandable.

2.2.4 Icons

Using icons in mobile design is a common concept. Schröder og Ziefle explored the concept of a completely icon based interface[45]. Their findings show that it can work, if certain conditions are met:

"results demonstrate that icon-based menus are basically viable and successfully applicable, if a users-centered procedure is pursued and if icons represent prototypical semantic knowledge."

Icons have a positive effect on interfaces by also providing the user with an example. The image in the icon is an example of what the icon refers to. In addition to this, Egido and Patterson found that combining icons with labels is also an improvement in usability over simply using labels [14].

2.3 End User Programming

End User Programming also known as End User Development is defined by Lieberman et al. as:

End-User Development can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact [33].

The most common example of EUP is Microsoft Excel ⁶. In Excel, users are required to use EUP in order to fully utilize the software. All core functionality in Excel are examples of EUP. Other examples are email-filters but also more abstract tasks such as 3D programs ⁷ and very abstracted programming environments such as Scratch ⁸.

When defining and creating relationships between devices in home automation systems, we expect the user to participate in the process through EUP. As I explored in Section 2.1, there are differences in the degree to which the applications expect the user to program behavior. Therefore, we must decide if some of the advanced functionality should be "gated" behind a curtain of programming, or if this should attempt to involve all users equally.

The goal for home automation EUP is for the user to never feel limited in their options, while at the same time not overwhelming the user with information and options. This tightrope balance is important to keep in mind when designing an EUP implementation in this thesis.

Cao et al. explored how end user programming differs from regular programming. They found that in EUP, users were interlacing designing with programming [8]. They

⁶<http://office.microsoft.com/en-us/excel/>

⁷www.sketchup.com

⁸scratch.mti.edu

show that users think differently to professional programmers and designers, meaning they need different tools and user interface elements at their disposal, specifically in regards to design and help when stuck.

There are other software which also employ elements of end user programming. Yahoo! Pipes⁹ is an application which allows users to create "mashups". Mashups can be summarized as web applications which interactively combine data from multiple internet sources [54]. A common example is gathering data such as voting or sales, and plotting them on a map.

Yahoo! pipes provides the user with a graphical interface, where the user can connect pre-defined "boxes" such as "fetch feed" and "sort" as shown in Figure 2.10. These blocks can be connected, and the user is in control of what they want to create.

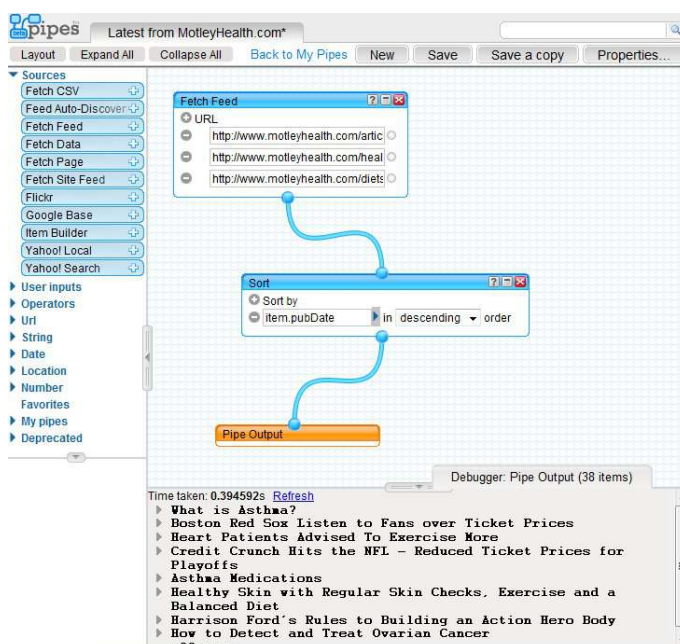


Figure 2.10: Screenshot from the Yahoo! Pipes application showing two "sources" connected by pipes. The output is shown in the bottom of the screen.

These pre-defined functions we saw in Yahoo! Pipes closely resemble the programming-blocks used in the Scratch web programming language. Maloney et al. describe the different design considerations they made to many aspects of the Scratch web application in their 2010 paper "The Scratch Programming Language and Environment" [37]. In Scratch, commonly used control structures such as loops and if-tests are abstracted into blocks, which the user can fit commands or functions into, as shown in Figure 2.11.

Users pull building blocks from a palette on the screen, onto a canvas where they can be used in the code. We can learn some of the design choices behind the implementation of building blocks in scratch in their paper:

- If a block supports parameters, example arguments are provided in order to showcase their application.
- Every block has a help screen, accessible via right click menu.

⁹<https://pipes.yahoo.com/pipes/>

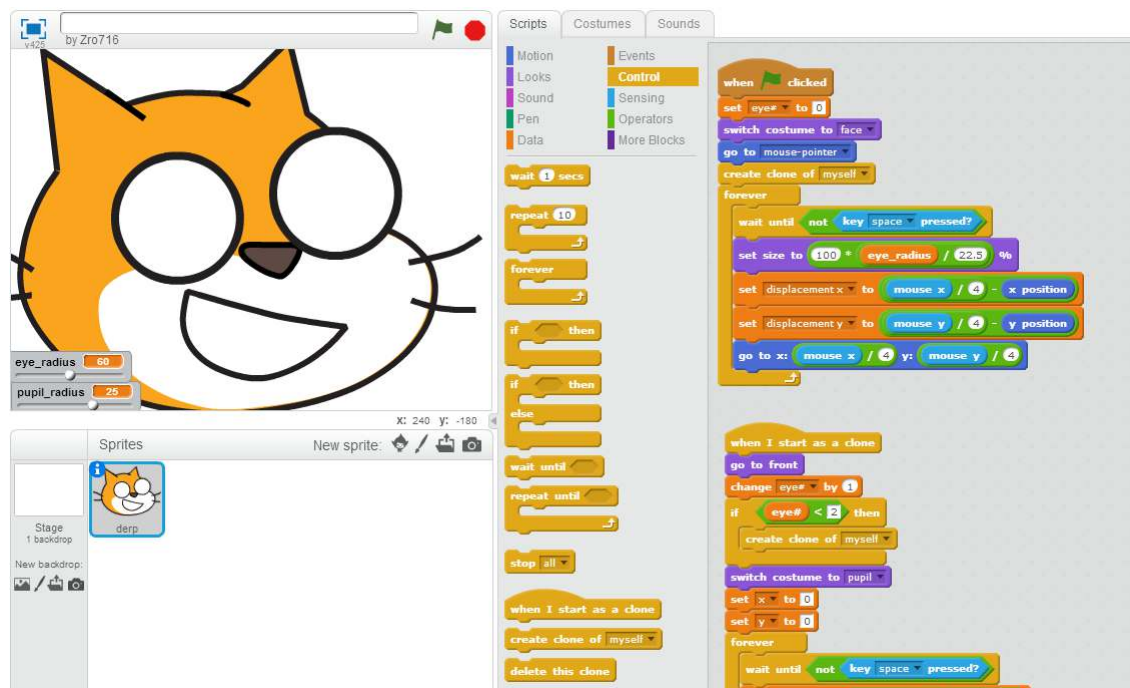


Figure 2.11: A part of a "googly eyes" program made in scratch, where the cat's eyes follow your mouse cursor. Window explanation, left to right: Output, palette, block editor.

- Blocks fit together only in ways that fit, thus eliminating the need for error messages.
- When a scratch script is running, the blocks currently being executed are highlighted with a white border, visualizing the script's progress to the user.

Blocks only fitting together in a certain way ensures that the logic "reads" well. It is also impossible to create syntactically "wrong" statements in Scratch [36]. In regards to syntax "reading" well, Leigh has found that students who are able to break down their algorithms into smaller problems are also more successful in implementing them [51]. Being able to read the logic out loud and pronounce the statements is important, as it could help the users to break down the problem or to think of it as natural language.

Louca has compared two novice friendly programming languages, MicroWorlds ¹⁰, and Stagecast Creator ¹¹. MicroWorlds is a written language, and he observed that novice students were having an easier time reading their logic in this environment, compared to Stagecast Creator, which has an interface resembling Scratch, but with no labels on any blocks. In StageCast Creator, logic is stored inside objects, and is not directly visible or readable. An image of the StageCast creator and the process of making a sprite move is shown in Figure 2.12.

¹⁰<http://www.microworlds.com/>

¹¹<http://www.stagecast.com/>

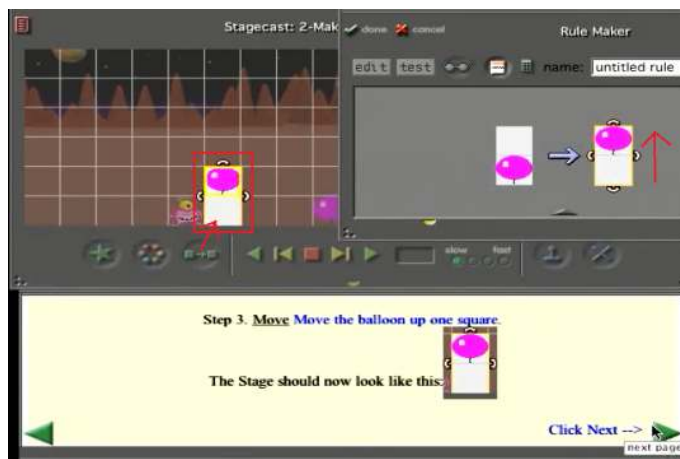


Figure 2.12: The red drawings shows the steps to make a the balloon animate upwards. Image from the StageCast introduction. The "code" or rule maker as it is called in Stagecast is hidden inside the yellow outline around the balloon.

The Stagecast Creator was however easier for the novices to create content with [34]. Scratch is also easy to generate content with, and its logic reads easily as shown in Figure 2.11. Providing easily readable code and ease of development is valuable attributes and part of why I look towards Scratch in this thesis.

Scratch is a successful non-text-based programming language, Maloney et al. [37] interprets Malan and Leitner [36] who found that when exposing entry level programming students to Scratch before seeing a text-based programming language, the majority of students continued to "think in Scratch-blocks" even after moving to text-based languages.

Chapter 3

Method

"So the fate... of the entire human race depends upon my wild guess."

Doctor Gaius Baltar

This chapter describes the methods used for all aspects of the workflow in this project. A large portion of this project is designing and developing a prototype which will be used to gather data concerning the research questions in Section 1.3.

3.1 Idea Generation

The starting point for this research is the Controlling the Home prototype as described in Section 1.4. My path from there to a desirable prototype will be in the hands of users whom I have involved in the project. Their first involvement will be in a future workshop, intended to create an initial influx ideas for the prototype, as well as a direction for the project.

3.1.1 Future Workshop

A *future workshop* is a brainstorm methodology. My application of the future workshop methodology follows the guidelines of Apel set in his 2004 review of the methodology "The Future Workshop" [2]. Apel explains that there are 5 phases of a future workshop, these phases are listed in the table below, what Apel calls "main phases" are listed in bold.

1. Preparation Phase
2. **Critique Phase**
3. **Fantasy Phase**
4. **Implementation Phase**
5. Permanent Workshop

The preparation phase takes place before the workshop begins, and consists of preparing the room by setting up recording devices to document the session, as well as chairs and the likes. I consider this not to be a part of the future workshop, as the participants

have not joined, and no ideas or content are being processed or created. This means I will only discuss the "main phases" of the future workshop.

The next phase, and the first phase in which the participants are involved is the *critique phase*. In this part of the workshop, participants are presented with current solutions, which are then criticised. From previous experience, I have found that this phase concludes naturally when new ideas to the criticism produced are no longer able to be suppressed. The sheer volume of new ideas will eventually take over, initiating the next phase.

The *dream phase* is the second main part of the future workshop. The concept is that after having critiqued existing solutions, the participants now know what to avoid, when they are set to suggest new solutions. During this phase, the participants can envision anything they want, without limiting themselves to a budget either in time, money, or other available resources and technology. This is the fundamental idea behind the dream phase. In order to help the participants create great ideas, I will provide them with both a whiteboard, and an abundance of pens and paper for smaller on-the-fly mockups of ideas.

When the users have created sufficient ideas during the dream phase, the *implementation phase* begins. In this third and final part of the future workshop, the goal is to create an image of a prototype accurately using the concepts and ideas spawned during the dream phase. This result will form the basis of the first design iteration of the prototype. The prototype will be used to provide an answer to the research question presented in Section 1.3. My goal is for the participants to produce a concept, not a final product. The fifth phase, the *permanent workshop* will be detailed on its own in Chapter 4.

Data Collection from the Future Workshop

The future workshop will be filmed. The video will be transcribed and provides more information than an audio recording as I can watch body language and mockup drawing. I will also use a separate audio recording device, as a safety precaution in case something is wrong with the video recording.

I will also take pictures during the future workshop, in order to document the results. Still images of any prototypes or ideas will be the fundamentals behind designing the first iteration of the prototype.

As the future workshop will be a prolonged session lasting up to two hours, there is a lot of information which could be gathered. After having recorded both audio and video from the future workshop, the recording will be transcribed. While transcribing, I will highlight specific elements and conversations which I find particularly interesting. By analyzing the future workshop in this manner, I am able to gather qualitative data such as minor comments which could prove valuable. If a user raises a concern which is too quickly dismissed by the group, or perhaps not heard at all, reliving the workshop and documenting such finds would bring them to the surface. The analysis could then in turn impact how the next part of the project is approached.

3.2 Evaluations

In order to evaluate the prototype both in and after development is finished, I will use evaluations with users and experts. This participatory method will be employed in order to gather opinions from outsiders to the project. This section covers general methods I

will use during evaluations. I will also be making design and implementation, the methods to help me in this is listed in this section as well.

The expert evaluation of the first iteration prototype is discussed in Section 4.3. The final evaluation and its results are detailed in Chapter 5.

3.2.1 Cognitive Walkthrough

During a cognitive walkthrough, an evaluator follows the scenarios which is to be completed, and looks for problems in the flow of the low-fidelity prototype which might cause problems in completing the scenarios or tasks [20, pp. 163-164] A low-fidelity prototype consisting of images will work similarly to a flowchart for this purpose. The cognitive walkthrough can gather more results if the evaluator asks questions about the interactions.

3.2.2 Tasks & Scenarios

Each of the two evaluations will consist of tasks where the users need to use the tools which I have supplied them with to complete. The tasks will be kept similar, so the results and which direction the progress of the concept is going is easier to gauge. There are however some differences - in the first test, the prototype will be low fidelity, while in the second test, it will be implemented as a high fidelity prototype. In addition to this, the final user test will also use the low-fidelity prototype for certain tasks.

For the first expert evaluation, the results are found in Section 4.3. For the final evaluations results, see Chapter 5. The feedback and observations during the evaluations might not be equal parts positive and negative, as users have an easier time criticising something, than praising it. This is connected to how people usually only give positive feedback if something exceeds their expectations [43]. Firman and Edvardsson had similar findings in their study of the complaints and compliments of train goers [15]. This skewing of data would also affect the material gathered from semi-structured interviews.

3.2.3 Semi-Structured Interview

Semi-structured interviews is one of many ways of conducting an interview. Clifford et al. [11] interprets Dunn's [13] view on different interview methodologies. They explain that there are three major branches of interviews: an unstructured interview, a semi-structured interview, and a structured interview. In a structured interview, the questions are always asked in almost the same way, and in the same order. At the other end of the spectrum, is unstructured interviews. These are often more like stories, where the interviewee sets the pace and the topics. The middle ground between these is a semi-structured interview, where the questions are predetermined, but the informant can choose to address them flexibly in their own way.

Barriball and While has consolidated a list of qualities which good interview subjects should have when conducting a semi-structured interviews [35]:

- Comfortable and unstrained in interaction with the researcher.
- They provide solid answers and in good detail.
- Thoughtful and willing to reflect on what they say.

- Generally open and truthful.
- Ability to stay on topic or related important issues.

The first three points in this list are qualities which can be easier to find in subjects which you know personally. Helping me should be the only incentive which users have when participating, meaning they will most likely attempt to aid me as much as possible during the workshop or evaluations.

At the same time, the last two points are issues which could be problematic. While instructed to be truthful, being completely honest and objective can be more difficult for subjects whom I have a relationship with. I trust my peers however, and I have received negative feedback from peers before, and trust to do so again. The last point is especially problematic, as going off-topic happens more frequently when interviewing someone you have a personal relationship to.

For more discussion around the bad qualities whom a interview subject could possess, Oppenheim [41] states: " Perhaps the most important determinant both of response rate and of the quality of the responses is the subject's motivation." As my subjects will have no other motivation or incentive than to help me and my research, I believe peers will provide the most valuable feedback during testing.

Naturally, the flow and order of questions in a semi-structured interview might differ between users, unlike in a structured interview. Therefore, I need a method of interpreting and analyzing these results. For this, I will use Open Coding.

3.2.4 Open Coding from Grounded Theory

Grounded theory is a qualitative analysis method. It was discovered by Strauss and Glaser and is described in their 1967 book "The Discovery of Grounded Theory" [18].

Open coding is described by Strauss and Corbin [50] as:

"The analytic process through which concepts are identified and their properties and dimensions are discovered in data."

This process allows researcher to aggregate data into categories, which can then be discussed and interpreted separately.

A big potential weakness of open coding presents itself if a team consisting of all members of the team is not present throughout the entire process of open coding [21]. Being alone in this project, this does not affect me. Instead I will be able to follow each part of the process. This could in turn mean that if I interpret something wrong, there is nobody with similar knowledge of the data who can offer a second opinion.

Following the first pass of open coding, I will move into *selective coding*. As hoda et al. describes it: "once the main theme of the research is clear, move to selective coding."

The results from the selective coding process are presented in Chapter 5.

Using categories resembling the research questions is natural, in addition to tags with a high frequency in the data. Complimenting these tags will be "what is being interpreted", as well as "what the interpretation is" . This should provide opportunities for categories and patterns to emerge from the data.

I will keep track of each of the new tags I selected through selective coding, so I avoid using more than one tag to describe the same thing, for example using both "cluttered" and "messy" which could be the same tag. The results compiled from this process are presented in Chapter 5.

3.2.5 Think Aloud

The "Think Aloud" technique can be applied to many different testing and evaluation forms in software design and development. Someren et al.'s 1999 literary review "The Think Aloud Method" describes the technique [1]:

"The Think Aloud Method consists of asking people to think aloud while solving a problem and analyzing the resulting verbal protocols."

From previous experience with the think aloud technique and its applications - instructing and urging users to think aloud while solving a problem will make both the interpretation of the results better, but also the results themselves. Users can sometimes see things differently when putting what they are thinking into words. It can also be easier to understand what is intuitive to the user, as users sometimes discard their first hypothesis of how an application works very quickly upon discovering that it works in another way. This insight to what they considered intuitive but dismissed quickly could be lost without think aloud.

This process is however not without challenges, Someren elaborates on what he finds to be one of the limiting factors of Think Aloud in his 1994 Report "The think aloud method: A practical guide to modelling cognitive processes" [53]:

"..if the information is nonverbal and complicated then verbalization will not only cost time, but also space in working memory because it becomes a cognitive process by itself. This will cause the report of the original process to be incomplete and it can sometimes even disrupt this process."

In similar thoughts, Khatami et al. [31] interprets Patrick Jordan who finds the same problem in his 1998 report "Revising computer documentation for comprehension: ten exercises in method". Jordan found that think aloud is asking the participants to perform two tasks - first is to perform for the research, and the second being to ask of talking about their thoughts. Sometimes these two tasks can clash together, and if they are having difficulties in the first task, it will be amplified as part of the working memory is being used to verbalize their thoughts [28].

This is a concern to keep in mind when using the method, thinking aloud should be a tool, but like a hammer, think aloud needs not be applied to all problems. With this in mind, should Think Aloud prove difficult for users, the following semi-structured interview will hopefully shed light on the topics which arose during the task solving.

Strike Team

As a small real world application of the 'think aloud' technique, I will employ what Blizzard Entertainment refers to as "Strike Teams", when they design and test their games internally ¹. Lead Game Designer Dustin Browder describes the Strike Team methodology as:

"Different people come into projects they're not a direct part of, play through the content that has been created, and give their feedback.

And it's the most brutally honest experience you could possibly have in your life. They'll just look at you point blank and say, 'Well, you're not gonna release this cause it sucks, right?' I mean they'll just tell you whatever."

¹<http://www.polygon.com/2014/5/21/5723572/heroes-of-the-storm-making-of-blizzard>

In order to gather quick feedback I will employ strike teams. The work of this paper is done largely in an open workspace at the faculty of computer sciences at Østfold University College. I therefore have other students with similar background to myself available at a minute's notice.

The strike team methodology allows for quick feedback on smaller isolated functionality in the design. While the original strike team methodology only highlights problems, I will also be open to suggestions.

One can also consult himself, by using what is known within the Information Technology field as "Rubber Duck Debugging".

Rubber Duck Debugging

Introduced by Hunt and David in their 2000 book "The Pragmatic Programmer". The book explains how useful it can be to explain your problems to someone else, but sometimes there are nobody nearby for you to consult. Especially useful when trying to find the source of a problem, explain the problem to someone else, and have them look over your shoulder, bobbing their head as you speak. Similarly, you could use an actual rubber duck to explain your problems to, which if it was in a bathtub, would bob its head in the same fashion.

Author David Thomas was introduced to the rubber duck when he was studying at the Imperial College in London, working with the developer Greg Pugh. Greg carried around a small yellow rubber duck, which would sit by his terminal when he worked. It was a while until David eventually mustered the courage to ask...

As such, the rubber duck provides programmers with a physical entity to explain their problems and trains of thoughts to [24, pp. 124-125]. The duck needs not reply, as simply stating the problem out loud can help solve it. This is a useful tool both when writing reports and code.

3.2.6 System Usability Scale Questionnaire

Interviews and task solving during testing involves the evaluator providing feedback and ideas which both criticise and praise the application. Therefore, even when asked directly what their overall impression of concept and its abilities to solve tasks, it can be difficult to gauge what the evaluator thinks overall. For this, I will use the small 10-part questionnaire "System Usability Scale" as introduced by John Brooke [6]. The SUS concerns issues with usability, and ease of control. SUS scores are not percentages, and Figure 3.1 instead shows how we put a letter grade to the SUS' results.

The SUS curve presented in Figure 3.1 does not contain the grade "E", this is not an error. Grades A-D are "passing" grades [4].

SUS scores are calculated in a unique way. Questions award 0-4 points, for positively worded statements such as 1,3,5,7,9 - the score is scale position minus 1. For negatively worded statements such as 2,4,6,8,10 - the score is 5 minus scale position. Adding these scores together and multiplying by 2.5 will yield the SUS score.

As the graph in Figure 3.1 shows, SUS results are not distributed linearly. Therefore, two telling factors from the results of the questionnaire will be the average score which users provide, and if any question consistently scores low across several users. From experience, as well as how the grades are distributed in the SUS, a lower score on one question is more telling of a problem than higher scores are of something working excellently. As such,

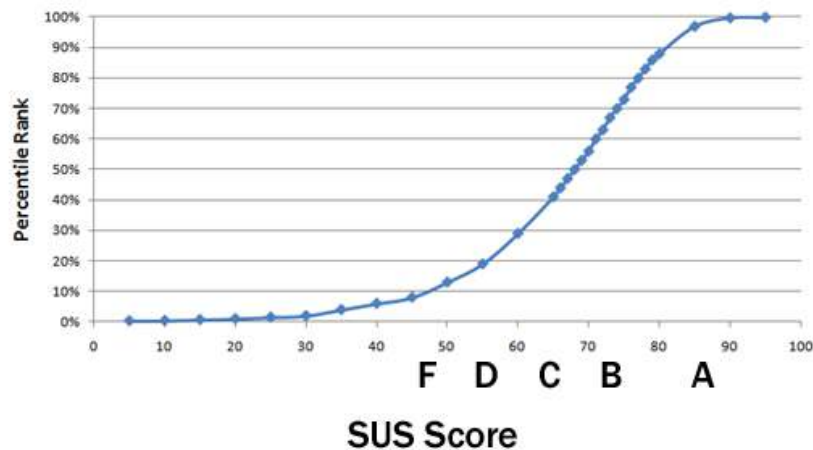


Figure 3.1: The System Usability Scale grading curve. 68 (C) is considered average.

these two factors will be the two interpretations of the SUS scores after my evaluations. An empty SUS questionnaire is enclosed in Appendix B.

3.3 The Prototype

This project will employ the use of two prototypes, a high-fidelity, and a low-fidelity prototype. This section will describe the functionality of each of these, separately.

The design process of the prototypes is described in Chapter 4.

3.3.1 Low-Fidelity Prototype

The low-fidelity prototype is a tool developed in inVisionApp ². The prototype allows the user to navigate through the graph interface in order to complete simple tasks. As is the nature of a low-fidelity prototype, it only allows the user to do the tasks, in a specific order [20, pp. 170-171]. The tasks which the low-prototype facilitates the completion of are enclosed in Appendix C.

3.3.2 High-Fidelity Prototype

This application allows users to fill in if-tests with different logic operators.

As shown in Figure 3.2, there are three types of blocks necessary to completely fill in an if-block. These are the circular device nodes, the square operator blocks, and the on, off, percent, or timer blocks. The user can drag new blocks into the editor from the toolbox. These blocks can be moved around and will snap into place when they are close to a block.

The user can interact with the "Logic Operators" toolbox, and the device nodes which are used in the tasks. None of the tasks require the user to change room nodes, they can therefore not be interacted with. Users can drag device nodes into the relationship editor, which will spawn a copy of the device node. This device node will snap into place when dragged close to its assigned slot in an if-block. There are three types of slots for blocks

²invisionapp.com

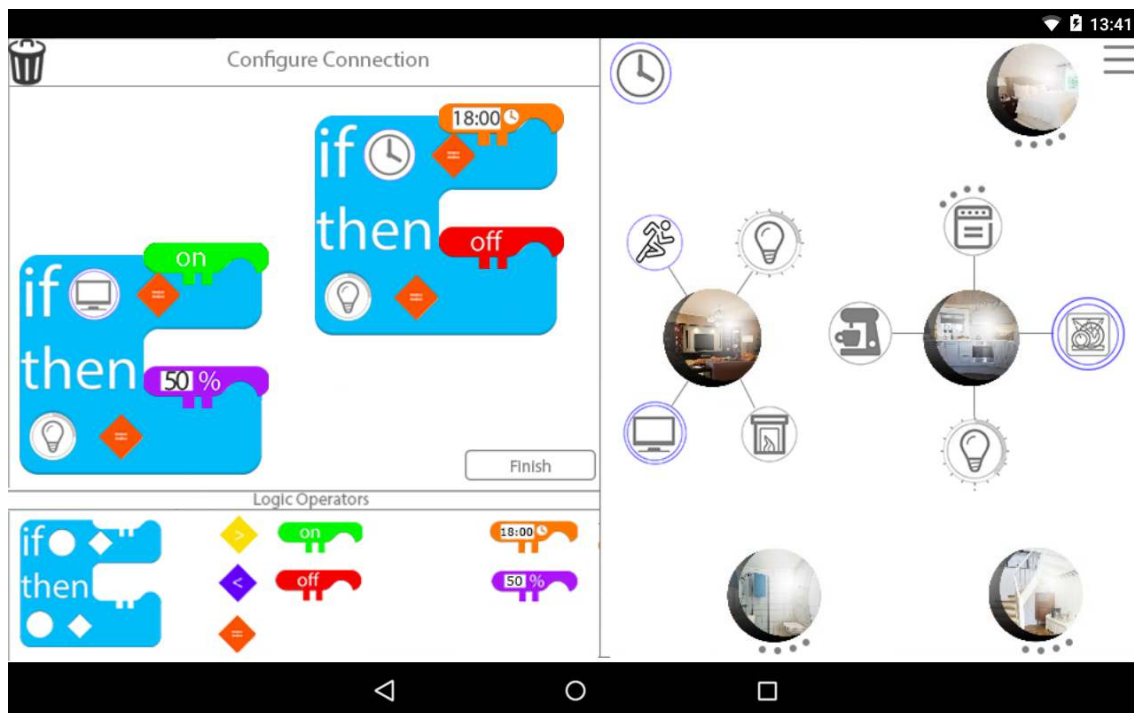


Figure 3.2: A screen from the high-fidelity prototype. The user has used two if-blocks to create logic on the light device.

in the if-test. The user can drag these nodes around, which is something which received positive responses in *Controlling the Home*, where users stated that they had a lot of fun dragging the nodes around [47].

Dragging a new logic operator from the toolbox will spawn new ones onto the editor. Users can drag them into the trashcan in the top left corner in order to delete them. Deleting an if-block will delete any operators attached to it as well.

The source code for the prototype is available at GitHub ³.

3.4 Summary

In this chapter, I have presented the methods which I will use in order to answer the research question of this master thesis. I have looked at methods in the way of gathering results during testing and design, as well as theory behind testing. The prototype which will be used during evaluations has also been presented.

³<https://github.com/Martineskerud/CTH2.0>

Chapter 4

Prototype

”Ideas are like rabbits. You get a couple of them and learn how to handle them, and pretty soon you have a dozen.”

John Steinbeck

A large component of this thesis will be the development of a prototype. The design will be derived from an initial future workshop, and refined through subsequent expert evaluation. The prototype will be my tool for answering the research question described in 1.3. The prototype’s design will be detailed in this chapter, beginning with a future workshop, then two design iterations and final testing. The next section details the future workshop.

4.1 Future Workshop

The future workshop methodology is detailed in Section 3.1.1.

I began the future workshop by having all participants sign a form of consent. This form is enclosed in Appendix A.

The participants in the workshop were three women and three men aged between 20 and 26. I have a personal relationship with all the participants, and their prior knowledge to the project was varied. All participants were computer science students. Table 4.1 contains a brief background on all participants.

Id	Sex	Age	Knowledge to the Home Automation Domain
1	Female	25	Co-author of Controlling The Home - Large knowledge
2	Female	25	Tester in Controlling the Home project - Some knowledge
3	Female	20	No prior involvement with Home Automation - Little knowledge
4	Male	24	No prior involvement with Home Automation - Little knowledge
5	Male	26	Tester in Controlling The Home project - Some Knowledge
6	Male	24	Tester in Controlling the Home project - Some knowledge

Table 4.1: An overview of the evaluators in the first test.

The participants were urged to think aloud throughout the session, as it would aid me in gathering data from the discussion [1].

The future workshop was a collaborative effort and all parties were equally involved. If the users were discussing healthily between each other, and I had no urgent input to contribute, I would sit quietly and observe. If the discussion died down, I would ask questions and try to incite more discussion.

4.1.1 Data Analysis

The future workshop was documented using both video and audio recording. The recordings were then transcribed the next day. This was to help keep the discussions in mind.

The data were then analyzed through a simplified version of open coding due to time. During transcriptions of the audio and video, I would highlight conversations which I saw had relevant discussion.

The first part of the future workshop, the critique phase is detailed in the next section.

4.1.2 Critique Phase

I began the critique phase by presenting the users with some background for the thesis and why they were here. I highlighted some of the challenges previously faced in the Controlling the Home project, as well as the challenges for this prototype and project. In addition, I highlighted Scratch as an example of an EUP application because I knew the team had worked with Scratch previously, and expressed my thoughts on EUP being a means to solving these problems.

I had prepared the prototypes I looked at in Chapter 2 to show to the participants. I highlighted the Controlling the Home prototype, the Ninja Sphere, Home Maestro and HomeBLOX. I explained how their solutions to both creating and representing relationships worked, and we discussed what we found displeasing about existing solutions. If the discussion staled or halted completely, I would try to bring up some of the problems which had previously been found in the Controlling the Home prototype, as well as features which I found cumbersome in the other aforementioned prototypes. As several users had some previous experience with home automation, their criticism also went beyond the prototypes which I presented. An example of this is the prototypes which were looked at during Controlling the Home as related work, which some users still remembered. During the critique phase, the topic changed often, and small topics which should be considered by the interface were discussed. One example of this is the concern for overloading circuits, which the participants did not find that current solutions considered:

User 2: *"Are you [me] going to consider the circuits which the devices are connected to? So it doesn't become too much?"*

User 6: *"Yeah that's a good point"*

Me: *"Good point!"*

User 6: *"You should check how much each circuit in each room can take, so you avoid switching everything on at once."*

As is shown in the quotes above, the line between critique phase and dream phase was thin. The suggestion above is indirect criticism, as to our knowledge, no current solutions consider circuit overloading.

The participants were eager to suggest improvements once a current issue was identified. Two users discussed the pros and cons of limiting the user:

User 1: *"The user shouldn't be able to connect every device to anything, unless you are a superuser."*

User 6: *"I like it in theory, being able to do anything. That is actually really cool in theory i think. But I can only imagine what would happen if you would give that to someone with less technical insight, they could start something and mess everything up."*

Instead the users reached a compromise, regarding which devices should be allowed to connect to one another. This solution seemed inspired by Scratch specifically:

User 1: *"I think everyone should be able to switch everything off, atleast, except for some things. (...) Except for fridges, and heaters in the winter. The user should not be able to switch those off."*

This conversation took place after I talked about the error handling in Scratch, where users are not allowed to make syntax errors, but where logic errors are fine.

User 2 had opinions on the Controlling the Home way of visualizing the state of a device:

User 2: *"I don't know, [...] I don't understand why some of these [device nodes] have double lines, half lines and such. So no, that I don't understand."*

[I explain the current solution]

User 6: *"What about floor heating?"*

Me: *"That's an inconsistency."*

User 1: *"Floor heating works, you can switch it on."*

Me: *"Yes, it's 'on' or 'off'."*

User 3: *"You don't think you could select the temperature?"*

[...]

User 2: *"Because I would've used colors or color brightness in some way. Red colors to symbolize heat, and brightness to indicate light. The way that's... Intuitive."*

The inconsistencies user 2 are describing is depicted and highlighted in Figure 4.1.

I also attempted to introduce the users to the concept of involving sensors in the interface. This conversation was derailed however, but led to a bigger discussion on device nodes:

User 6: *"Are they [sensors] visualized?"*

User 3: *"Not at all. No."*

User 6: *"That sounds right cause I can't see any."*

User 5: *"I think you're onto something there, the need for visualizing what the different nodes are. [...] Representing a room could be different from a node. A device like a coffee maker should be represented in a different way. This would create different expectations. So what I think one of the biggest flaws right*

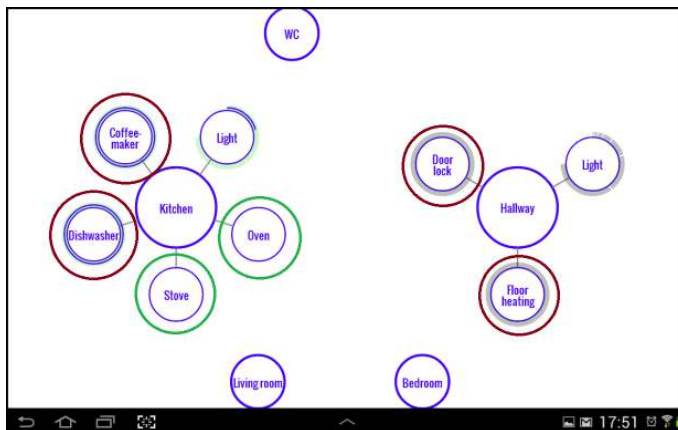


Figure 4.1: Inconsistencies in the Controlling the Home prototype. The red circles are devices which should behave similarly, but the pair in the kitchen node has double blue lines, while the hallway nodes have one blue line and a thick grey outline. The grey outline is intended to be 'off', but this was not clear to users. The green pair has only a blue line as they have child nodes of their own similar to rooms, but the users did not understand why they had no border.

now is, that in order to quickly sort the different devices you need a different representation for each."

User 2: *"Yeah, you're thinking icons and such?"*

User 5: *"Yeah icons maybe. But it's also the part of trying to maintain the circle [circular node visualization], it's become kind of a 'thing' in the graph interface. So if..."*

User 2: *"We could have circular icons..."*

User 5: *"I have a wild idea of how this could be visualized"*

The 'wild idea' which user 5 described is depicted in Figure 4.2. Sensors would later be revisited in the dream phase.

The critique phase was the shortest of the phases. The users were eager to suggest improvements. The conversation above was the transition from critique phase into the dream phase.

4.1.3 Dream Phase

As the users were running out of criticism towards current solutions during the critique phase, the session naturally transitioned into the dream phase. Figure 4.3 shows some of the participants discussing during the dream phase. Of the three phases in the workshop, the dream phase was the longest at 50 minutes, while the other two phases were 20 minutes each. There was no fixed duration to the future workshop, but I had told the participants in advance that the workshop could last up to two hours. I provided lunch to justify the duration, which all participants enjoyed.

The users suggested new solutions throughout the entire dream phase, ranging from small improvements to larger concepts and implementations. The next part of this section is divided into topics resembling the research questions for this thesis. Often pros and cons for each suggestion is described, while this is not the critique phase, it was natural

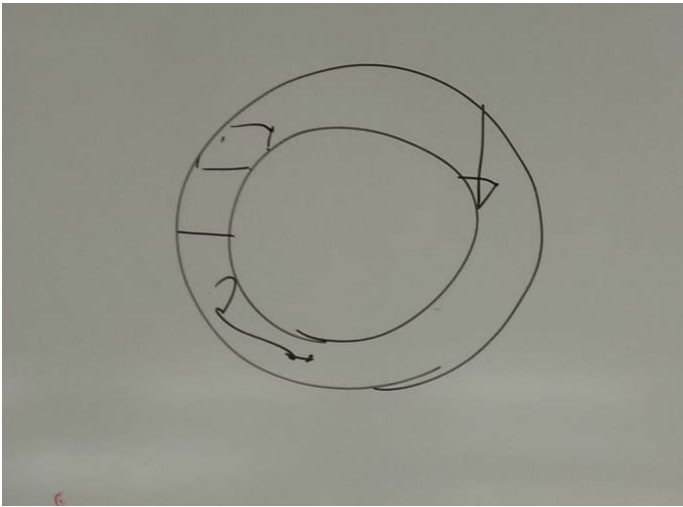


Figure 4.2: A new suggestion for the room node, the border would contain an image of the room, while the middle could be used for an icon or other functionality.



Figure 4.3: A discussion around an idea between the participants.

for the users to explain their thought process behind each suggestion, as well as to reflect around any problems they thought of by using think aloud. The second subquestion to the research question, concerning a non-text-based language is not found in the dream phase. This became a research question after the results from the implementation phase were processed.

The next section details the group's ideas regarding integrating relationship functionality.

Integrating Relationships into the Interface

The participants focused a lot of their efforts around topics relating to the first research subquestion:

How can the functionality to create relationships between devices be integrated into a graph based interface for home automation?

While the dream phase and implementation phase were kept separate, users would sometimes draw their ideas during the dream phase. Some of these drawings, such as the previously shown drawing in Figure 4.2 were created during the dream phase, but will therefore also be referred to in the next section detailing the implementation phase.

Integrating the relationship functionality into the interface was a big topic during the dream phase. The previous discussion led the users to a new idea, where the user would be able to create logic through the interface by using either a "simple" or an "advanced" editor, based on their skills:

User 1: *"There are a lot of household devices which have a lot of options, like a dishwasher. The Samsung smart devices can give feedback too."*

[pause]

Me: *"Has anyone considered that the current way of creating a connection is too quick? [insinuating lack of options] It's just drag a node, longpress, and drop it on top of another."*

User 2: *"What I think is that at some point [when defining a relationship] you should get a new screen with options."*

Me: *A clock and the likes, sensors?*

User 2: *"Yes anything that has to do with that connection which can be used to tweak. Timers, rules, those kinds of things. You could look towards the Scratch logic."*

User 5: *"I think you should go for two interfaces. You have the simple user, and you have the advanced user who has a lot of demands. So if you have the old simple press [the way you made connections in Controlling the Home], then you'd just be presented with 1-4 small quick choices."*

User 6: *"Yeah, I was thinking a little pop-up next to it"*

User 5: *"Could be like 4 choices for instance?"*

User 6: *"Yes just simple choices."*

User 5: *"Yes, but you need the option of going in depth too, like a 'deep editor'."*

User 2: *"Yes, 'show more options' or go to 'advanced' button of sorts."*

User 5: *"You would be able to do more there. [...] What's interesting when making these events yourself is knowing which properties the devices have, and how you can access them. But it has to be different for each device, and you just have to accept that this can't be standardized in this way. [...] At the same time, it would be interesting to have different events to listen to, which causes something, like an 'on', 'off' or 'percent' event. Like if a light is set to 20% strength, or 30% finished if it [the device] has progression."*

In order to use the advanced editor to its fullest potential, users would be able to browse the graph network and select nodes which they would keep in their toolkit and have available when creating logic in the relationship editor. The connections created in the relationship editor would instantly update the view in the graph, providing the user with feedback which assists their feeling of directly manipulating the graph's content.

The Controlling the home prototype does not visualize relationships unless both devices involved in the relationship's rooms are shown on screen. I wanted some feedback on this, and went back to the slides I initially showed the participants during the introduction. The Controlling the Home slides which highlight "invisible relationships" are shown in Figure 4.4.

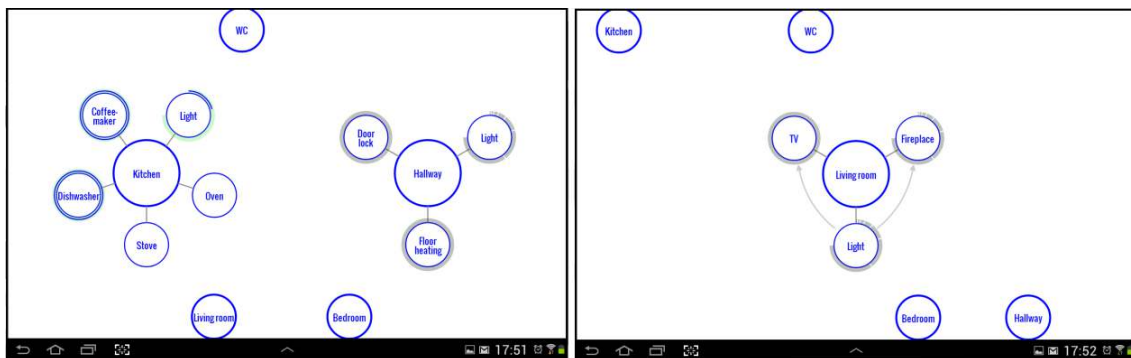


Figure 4.4: The pictures presented to the users of the Controlling the Home prototype. Here, there are relationships between devices in the light room, which are invisible when the room is closed.

The users had suggestions for clarifying the "invisible relationships":

User 2: *"If it's like this then, you got two rooms, living room and bedroom, then now there's no connection between them, because they're not both visible. The TV here is connected to the bedroom, so that connection would show up?"*

User 6: *"There's an arrow between them yes?"*

User 2: *"And if we pretend they're both collapsed, then there would be an arrow between them, and if you expand then it appears in the right place."*

Me: *"What if, if there were more than one connection [from a room to another], then there would still only be one arrow, to avoid information overload? You would see 'that there is something here' and then you could dig deeper to find more."*

User 3: *"You could have different types of arrows, which look a bit differently if you have more things connected together, but you don't have to see which until you open it up."*

User 1: *"Room connections could be thicker too."*

Another discussion also concerned the relationship arrows. In this discussion, the users talk about information in the relationship arrows, and not their visibility directly:

User 2: *"So you can press the line [relationship arrow] to get more information then?"*

Me: *"Not currently no -"*

User 2: *"Well that could be a thing?"*

Cause you have to be able to configure the rules one way or another, delete them or whatever. But I actually think a line like that works."

User 5: *"A label, text, or something is missing on that line [the relationship arrow]. The label would describe the event that's happening. For instance, if the line is connected to a relationship where the light goes on, but not off, it would be nice if there was an indication that this was an 'on event'.*

At the same time, if there are a lot of events, then it can be a challenge not to clutter the interface either."

Another topic which was discussed, were visualization of device utilities, as well as sensors in the interface.

Device Utilities and Sensors

The next research subquestion in this thesis is:

How can device utilities and sensors be visualized in a graph based home automation interface?

A major suggestion for the interface's device utility visualization was made by user 5:

Participant 5 presented a comprehensive idea to the structure of a home automation interface, which was described as an operating system by itself:

User 5: *"I think we're missing a step in this process, and that is when I open up the TV for example. When you click the TV'n again, I miss a way to open, kinda like android, so it opens an app which is connected to the TV. This way, you can swap things out, and develop things yourself. Lets say this comes from Android Marketplace."*

Me: *"Do you mean like a literally an app? Or do you mean like VGA is an app, HDMI is an app, and so on?"*

User 5: *"I think that with the TV, I want an interface to be displayed where I have the various channels available as buttons, maybe a TV program once I've navigated to a specific channel. There are a lot of different ways to visualize a TV remote, but it would have a completely different interface than the oven would for instance. I think there's a need for different app structure on each different element [device] when you can want to switch things on or off. I think we're closing in on describing an operating system here though."*

In addition to this, many suggestions were made to functionality which is related or which would impact device utility visualizations.

One such suggestion was to remove the text from nodes, and introduce icons. This is a continuation of a previous discussion regarding icons:

User 5: *"If a graph still is a graph, with some margin around the border here [user is drawing on whiteboard, the result is shown in Figure 4.2]. Well if this is a lamp, and this is the end of a couch. You won't see the entire room, just parts of it. So imagine now that this is a picture of a living room, then there are certain things which we expect to be in a living room, but still a large part of the photo is missing."*

[...]

User 2: *"I would suggest that instead of having it say 'coffee maker' here, that there was an icon of a coffee maker. But it depends how well that would work. Can you recognize the difference between a dishwasher and a washing machine?"*

User 5: *"I agree, if a coffee maker is introduced to the system, it should have an icon which resembles the coffee maker you got. Not a general icon. Like when you add a printer to your network, it tries to resemble the printer's look."*

[...]

User 6: *"I can see some problems with suddenly introducing icons on top of this interface. It's pretty clean the way it is, using the text only. Or maybe it would be cleaner to use icons..."*

User 2: *"I'm thinking icons, not miniature photos."*

User 5: *"Yes icons, pictures were more for rooms."*

User 1: *"You could also consider that there's a lot of different people who could live in one home. They might not all be able to read, like kids for instance."*

User 6: *"Good point!"*

User 1: *"It's supposed to be a simple interface though, so it depends if you're able to have enough unique icons."*

This discussion shows the back and fourth dynamic between the future workshop participants. The users indicate a problem, with too few icons being available, and other participants contribute existing solutions such as drawing parallels to printers. In terms of room nodes, using pictures is something which comes easy to the platform, as most mobile devices have an integrated camera. The users thought that adding icons to the interface, and pictures to room nodes could clutter it, especially pictures. They were open to trying it however, seeing how there are only a few room nodes in the interface.

Pictures were not the only suggestion to room nodes, light visualization was also discussed. User 2 points back to her previous ideas of using color in the interface:

User 2: *"This relates to what I talked about earlier, I don't know exactly how it would work for all appliances, but maybe lights could have a glowing effect, while warm devices could glow red. Depends how this will all look when you combine it together however."*

[...]

User 5: *"In regards to energy visualization, or light visualization. I talked about imagery and peripheral vision, even if this is more central vision though it might not matter, maybe have the light in the room reflected in the photo? If the picture is greyed out, then there's no light in the room, but more light indicates that it [the light device(s)] is on."*

This was feedback which resembled ideas which had previously been discovered in Controlling the Home, while not mentioned in the paper as it was not included.

Sensors Sensors are a part of the tool kit available in home automation solutions, and is frequently used [7]. Integrating and facilitating their use is therefore potentially beneficial to the interface. I previously tried to bring up the topic as shown above, but instead of focusing on visualization, the participants dreamed about possible applications involving sensors:

User 5: *"Let's talk sensors again, like a movement sensor. You could connect them to events, then say if you've activated vacation mode [a configuration of lights while on vacation] then if sensors are activated, you get an email. Maybe a camera too so it can send the photo by email"*

User 6: *"If sensors are triggered while in vacation mode, you have a security breach."*

User 1: *"Fire alarms!"*

User 5: *"Fire alarms too, all that stuff can get connected."*

User 2: *"Maybe you could get a message when you have to replace the batteries in the smoke detectors? That would be useful."*

In summary, the group found sensors to be useful, but did not suggest anything in terms of their actual visualization.

Device States

The final research subquestion is as follows:

How can the state(s) of devices be visualized in a graph based home automation interface?

Despite the previous criticism towards Controlling the Home's device states, no suggests were made to improving them. Similarly to sensors however, several suggestions were made to functionality which would impact the device states or which was related to the problem domain. Closely related to the integration of relationships in the interface, a new menu was suggested which would provide an overview of active devices:

User 2: *"You could use a list to get an overview, then you could have a button to see which devices are on, just like a regular list. Then you could switch things on or off too."*

[...]

User 1: *Then you would get a list of different things [devices or connections].*

User 5: *"Even though there is a lot of focus on the graph here, it could be natural to use lists in certain situations here."*

This list was later concretized during the implementation phase, and is shown in Figure 4.7.

The users thought there were ways of getting additional information from the device nodes. The participants looked towards Google Now in how the system could be made context sensitive:

Me: *"What's Google Now ¹?"*

User 1: *"It's the thing that tells you the weather."*

[...]

User 1: *"What you could have though is when walking into a room, that node pops up automatically!"*

Me: *"I like that!"*

User 3: *"Is it possible to have different users here? It would be neat if you could have preferences so when you walk into a house, it knows temperature and so on. I like it a little colder for instance. It would be natural if there are more users."*

[...]

User 1: *"But if you're doing something, it shouldn't interrupt you[swap room while if the app is open]."*

[...]

User 6: *"I think there's merit in looking towards google now here, if you open the app in the room, but walk around with it then nothing will happen [when you change rooms], You could have a function so that you shake it to swap rooms for instance."*

This is very similar to how Bill Gates' home automation system already functions as mentioned in Section 1.2.

The next part of the future workshop would be the implementation phase after a lunch break. The concepts which the users wanted to develop further into mockups will be detailed in the next section.

4.1.4 Implementation Phase

In order to take something concrete away from the future workshop, we started the implementation phase by creating a canvas on a whiteboard, allowing the participants to draw interface suggestions.

The suggested designs from the Future Workshop is illustrated in Figure 4.5, 4.6 and 4.7 below. Figure 4.5 shows an "advanced view" to a proposed relationship editor. The

¹<http://www.google.com/landing/now/>

author of this idea described it as "similar to Visual Studio"² in the way the properties panel folds and provides more options.

Some participants found that the solution depicted in Figure 4.5 could be too complicated for a hand held tablet device, and that there was a need for a simpler interface. We suggested an interface using elements from the Scratch application described in Section 2.3. The suggested design for this middle ground between the "Visual Studio" solution and the simplified solution found in the Controlling the Home prototype is depicted in Figure 4.6.

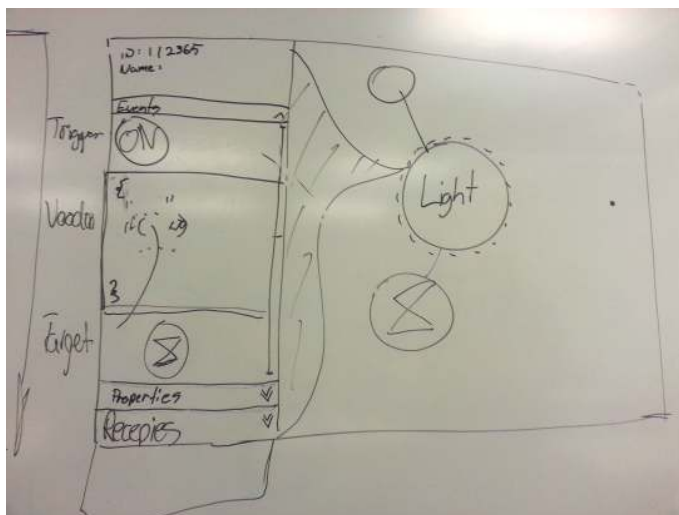


Figure 4.5: This was thought to be the "advanced view". To the left, the user can actually program directly into the interface. Two thirds of the screen is used to allow users to stay connected to the old view, not losing their sense of position in the interface.

This solution would use the entire screen space for the relationship editor. Concerns were raised that this could be too different from the main graph view, and the user would lose control, not knowing where they are. A solution to this would be to use design techniques such as good animations in order to clarify the user's position in the interface.

A mockup was produced of a separate idea which would directly address issues found during evaluation of the Controlling the Home prototype. A side-bar menu which would slide in similar to how the Facebook Messenger³ app interacts with the Facebook app⁴. This suggestion is depicted in Figure 4.7.

This menu or layout solution is referred to as an "off-canvas-layout"⁵. The users could not quite decide on the purpose of this tab. The options were very similar, albeit with small differences:

1. A list of active devices. During testing of the Controlling the Home prototype, we asked the users to switch off all lights in the "house". In order to solve this task, we found that the users needed to iterate through all room nodes, which many found tiring. Providing the user with an overview of active devices would make this process

²<http://www.visualstudio.com/>

³<https://play.google.com/store/apps/details?id=com.facebook.orca>

⁴<https://play.google.com/store/apps/details?id=com.facebook.katana>

⁵<http://zurb.com/playground/off-canvas-layouts>

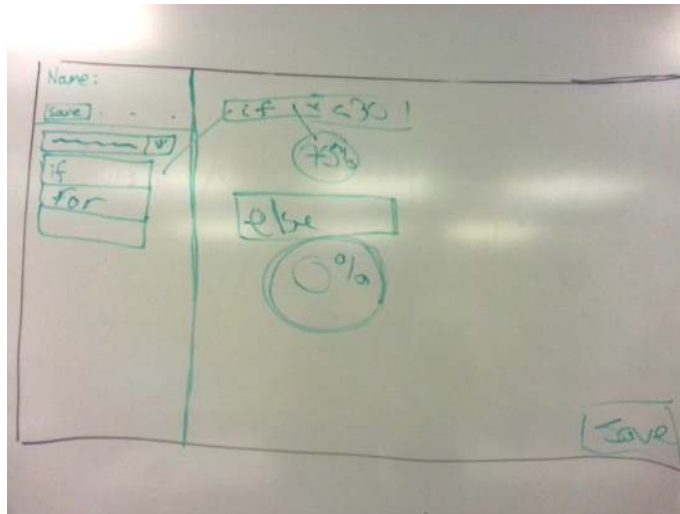


Figure 4.6: The "simple" editor. Here users can use pre-defined structure blocks to create logic between devices.

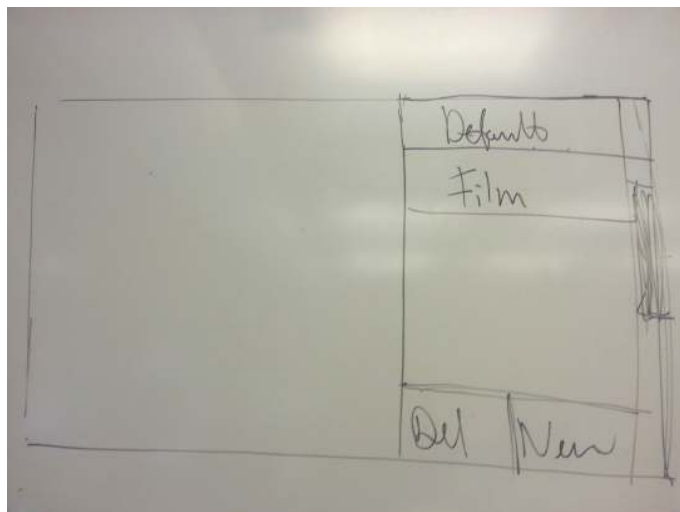


Figure 4.7: A side-scroll bar showing scenes and active devices. This was thought to be a small addition to the prototype, which could increase the overview of the application.

much faster. While lists were something we wanted to avoid in the Controlling the Home prototype, the users stated in unison that sometimes lists were the correct option.

2. A list of active connections. Similar to scenes, but would list the devices in pairs, which could be more accurate, but also more cluttering. This would address a problem discovered during testing of the Controlling the Home prototype - as a relationship between two devices would only show if both device's rooms were open, the user would need to iterate through all combinations of rooms to find all active relationships. With 5 rooms, this meant 25 different combinations of rooms, which took each user about a minute. For providing an overview of your active devices, it is clear that this solution was non optimal. This list would specifically address this issue. Furthermore, if the user creates a relationship which defines all lights to switch off, this would make the scenario previously mentioned easily solvable.
3. A list of active scenes. A scene would serve as several devices grouped together, or a group of relationships.

A followup idea to the swipe-tab was to enable two swipe tabs, thus being able to implement two of the ideas from the table above. This would mean one tab on the left, and another on the right. Whether the user should be allowed to keep both tabs open at the same time was not discussed.

The language to be used inside the non-text-based editor was also discussed:

User 5: *"[...] think it needs to be simple structures"*

User 6: *"Isn't that why you wanted something Scratch-esque?"*

User 1: *"Can we draw a square there for some scratch suggestions?"*

User 5: *"I would rather save that space for having 'recipes' there instead."*

Me: *"Recipes being a pre-defined structure, like a template?"*

User 5: *"Yes, pre-defined rules like for instance an if-structure someone else has made, or which you've made yourself. Like if-this-then. Then you can drag one of these structures up and you don't have to write it yourself"*

[...]

User 6: *"And a pull down menu here, which says something like different categories [of block statements]. And when you pick a category, more logic blocks appear, then you can build like in Scratch."*

No decisions were reached in regards to the programming language for the relationship editor, but discussions showed the general ideas in implementing a non-text-based language in the relationship editor.

4.2 First Iteration Development

The future workshop's dream- and implementation phase described in Section 4.1.3 and 4.1.4 yielded ideas to continue into the first iteration design. The ideas concretized in the

first implementation phase provided an actual suggested implementation. There were also several smaller suggested improvements in the dream phase which were never sketched.

The Controlling the Home interface was designed for the purpose of creating a graph interface. The choice of adding connections and relationship functionality was made in order to give the prototype more functionality, allowing us to test the graph concepts of the interface. The future workshop provided me with several suggestions to improve the design of all aspects of relationships, that is both creating, visualizing and using them.

The first design iteration would be a low-fidelity paper prototype, for the purpose of testing the flow and the design of static elements in the application [20]. The design of this low-fidelity prototype is detailed in this section. This development and design section has some elements which are not as relevant to the research topics found in Section 1.3, but which would be part of bringing the interface's look forward based on suggestions from the future workshop. An example of this is changes made to room nodes. These elements will be discussed as future work in Chapter 7.2.

4.2.1 Room nodes

The new room nodes would use a reverse-fish-eye effect of an image of the room. I used the reverse fish eye effect in order to show more of the room in a small image.

Figure 4.8 shows two choices for the room node design.



Figure 4.8: Left: reverse fish eye effect. Right: No distortion to the image.

The glare was added to the image in order for the distortion to feel more in place, this change was based on strike team feedback where both drafts shown in Figure 4.8 were presented. The strike team was composed by a peer who co-authored Controlling the Home.

The room node circle would then receive a glare or a smaller part cut-out from it to illustrate the total light level in the room. Whether this information came from the lights themselves and their status, or a light sensor remains to be decided. Being a low-fidelity prototype, this would not have an animation when the light level of the room changes. The children of room nodes, the device nodes, would also be changed.

4.2.2 Relationship Arrows

During the idea phase of the future workshop, the participants provided several ideas concerning relationships. The dream phase of the future workshop was detailed in Section 4.1.3.

The users suggested that the arrow indicating a relationship between two devices would show additional information about the relationship, for example if there are any timers attached, and which "direction" the relationship has.

This idea would not be implemented into the first iteration prototype. Instead, I wanted to test the current version of the arrows, which only provide the user with the knowledge that there is a relationship between two devices. The relationship arrow would also serve as an entry point to the relationship editor, selecting the relationship itself. This will be detailed in the section below.

4.2.3 Relationship Editor

When creating the relationship editor, I looked towards the solution produced during the implementation phase of the future workshop. There were two solutions created, a "simple" and an "advanced" editor. As the advanced editor had text input, I chose to pursue the simple version, where the user would use pre-defined logic statements and code blocks to form relationships. It would also continue the direct manipulation interaction style from the graph interface. The simple editor produced during the future workshop is depicted in Figure 4.9.

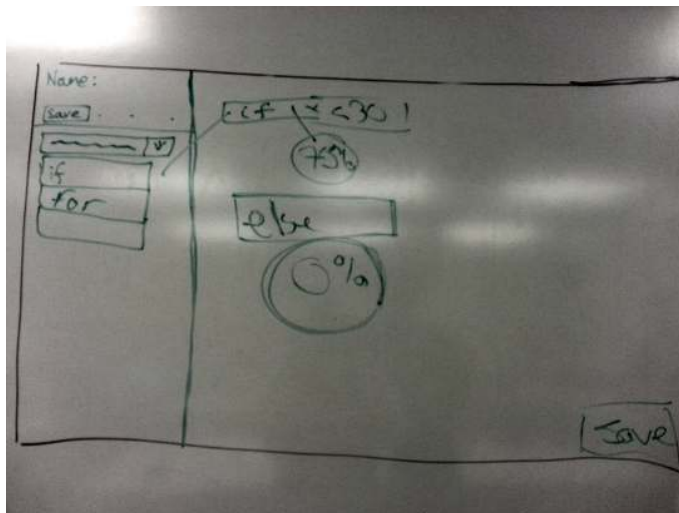


Figure 4.9: The "simple" editor drafted during the future workshop. Here users can use pre-defined structure blocks to create logic between devices.

When designing the relationship editor for the low-fidelity prototype, I tried to match the existing design of the prototype from Controlling the Home, as well as the new room and devices nodes which will be detailed in Section 4.2.6.

The functionality for creating relationships was visualized above in Figure 4.9. The workshop left some details out of the design, and I needed to complete the interface based on the feedback, and my own thoughts. To aid me in this, I would use a strike team.

The strike team was used to gather quick feedback during a five minute session. The team consisted of three previous testers, and one user with no previous affiliation with the project. The work which they were to critique is illustrated in Figure 4.10.

The following list contains the strike team's feedback:

- Divide the left relationship creation interface vertically, not horizontally.

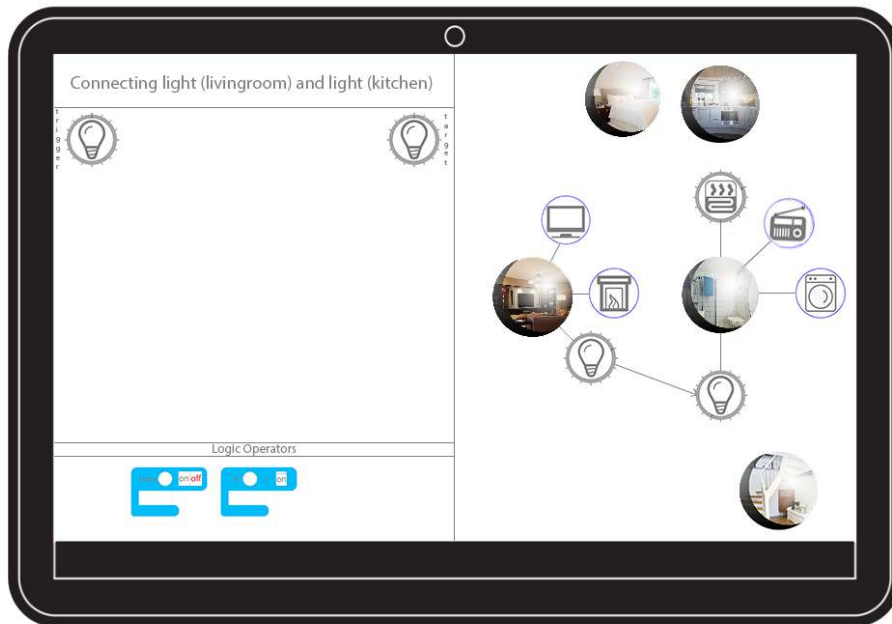


Figure 4.10: An image of the relationship editor during development.

- Flat design? iOS uses flat design, the room nodes are conflicting with the design.
- Pull entire room nodes over to add all devices in the room.
- The interface is not intuitive enough.
- A member of the strike team expressed that the current version could lead to an information overload, and that there were too many small details. Instead, he suggested that the relationship creation functionality should be an overlay across the entire screen, which would be prompted to appear in the same way as the current off-canvas-layout by dragging a device on top of another. He believed this would emphasize the importance of this functionality to the user, in addition to believing this is not something users would do commonly, but rather selectively "now and then".
- The text describing devices should be underneath, similar to how people are used to icons in for instance Microsoft Windows.
- When interacting with nodes in the relationship editor, the graph view should update, showing which device you have selected, and updating the relationship arrow to show relevant information for the relationship. This would aid concerns which were raised during testing of the Controlling the Home prototype, where one user expected the relationship to work "the other way" where the trigger device, and target device, were swapped.
- Arrange the rooms as a hierarchy (later the user altered his opinion when he got a better understanding of the interface, seeing how nodes are already hierarchically

structured.)

There were noticeable differences in the feedback between the testers who had previous knowledge of the project, and the one who did not. At the end of the session however, and having been briefly introduced to the interface, he too understood the concept.

There were mixed feelings towards using horizontal or vertical lines in the relationship editor. I will keep the horizontal lines. If the expert evaluation shows that this is not the way to go, it can be altered. For now, the design will attempt to match the result from the future workshop's implementation phase.

This strike team feedback altogether was not as fruitful as I had hoped. The majority of the feedback were related to high-fidelity implementations or feedback which was directly contradicting what had been discussed in the future workshop. There was one suggestion which stood out however.

The suggestion of completely separating the relationship editor from the graph interface was particularly interesting. If the integration of the relationship editor in the interface received harsh criticism during the following expert evaluation, this solution was something to consider. For now, I believed it deviates too far from the design proposed at the future workshop and resources to make such an alteration to the design were unavailable to me.

A part of the interface which I did not get any feedback on, and which was not detailed in the future workshop is what I called the relationship palette. The top section of the relationship editor would hold device nodes which could be used to create the relationship. The palette was populated by dragging device nodes from the graph interface. If the relationship editor is opened by dragging a device node on top of another, the same way one created relationships in *Controlling the Home*, then the palette was populated by both devices nodes.

The next step was to create the first draft for the programming language to be used inside the relationship editor.

4.2.4 Logic Operators and End User Programming Facilitation

As previously mentioned, and described in Figure 4.9, the user would be able to use simple programming structures to create logic in their home. The easiest example of this is an if-this-then-that structure closely resembling what was observed in existing applications as presented in Section 2.1.

As the users were already presented with current EUP applications such as Scratch, the idea of using scratch based blocks for this programming became prevalent in the future workshop, while never sketched in greater detail than what was shown in Figure 4.6, the suggested design resembled Scratch.

The earliest version of a logic block is illustrated in Figure 4.10. As shown previously, the product of the implementation phase left some room for interpretation. This part of the module was therefore developed in large part alone, utilizing rubber duck debugging and strike teams, these methodologies are detailed in Section 3.2.5.

As work progressed on designing the logic block, it became apparent that it needed slots for more options, particularly a slot for different operators, as well as an 'on' or a 'off' block. These operators were limited to greater and less than, as well as equals. This was in line with the programming language discussion from the future workshop found in Section 4.1.4.

An example of an "if-test" block is illustrated in Figure 4.11(b), and the parts which make up the if-test is shown in Figure 4.11(a).

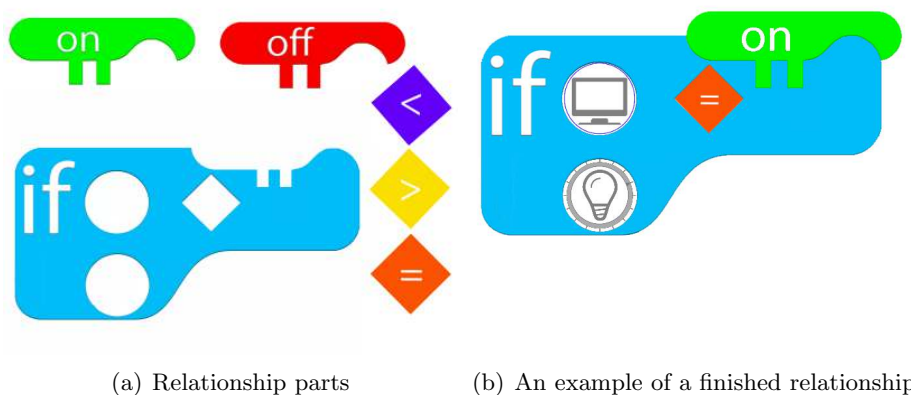


Figure 4.11: The left figure shows the various parts the user can use in order to create a relationship. An example of a finished relationship is shown in the right figure.

This logic block resembling a key would allow for 'on' or 'off' blocks to fit into the slot on top, and would enable device nodes to fit into the open circles. When dragging a device node, the slots should indicate that the node could fit there, by glowing or otherwise standing out from the background and surrounding elements. This would not be implemented in the low-fidelity prototype however.

The if-block illustrated in Figure 4.11(b) is complimented by two blocks. The user can choose between "greater than", "smaller than" or "equal" as their operator, as well as a boolean block indicating "on" or "off". In addition, a percentage block will be included.

The percentage block will let the user choose which strength they want floating value devices such as audio or lights to have.

4.2.5 Menu for Overview of Active Devices and Relationships

Alongside the relationship editor, the users suggested a second major new functionality to the prototype. This would be a list of active devices or relationships, showing their states and providing the user with further overview of their home. I chose to populate the list with relationships configured in the home, both active and inactive ones. This decision was made based on previous work in *Controlling the Home* where the menu was considered, and my overall perception of what would be more useful to the concept.

User 5 suggested that this list could be navigated to by using a swipe gesture from the edge of the tablet, in the same way that Windows 8⁶ display its sidebar menu. Figure 4.12 shows an example of an off-canvas layout.

As the relationship editor appears from the left side of the screen as shown in Figure 4.10, this overview would swipe in from the right. When designing the overview menu, I looked towards the previous product of the future workshop shown in Figure 4.7, with the influence from existing applications such as the Facebook Messenger shown in Figure 4.12.

⁶<http://windows.microsoft.com/nb-no/windows/home>

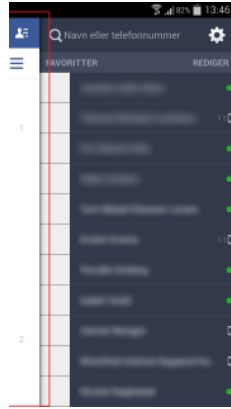
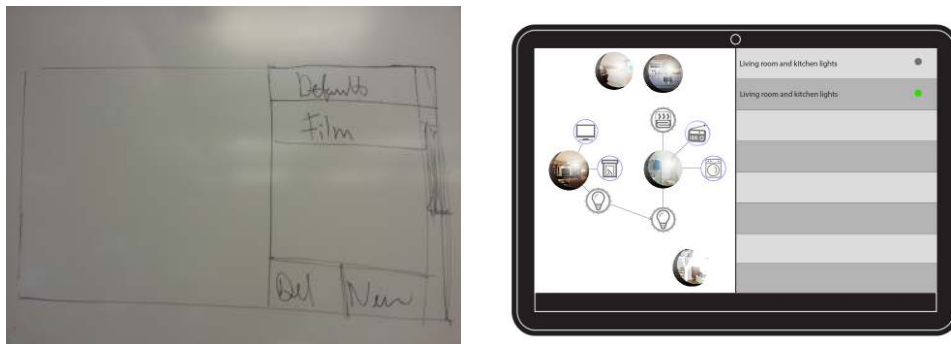


Figure 4.12: An off-canvas layout used in Facebook's Messenger Application. Red: What used to be on top is now slid to the side, making way for the contact list on the right.

The comparison between the existing draft from the future workshop and my design is shown in Figure 4.13.



(a) The relationship overview menu as drafted during the future workshop. (b) My implementation of the relationship overview menu for the first iteration low-fidelity prototype.

Figure 4.13: A comparison between the implementation, and the inspiration for the relationship overview menu.

Copying the design from the future workshop was not as complicated as with the relationship editor. For the overview menu, the future workshop's draft left little room for interpretation. I added a dot at the right of every relationship which would indicate if the relationship was active or not.

4.2.6 Device States, Sensors, and Device Utilities

During the future workshop, replacing the text on device and room nodes in favor of icons and photos was discussed. The device nodes would therefore be replaced by icons representing each different device. The device node icons would be selected by myself. I tried to choose icons which are self-explanatory, otherwise known as "semantic knowledge" [45]. This change is also in line with Google's Android design guidelines which state "pictures

are faster than words”⁷.

As mentioned in Section 4.1.3, most users, even some with previous involvement in the Controlling the Home prototype did not completely understand the meaning of the device’s states in the existing interface. Even after explaining thoroughly, there were several inconsistencies which led to criticism. These inconsistencies are illustrated and highlighted in Figure 4.14.

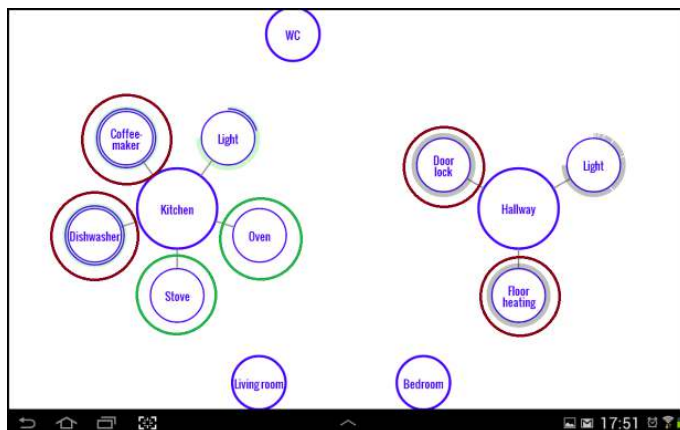


Figure 4.14: Inconsistencies in the Controlling the Home prototype. The red circles are devices which should behave similarly, but the pair in the kitchen node has double blue lines, while the hallway nodes have one blue line and a thick grey outline. The grey outline is intended to be ‘off’, but this was not clear to users. The green pair has only a blue line as they have child nodes of their own similar to rooms, but the users did not understand why they had no border.

Participant 2 who had previously suggested the visualization of light levels in the room, suggested applying the same technique to both heat and light levels of individual devices. The feedback to this was mixed, and user 1 believed this would compromise the interface and be difficult to visualize satisfactory. The most prominent issue which the users agreed on was therefore the problems highlighted in Figure 4.14 previously. Figure 4.15 shows a new addition to the interface based on strike team feedback, the child node indicators. These indicators would be applied to both devices with multiple children or states, and to room nodes as they too have child nodes. This was thought of as a small addition to the interface, which could give the user a sign of what to expect when interacting with the node. The number of circles represents the number of child nodes or utilities.

I chose to begin the changes by altering the floating value device nodes. Figure 4.15 shows a quick white board mock up of a new floating value node made during strike team feedback while design of the first iteration. This was the same session as we discussed room nodes as mentioned previously.

The problem, and the need for a new floating value device node can be traced back to the decision of using a custom gesture dubbed the “power adjustment gesture” in the Controlling the Home prototype. The power adjustment gesture is faster than long pressing, this problem is detailed in the Controlling the Home report [47].

However, if not to redesign from scratch, the design should accommodate this feature, and support it gracefully. Therefore, in order to avoid clutter, I suggest using a “knob” for floating value device nodes. The goal is for the knob to allow users to immediately

⁷<http://developer.android.com/design/get-started/principles.html>

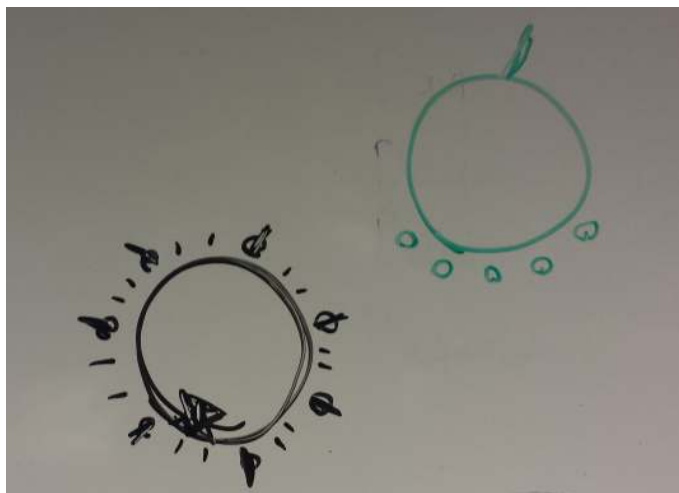


Figure 4.15: Bottom left: The "knob" for dimming. Top right: A device or room node with children illustrated as small circles.

recognize that this button can be rotated, and to do so intuitively.

Devices which have only two states, that is on or off, will be made less inconsistent. As previously discussed, binary device nodes had two different settings for 'on', which was confusing users. Binary device nodes will now instead only have a grey outline if the device is 'off', or two blue outlines if the device is 'on'.

Sensors will be visualized in the same manner as a device node. This was the only suggestion for sensors in terms of visualization during the future workshop.

4.2.7 Developing the Low-Fidelity Prototype

After designing the basic functionality described in the previous section, the relationship editor, the relationship overview menu and the new room and device nodes I began to design the user test.

First, I designed the questions and tasks for the user to complete during the interview. As an initial guideline, I looked towards the tasks we used in the Controlling The Home project. The training tasks covered some of the same areas, and were intended to help the users familiarize themselves with the navigation in the prototype. I would also show the users how the old Controlling the Home prototype navigated in an effort to make the low-fidelity prototype more immersive to the user. The subsequent tasks were tailored to address the new features in the prototype, in order to garner feedback on how EUP was facilitated, the integration of relationships as well as the visualization of sensors and device states.

Then, knowing the tasks, I created images of each screen using Adobe Photoshop⁸. These images were then connected to each other by using the web tool inversionapp⁹. Invisionapp allows for images to be mapped to each other by creating "hotspots" on one image, which when clicked will bring the user to a new screen, with its own new hotspots.

This type of prototype therefore does not support any other gestures than "tap", for this prototype, the lack of "long press" and "drag" would need to be compensated for

⁸<http://www.adobe.com/products/photoshop.html>

⁹www.invisionapp.com

through imagination by the evaluators. The expert evaluation's tasks are enclosed in Appendix B.

4.2.8 Final Thoughts Before User Testing

As the implementation of the design was coming along, simple issues were discovered through an elementary cognitive walkthrough with my supervisor [20]. One such issue was how the relationship menu would open if only one device was active for the relationship. For instance, creating a timer for a single device, would require the device to be dropped on top of a second device to interconnect them, but there was no second device in this example. Furthermore, if the task is to connect two devices which share the same icon, there needs to be something to distinguish between them by.

I decided that moving onwards with testing was more important, and if the situation arose where users struggled opening the relationship editor through initiating a relationship between a device and a timer, then I would intervene and alert them of the problem.

Another, bigger issue also surfaced. The logic "read" poorly, in certain situations as shown in Figure 4.16.

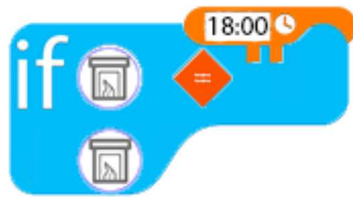


Figure 4.16: "If the fireplace is 6pm?" This statement is confusing, and not intuitive.

I chose not to address these issues prior to the user test, as allowing the users to speak their mind on how to improve it could be more valuable than polishing the prototype and improving issues such as the aforementioned and any consistency issues between screens. The development had also been lengthier than expected, and moving on to the test was desirable.

4.3 First User Test

The first user test would provide feedback to the design originally derived from the future workshop. The first user test will be referred to as an expert evaluation.

The evaluation involved users with a varying degree of domain knowledge and prior involvement with the project. All evaluators were computer science students with relevant knowledge to testing and graph theory and were therefore considered experts.

Their backgrounds are listed in Table 4.2 below:

I invited participants to the test selectively in order to get a diverse pool of knowledge in regards to home automation.

Id	Sex	Age	Knowledge to the Home Automation Domain
1	Male	23	Co-author of Controlling The Home - Large knowledge
2	Female	26	Co-author of Controlling The Home - Large knowledge
3	Male	24	Participant in Future Workshop - Little knowledge
4	Male	29	No prior involvement - No knowledge
5	Female	21	No prior involvement - No Knowledge
6	Female	25	Tester in Controlling the Home project - Some knowledge

Table 4.2: Overview of testers involved in the first user test.

4.3.1 Execution

The users were all students from the Faculty of Computer Sciences at Østfold University College, they were friends and colleagues and had no other incentive for being testers other than helping me, and the free drinks I provided during testing. The test set up is depicted in Figure 4.17. The users also varied in their academic progress, from first year undergraduate students, to master students.

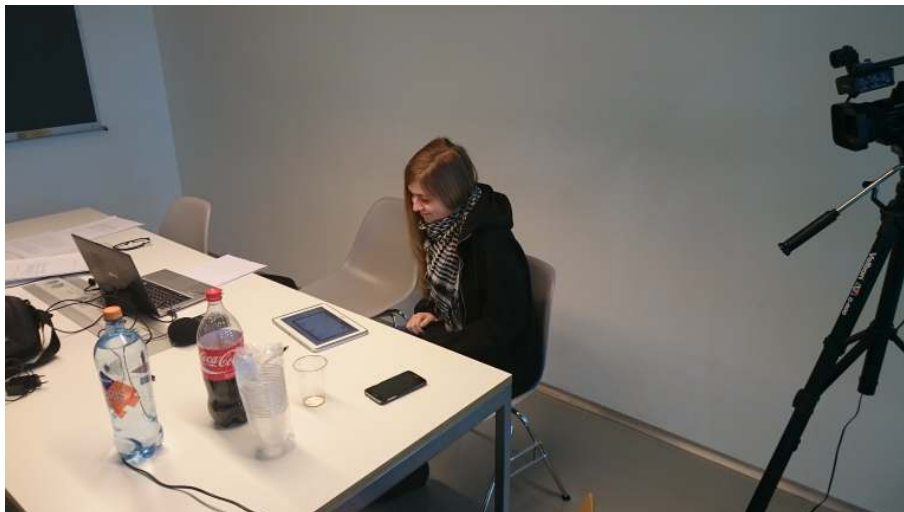


Figure 4.17: The setup for the user test. I would sit next to the tester, in order to aid during training tasks and to observe their interactions with the interface.

After having signed a form of consent, I summarized the content of the evaluation from start to finish. Each user evaluation took between 30 minutes and an hour. This depended on how much the users were utilizing "think aloud", and the length of the interview.

The prototype which we used during the evaluation was a low-fidelity digital prototype. The process and tools used to create this prototype were detailed in Section 4.2. I chose to give the users a quick explanation of all the features in the prototype, which they would need to know in order to solve the tasks. This was necessary due to the prototype being low-fidelity, which meant the user did not get the necessary feedback from the prototype to explore it completely on their own. Some participants with prior involvement in the project did not need as much guidance, as they were more familiar with the interface. The tasks which the users solved are enclosed in Appendix B.

The introduction varied slightly based on the test subject, if they had no prior involvement with home automation, I would explain the background in detail, but for the first two testers who were co-authors of *Controlling the Home* [47], I skipped most of the introduction, beginning the background introduction from where *Controlling the Home* finished.

After having explained the project's background to the users, I introduced the users who had not previously seen the *Controlling the Home* prototype to it in order to aid them in understanding how navigating the graph interface works. This is important, as remaining in control of where you are in a graph interface could be difficult without the animations and feedback to the users as explained by Friere and Rodriguez [16]. As the feedback from making the wrong move is nonexistent when using a low-fidelity prototype, my guidance would therefore help the users understand their actions. While this prototype's interface is similar to that of *Controlling the Home*, I was careful not to let them play too long with the old prototype - only enough to get a feel of the gestures. This was to avoid misunderstandings of which prototype would be used for the test. Some users stated that this made it easier to pretend the low-fidelity digital prototype gave more feedback during the evaluations.

Due to limited resources, while several tasks could be completed in many different ways, the prototype did not accommodate for every possible solution to all tasks, meaning the user could often think 'correctly' but choose a solution which was not eligible in the low-fidelity prototype. In these cases, I would reassure them that they were thinking correctly, but that the interface was not giving them the response which it should. I would then aid them in finding an alternative way to solve their problem, if they did not find it themselves.

The first five training tasks were designed to familiarize the user with the interface. The 6th and final training task would teach the user how to create a relationship between two devices in the prototype. The task was to connect the lights in two rooms to each other so switching one on would also switch on the other.

After the user had completed the six training tasks, I had created six more tasks, specific to relationship functionality. The focus of this user test was to evaluate how well the proposed concept's EUP implementation allowed for the user to create relationships, and how well they were visualized in the interface. In addition, the users were urged to provide feedback on visualizing the states of devices as well as sensors during these tasks.

The user's experience completing these tasks and the previous training tasks would be topics during the interview.

While testing, I urged the testers to think aloud, and to motion which gestures they would use before "tapping" to go to the next image in the prototype. These gestures were the primary reason to introducing them to the *Controlling The Home* prototype before starting the training tasks. I would then conduct semi-structured interviews, and sometimes the users would be so good at thinking aloud, that discussion and feedback started already while solving tasks. If this was the case, I would modify the interview, pointing back at what was previously stated during the test as a reference point for opinions voiced by the tester. If a question had been thoroughly discussed and answered either during testing or earlier in the interview, I would skip the question. After the interview, each tester was handed a Systems Usability Scale questionnaire. This questionnaire is enclosed in Appendix B.

4.3.2 Data Analysis

The test was documented by both recording audio and video. The results I present here are gathered from both the training tasks, scenarios and the following interview. Due to technical difficulties, the last participant's session was only captured half-way through its duration. Being aware of this problem immediately following the interview, I wrote down the biggest issues and ideas from this user, in collaboration with her.

The transcribed data were processed in the same way as the data following the future workshop. This process was described in Section 4.1.1.

The interview technique used was a semi-structured interview. Still, there was a similar pattern in all interviews, and the results to be presented were first sorted by broad topics in lists, then split into more categories and formatted.

If time was no concern, I would have made this analysis with a more thorough version of open coding. However, due to the time constraints of this thesis I believe my methodology extracted the important themes and ideas from the data in an efficient way.

4.4 Results from the User Test

Before the interview, I repeated for each tester that the emphasis of this session was to gather feedback on the concept rather than the prototype itself. From the results and impressions I made during the testing, this message was received by all testers. The results from user testing are structured in accordance to the research subquestions of this thesis.

4.4.1 Relationship Integration in the Graph Interface

The first subquestion in this thesis is:

1: How can a graph based interface for home automation integrate relationships between devices?

The integration of relationship functionality in the interface was done largely through introducing three new concepts to the interface:

1. Relationship Editor
2. Relationship Overview Menu
3. Relationship Arrows

Each of these concepts were tested in the evaluation. These were the biggest additions to the interface, and I gathered more feedback on these elements than the remaining topics. As the amount of data gathered on this topic was large, the positive, negative and feedback sections are further structured by subtopic.

Positive Feedback to the Relationship Integration

Being composed of several new functionalities, the relationship integration gathered a lot of feedback. What was well received in the overview menu, relationship arrows and editor will is detailed separately below.

Relationship Overview Menu The menu received positive feedback from all users, who stated it provide a good overview, and being able to switch relationships 'on' or 'off' was something most testers could envision being a useful tool to have. It was also the main entry point to editing pre-existing relationships, users could also enter the relationship editor menu by pressing the arrow indicating the relationship, but users were split on how this felt. Some thought this was a logical idea, others expected it to have more mass if it was clickable, or that it could easily be misclicked.

Relationship Arrows The relationship arrows were a component of the graph interface. No questions or tasks used or addressed them specifically. Still, some tasks had the option to involve relationship arrows, and therefore they received some feedback. As mentioned above, the relationship editor had three entry points available to the user:

1. Dragging a device on top of another,
2. Selecting a relationship in the relationship editor
3. Pressing the arrow indicating an active relationship between two devices.

This third way of entering the relationship editor, clicking the relationship arrows was criticized by some users. Figure 4.18 shows the application with a relationship arrows present. The arrow goes between two light device nodes.

The users expressed that the arrows were clear in indicating that there was "something" there. After creating a relationship between the fireplace and a timer, user 4 remarked:

User 4: "I can see that an arrow has spawned there, a curve of sorts above the fireplace, meaning something is already set up there. However, it is not obvious that there's a program there."

He was also happy to see that his old code remained when revisiting it by pressing the relationship arrow:

User 4: "I choose to press the arrow, didn't think that was a logical choice though [he was aided by the outline helper in the prototype tool]. Now I'm happy, my previously created code is still there, so I'll just select the timer block and adjust for a new time."

During the interview, the same user had more thoughts on the arrows:

The concern is that if you have too many connection, it can get cluttered and messy. But there aren't that many who are going to have too many connection I don't think.

The user would later have more suggestions on how to improve the relationship arrows as well.

Relationship Editor The relationship menu gathered most of its positive feedback in regards to the Functionality within it, the end user programming. The relationship editor, when opened, would push the graph interface together, making it more compact. All users were able to understand their place in the interface despite the lack of animations in the low-fidelity prototype.

Problems with the Relationship Integration

Introducing both the relationship overview menu and editor to the interface was not without problems. There were several problems identified across multiple evaluators.

Relationship Overview Menu There was no visual cue indicating the presence of an off-canvas layout to the user. Most users commented that this was a definitive error in the design, which needed to be corrected:

User 1: *"I haven't explored that part of the app yet, so without anything in the corner here, I wouldn't be one of those windows users who thinks 'yeah! If you pull from the side here, you'll get a menu'"*

[...]

I would've had something there to indicate that the menu was off-canvas. Or like in Android apps, that 'three stripes symbol' in the corner. If you'd put that in it would be great."

User 2: *"It's difficult to know that there is a menu there, so maybe an animation which indicates it [the presence of the relationship overview menu]."*

Relationship Arrows When asked if they found something unexpected in the prototype's behaviour, the relationship arrows came to mind for one user: "When you suggested I press it, I didn't know what would happen". Two other users said similar things about the flimsiness of the arrows by stating they would need more "mass" in order to look interactable. A third user also thought they were too small to be interactable, and could be easily pressed on accident - leaving the user confused. In summary, the users were split in whether using the relationship arrows for navigation was confusing or not.

Relationship Editor The palette was criticized by all users. There were several large issues with this functionality which the users pointed out during the evaluation.

The relationship palette, populated by two light nodes is highlighted in red in Figure 4.18. This figure also shows a problem which occurs when the user needs to use two devices with the same icon in a relationship; there is no way to distinguish them from one another.

There was no way for the users to know which light corresponded to which device node icon on the palette, the users found this clumsy.

One of the tasks involved creating a relationship between a single device and a timer. This posed a problem. As the regular way of opening the relationship editor was to drag one device node on top of another, this could not be easily accomplished when there was only one device involved. Using the same logic as one would when creating a relationship between two devices, the user would need to drag the device on top of itself. However, this would be ambiguous with "un-doing" the action. While this problem was more tied into the interface's flow and navigation, it also impacted how the palette was used when manually creating a relationship. When dragging one node on top of another to create a relationship, the relationship palette was automatically populated with the nodes involved in the relationship. However, when manually opening the relationship editor, without dragging nodes on top of each other, the palette was not populated by any devices. Users

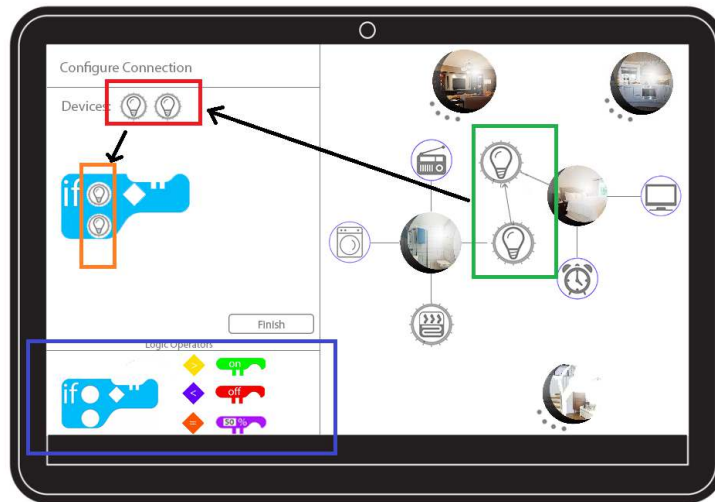


Figure 4.18: Green: The device nodes for lights in two rooms. Dragging one on top of the other folds out the Configure Connection menu. The relationship between them is visualized by a relationship arrow. Red: The device nodes which are possible to be used when defining the relationship. Orange: The device nodes used in the relationship. There is no feedback to the user explaining which icon corresponds to which device as they are identical. Blue: The Toolbox

found this to be confusing. When explained, the users understood the behaviour, but still thought it should be changed.

Suggestions for Improving Relationship Integration

The expert evaluation gathered suggestions to all topics concerning relationship integration. This section is divided into three parts, the relationship overview menu, relationship arrows, and relationship editor.

Overview Menu Figure 4.19(a) shows my design, where the "on/off" button is a green or grey dot. This should be a switch button similar to what users are experienced with from other apps. A switch button is shown in Figure 4.19(b).



(a) My implementation of a toggle. In- (b) The Android switch button. A toggle inspired by the status light in the Facebook between on or off. Messenger app.

Figure 4.19: My switcher in comparison to the default android switcher.

This is a relatively minor suggestion, showing the lack of criticism to the relationship overview menu.

Relationship Arrows Half the users wanted the relationship arrows to do less by not navigating, while the other half suggested they could show more information as well. This information would describe the relationship and potentially provide a better overview to the application. One suggestion was to color the arrow green or red at each end, indicating if the relationship switches a device off or on. An example of how this would work is having the relationship arrow go from red to green if switching off a device would switch another on. While discussing this, user 4 had even more suggestions to compliment this change:

User 4: *"If the relationship involves a timer, we can put a clock next to it. You could press that clock icon and I would expect it to bring up the relationship editor for changing the time!"*

In summary, there were a lot of opinions and suggestions for the relationship arrows. The most common factor between all suggestions was for them to not remain the same.

Relationship Editor As the palette was problematic, the users suggested several improvements and changes both during testing and after testing during the interview. User 1 suggested removing the palette altogether:

User 1: *"The palette you got up here, it probably isn't that necessary seeing how you have to pull from here[the main graph view] anyways."*

Removing the palette either in favor of something else, or because users found it redundant were common suggestions. However, creating relationships between two similar icons was not the only problem with the relationship palette.

User 4 suggested to keep the palette, but to visualize which device belonged to which room: "I would have combined the room symbol with the lamp symbol in some way, to make a artificial identifier, or use text."

Previous suggestions included leaving the icons as they were, but upon being interacted with, their corresponding node in the graph would light up, indicating that they were the same node. User 4 also voiced his opinion on how this would be clumsy, and how it should be possible to get this overview without interacting with the devices, reducing errors.

Furthermore, the proposed design in the interface was therefore to open the relationship editor menu manually, and from there fill in the building blocks into the relationship blocks. User 3 suggested a solution to this problem:

"Well, when you move icons on Android you get this trash can here, amongst other things. Maybe if a clock icon appeared there as well?"

User 3 was not alone in finding this work flow to be awkward. Other users also complained and it was clear that the interactions when creating a timer relationship could be improved.

4.4.2 End User Programming

The second subquestion in this thesis is:

2: *How can a design for a non-text-based programming language for a home automation interface on a mobile device be designed?*

This section is divided into three parts, positive feedback, negative feedback and suggestions.

What the Users Liked

In regards to creating the relationships, all users said that they found it intuitive to drag pieces from the toolbox at the bottom of the relationship editor.

The testers were asked to complete several relationship-centered tasks, where some of them would be the exact same task, but involving different appliances. An example of this is "switching device A on should also switch device B on" - it does not matter which device A and B are, the concept is the same. Most users were able to do tasks which they had done previously or which resembled previous tasks closely near instantly and without error. It was easy to spot when something went completely seamlessly for the user, and I would always ask if this task was easy because it was becoming intuitive, or if they were simply remembering the steps from before and reapplying them again. The answers were split, where some users said they now understood how they should complete the task, while others said they did not think much of it but just applied the same concepts again. None of the testers had any problems doing a task again which they had previously completed.

When interviewing users about their thoughts on the topic of how non tech savvy users would be able to use the system, I led them by suggesting "such as your parents" in order to give a concrete example to work with, asking the interviewees to conjure a character on the spot could have been distracting.

User 3 reflected on this, but swayed away from his parents though, and instead drew parallels to younger non-tech savvy users:

User 3: *"Maybe, I don't know how easily most people will handle if-statements. The 13-15 year olds I taught earlier this week grasped the mindstorm if-blocks really quickly. I don't think my dad would've understood it though."*

[...]

User 3: *"It [Lego Mindstorms¹⁰] is pretty similar to this. You put out an 'if' then there's a drop down menu to select from."*

Most of the testers had little faith in older users being able to use the concept to its full potential or at all.

What the users didn't like

The prototype attempted to simplify creating a relationship by merging "state" and "action" into the same icon. One way of looking at this is to view the device node as an object, which internally has a state. By state, I refer to the current state of the device - if it is powered on, off, or by a floating value such as a dimmed light. By "action" I mean the device's representation of itself in form of an icon in the programming logic. However, the device nodes appeared smaller when placed inside the "if-block" in the relationship, and customizing them was not intuitive to any of the users. I observed that once the user placed a piece inside the if-block, it was not intuitive to edit or configure the device node further.

In the tested prototype, a relationship could read like this: "if the fireplace is 6 pm, then fireplace". This would make more sense if we knew that "then fireplace" also was able to describe the heat level of the fireplace: "then the fireplace is at 60%". This

¹⁰<http://mindstorms.lego.com/>

was not coming across clearly to users however. Improvements would be suggested and implemented to correct these issues, and are described in the next section.

Some users commented that creating the relationships was tedious altogether. They thought there should be a way of solving common problems and relationship structures easier. An example of a common relationship which needed a bit of code to get going was connecting two lights to each other in an open living room and kitchen set-up. This required too many interactions and logic, and should come more easily thought some testers. In the test, the user was asked to connect two lights to each other. The end result would be that switching one on would switch the other on, regardless of which one you switched on. In order to program this, the user needed two if-statements, one for each scenario. However, user 4 thought that not being able to define a relationship within a minute was not a problem at all:

User 4: *"I don't think many are going to have a lot of relationships anyway. Perhaps lights are common.*

[...]

They [relationships] could be a good solution to first world problems [in the home] though, a nice solution to some of those potentially."

Some users raised concerns that non tech savvy users could have difficulties using the interface, specifically creating relationships between devices:

User 5: *"My mother wouldn't even touch this, you know how the older generations are, she would be afraid she would delete everything or crash the system."*

[...]

User 3: *"My mom and dad aren't technical at all. I don't think they would've been able to complete these tasks."*

Creating relationships between devices was the most technical part of the evaluation, the remote control functionality was thought to be usable by less technical users in Controlling the Home [47].

Suggestions to Improving EUP in the Interface

Some users also wanted to define the relationship as "if the clock is *greater than* 8 pm, then fire place...". User 1 reflected on this, and that this could be because of how this translates better into actual program code which he was very familiar with. User 1 was very active in thinking out loud:

User 1: *"Then an 'ant eater' shows up with '8' [User 1 thought the timer block resembled an ant eater]. If the fire place is more than 8? Maybe? Do I have to press this again, yes, then the fireplace is being affected. I wonder, greater than, equals! 'If the time equals 8. "*

[...]

User 1: *"So if the time is 18, then the fireplace is switched on, maybe we need a block to switch it off too. "*

Me: *"The prototype doesn't facilitate that."*

User 1: *"So then you have an action which happens at 18. If the fireplace is under DDOS attack, and it skips eight though? It would probably be fine."*

What user 1 was trying to explain, was that if the logic misses the timestamp at 18:00 sharp, then the fireplace is never switched on. The logical choice would be to say 'the clock is greater than 18:00'.

As a contrary reflection, user 1 found it natural to define the relationship using *equals* if the fireplace switches itself on once, instead of being a continuous action:

Me: *"You would've used 'greater than 6'?"*

User 1: *"Yes, then it would've been continuously switched on, which would've been weird as well."*

No conclusion was reached on which operators made more sense in regards to relationships involving timers.

User 2 wanted to redesign the if-block. She drew a suggestion for a new logic block, which she thought made more sense. This is depicted in Figure 4.20, and would later be implemented, the implemented version can be found in Figure 4.24.

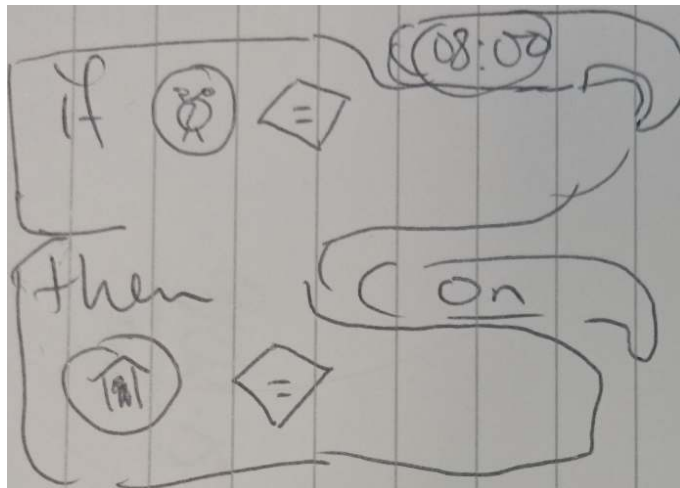


Figure 4.20: User # 2 drew what she thought would be a more logical way of defining relationships. This would later be implemented, the changes and comparison to the old prototype is shown in Figure 4.24. This solution also uses a clock as a device node.

When subsequent testers pointed out the same issues as this new if-block would address, I asked them for possible solutions, and after brainstorming on their own, I would present them with the solution shown in Figure 4.20. The four last users thought this was an improvement over the current design, which would solve some of the issues currently in this prototype.

Some users suggested that a tutorial would be necessary with this software, and that it should describe these common relationships, so that users themselves could learn from this, and start off at a "initial safe spot" where something works. This also hints at the necessity for templates or template functionality. Users thought this was something which could be suited for online sharing.

4.4.3 Sensors and Device Utility

The third subquestion in this thesis is:

3: *How can device utilities and sensors be visualized in a graph based home automation interface?*

This section is divided into three parts, for positive feedback, negative feedback and suggestions in regards to this topic.

Positive Feedback

The "child node indicators" shown in Figure 4.21(a) and 4.21(b) received positive feedback from some users, while other users paid no attention to them and did not comment.



(a) Stove with four child node indicators. (b) Room node with three child node indicators.

Figure 4.21: Child node indicators on both room- and device nodes.

User 4 thought the child node indicators were a good addition:

Me: *"Did you think anything worked well in the interface? If yes, what?"*

User 4: *"It was very straight forward. It was easy to get an overview with the child-elements. They were on the hot plates too, so you immediately saw that you could control each one individually."*

The child node indicators did not receive a lot of feedback, and went unnoticed by several users.

Negative Feedback

One user commented that they did not understand what the child node indicators were. After a short explanation of what they represented, the user responded that they liked them.

Device utilities would receive a well thought out suggestion by user 3, who thought the current implementation of using child nodes as device utilities could be improved. His suggestion is described in the next section.

Suggestions

In the Controlling the Home prototype, a custom gesture dubbed the "power adjustment gesture" was introduced. In this user test, this gesture did not receive neither criticism nor appraise, which is probably attributed to the low-fidelity prototype, which did not allow users to use this gesture. However, user 3 had a detailed concept for further integrating it into the interface. This gesture is not native to any mobile platforms, and was therefore under a lot of debate from the users in Controlling the Home [47].

The way a user performs this gesture is by pressing a node, quickly dragging outwards from the center of the node, and rotating around the node in order to adjust a floating value device node. The gesture is visualized in Figure 4.22.

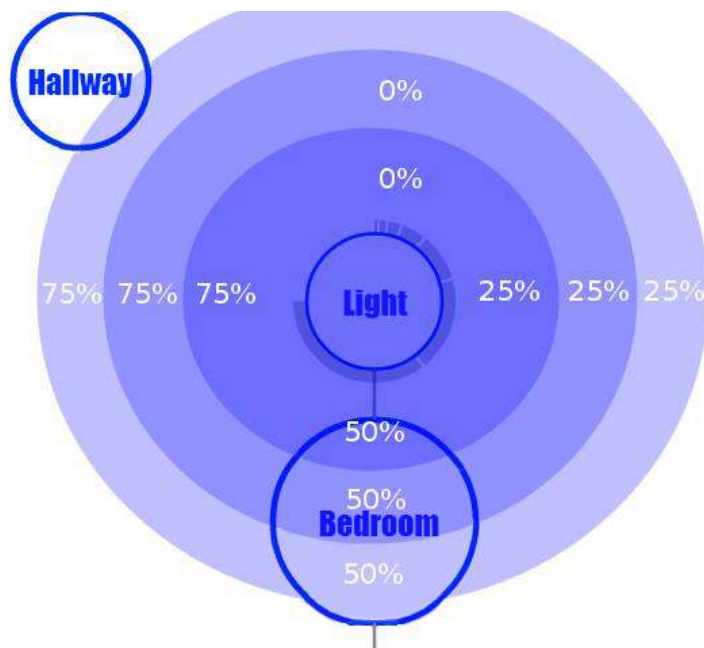


Figure 4.22: A visual representation of the Power Adjustment Gesture. Having briefly pressed light node in the middle, without releasing their finger, run it outwards away from the node pressed. The power of the device will be adjusted based on the finger's angle to the light node.

User 3 suggested that this gesture could be integrated to also work on device with multiple options, or "utilities". Currently, selecting different modes such as dishwasher modes was done by tapping the device, and selecting one of its child nodes representing different utilities. This suggestion could integrate the power adjustment feature into more of the interface, and reduces the need additional child node trees. An example of a device with multiple utilities is a washer, where the user can select different programs based on the fabric they are washing. User 3 used this as an example when pitching his idea:

User 3: "Can it [washing machine] run different settings?"

[...]

User 3: "Like put a wheel here. On a washing machine, you use a wheel to select settings."

This feature would potentially integrate the power adjustment gesture into more parts of the prototype, and allow users to select device utilities quickly. My immediate reaction

to the suggestion of this design was to compare it to video game menus, which the user also could relate to. This interaction style is similar to how some video games implement menu selections, as shown in a screen shot from Dota2 ¹¹ in Figure 4.23.



Figure 4.23: An example of a chat menu from the video game Dota 2. The player will hold down a button to trigger the menu, and steer their mouse or joystick in the direction of the item they want to choose. This is the same concept as the tap-and-drag interaction suggested by a user during testing.

¹¹www.dota2.com

The user thought this would be an accurate and quick way of reducing bloat in the interface, as three levels of child nodes could crowd the interface. This suggestion led me to look for more information on radial interfaces. This theory can be found in Section 2.2.3.

As previously mentioned, an overlay was suggested could solve the interface's issues of opening the relationship editor when creating a relationship with a timer device. User four thought it was a good idea to also use this overlay for sensors.

4.4.4 Device States

The fourth subquestion in this thesis is:

4: *How can the state(s) of devices be visualized in a graph based home automation interface?*

This section is divided into three parts, for positive feedback, negative feedback and suggestions in regards to this topic.

Positive Feedback

Device states were generally understood amongst users. User 4 was very direct in his feedback:

User 4: *"I can clearly see now that it was switched on, as a second blue ring appeared."*

[...]

User 4: *"I envision that if you 'long press' it [a device node] then you can do more advanced things. So I'll just press once to switch it on or off."*

User 4 had correct assumptions when he first approached the interface. All the testers found it intuitive to "tap" the device node to switch the device on or off.

Room nodes were successful in using a picture instead of text. All users would select the correct room every time, and I believe that if the pictures resembled their own home, they would be faster in selecting them as well.

Negative Feedback

While not strictly relevant in terms of device states, room nodes were also commented on. This feedback is useful to take away from this evaluation for the purpose of future work. Some users found the room nodes to be confusing. User 4 said:

User 4: *"The pictures. Even if I can tell that this will be difficult in reality, when a user needs to separate them [room nodes], he would need go and take pictures of his house".*

User 4 liked the idea once it was explained further however. He had forgotten about the built in camera present in most mobile devices today, and imagined the user needed a second device or application to capture images before uploading them to the prototype:

User 4: *"Oh that's right! As soon as this is an Android application, and you can press 'capture image' if you work some more on this, it's a completely different story."*

This kind of feedback shows the importance and benefits of think aloud. In addition, it also triggered me to repeat myself in emphasizing that the concept was being evaluated, not the prototype itself.

As a side note, the users were also quick to spot inconsistencies, and most commented that they found hot plates being binary, instead of floating value odd. This was an inconsistency as other heat sources were floating value nodes.

User 6 found the floating value device nodes to be hard to see:

Me: *"Okay, I want you to reduce the floor heating in the bathroom"*

User 6: *"The floor. How am I supposed to know that this is just the floor, and not a radiator? I'll use the same gesture [power adjustment gesture], pressing it. I can't see any feedback."*

Me: *"If you look really closely, you can see a little black indicator"*

User 6: *"Oh, that was not easy to see... [...] That's just impossible to see."*

There were also suggestions made to improve these issues. These are described in the next section.

Suggestions

Device nodes received both positive and negative feedback. These opinions were both towards the same topic: whether the visualization of devices being powered on or off was understandable and served its purpose.

As previously mentioned, the floating value nodes received criticism by several users, amongst them user 6. Without making a concrete suggestion, her tone implied that the floating value nodes needed to be made significantly more visible.

As a final note for floating value device nodes such as dimmed lights or heating appliances, one user wanted an additional percentage reading on top of the device node, to make it clear which power the device was giving. During testing they were visualized as a 'knob', with no text. This suggestion closely resembles strike team feedback about device nodes, which was presented in Section 4.2.3. This feature was also included in Controlling the Home, but was left out of the low-fidelity prototype tested now, as testing gestures and feedback on low-fidelity prototypes was not the goal of this evaluation.

4.4.5 Questionnaire results

The questionnaire used was a System Usability Scale (SUS). I emphasized that they were evaluating the concept, and not the low-fidelity prototype. The results averaged 76.6. The full scores are visible in Table 4.3 below.

We can use the curve in Figure 3.1 to see that our average of 76.6 translates into a B-

Question	User 1	User 2	User 3	User 4	User 5	User 6	Average
1	3	4	3	4	4	3	3.5
2	3	3	2	3	3	3	2.33
3	1	3	2	3	3	3	2.5
4	2	4	4	3	4	4	3.5
5	1	3	4	4	3	3	3
6	2	3	4	3	4	4	3.33
7	3	2	3	2	4	3	2.83
8	2	4	4	3	4	4	3.5
9	2	3	2	3	3	2	2.5
10	2	2	4	3	4	4	3.16
Score	52.5	77.5	80	77.5	90	82.5	76.6

Table 4.3: The data from the expert evaluation's questionnaire.

4.5 Considerations and Other Observations

The user test provided me with valuable data to move into the next design iteration. This section is split in two, first other observations which did not fit under a research question specifically, and then considerations and observations regarding the evaluators.

4.5.1 Other Observations

An interesting opinion which many users shared, was how the learning curve of the software was not too off-putting. The users explained that if they had already gone through the trouble of buying the necessary back end devices to make this system work, figuring out the interface would be "the least of their worries" one user said. They also thought they would be very motivated to do so as well having invested time and money into the hardware and infrastructure already.

This is not to say that usability is not a concern at all, it is and always has been. But it is an interesting consideration, that the current target audience of home automation will be less intimidated than possible future generations where houses and apartments are built with home automation in mind. Brush et al. interviewed smart home owners in 2011, and shed some light on many different concerns in home automation [7] and found support for this, where lower cost in money translates into greater cost in time commitment.

The platform used for creating the low-fidelity prototype, *invisionApp*¹², provides the user with a green "outline" if they do not press anywhere which leads to a new picture. This green outline highlights the areas which have been predetermined to send the user to a new picture. Some users commented that this helped them understand where to press. I observed that many users learned this behavior and would use it when stuck. When this occurred, I tried to aid them and ask them why they pressed where they did, and how they felt about the "intended" choice.

¹²www.invisionapp.com

4.5.2 The Users

Amongst the users involved in the test, those who had come further along with their education had participated in more user tests, and had an easier time directing their feedback at the important topics. This became very clear when transcribing the results. During testing, it did not feel like the novice students were considerable "less valuable" to me than the more experienced testers who did not need to be as reminded to talk aloud or to elaborate their answers. When looking back at the results however, novice users talked less overall, which resulted in less data.

As all testers were computer science students, they were considered "experts", but some experts had more experience than others. Two of the testers were my co-authors on *Controlling The Home*, and they were able to provide different insight compared to the other testers. While marginally, I believe that them having a sense of ownership to the prototype made them more positive to it.

4.6 Second Iteration

In this section, I will look at the steps taken in order to move from the first iteration concept, to the second iteration concept. This includes considerations in order to accommodate new methods of testing during the final evaluation, such as altered prototype formats.

While the prototype needs to be tailored around the test, the test can be made to accommodate the prototype as well. Certain changes to ensure that the test yields valuable results can be made both to the concept, and to the test itself.

This section describes the process of creating and testing a concept for visualizing sensors, device utilities, states while allowing the user to create custom behavior through elements from end user programming.

4.6.1 Developing the Prototype

Following the advice from my adviser and a look at the resources at hand - I will implement a part of the interface on the Android platform as a high-fidelity prototype. During the first user test, the part which the evaluators struggled the most with while "envisioning" in the low-fidelity prototype were the off-canvas layout as well as the rule editor. Therefore, only the rule editor will be implemented as a high-fidelity prototype. The off-canvas layout was less complex, and due resource constraints, would not be implemented. The off-canvas relationship overview menu introduced fewer new elements and had a significantly lower complexity compared to the relationship editor.

The graph interface has not changed much since Controlling the Home, and will not be interactable either. This causes some small issues as the second iteration's relationship editor will be tied closer to the graph view. As a compromise, certain device nodes will be interactable.

4.6.2 Designing the Second Iteration Prototype

The changes to the first iteration's design were based on the feedback gathered from the first user test. Through think aloud and interview sessions, the users provided possible solutions to all problems and aspects of the prototype which were criticized. These solutions would be the premise for the alterations to the design proposed for the second iteration. There would also be some last minute changes done as part of strike team feedback which is detailed in Section 4.6.5

Integrating Relationships in the Graph Interface

Integrating relationships in the interface was done through the introduction of the relationship editor, arrows and overview menu.

Relationship Editor The relationship palette will be removed based on the feedback and criticism it received. Instead, users will drag device nodes directly from the graph interface instead of being able to store them in the palette.

Relationship Arrows The relationship arrows were intended as being a big part of the graph based interface. The edges between nodes is a common element in graph based

interfaces such as tube maps [49], and felt like a natural addition. In the tested prototype, they served as an entry point to the editing the relationship. The users found this clumsy and non intuitive. There were many other entry points to this functionality, and the users who did use the relationship arrow to navigate to the relationship editor later said that while it was logical to use the arrow, using the relationship overview menu felt more natural.

Relationship Overview Menu The relationship overview menu was well received, where the only complain was the on/off icon, which was also suggested changed to a android on/off switch as shown in Figure 4.19(b). It also needed a small icon to indicate that it was an off-canvas layout, as there was no indication that the menu existed to the users. These changes did not amount to any major changes in the work flow of the prototype. Depending on how the users solved tasks in the previous user test, between 3 and 4 tasks utilized the relationship overview menu. It was tested thoroughly. Therefore, it would not be updated for the next iteration, and not included in the test.

End User Programming in the Interface

As previously mentioned, this solution is very similar to Scratch, which has been tried and tested [37] [36]. In order to make the relationships more understandable to users, device nodes used for creating a relationship would no longer serve both as an action and as a representation of itself. Only the latter would be the role of a device node in a relationship. The action portion of the device node in the first iteration was confusing to users, is now defined through other logic-blocks. This concept is shown in Figure 4.24.

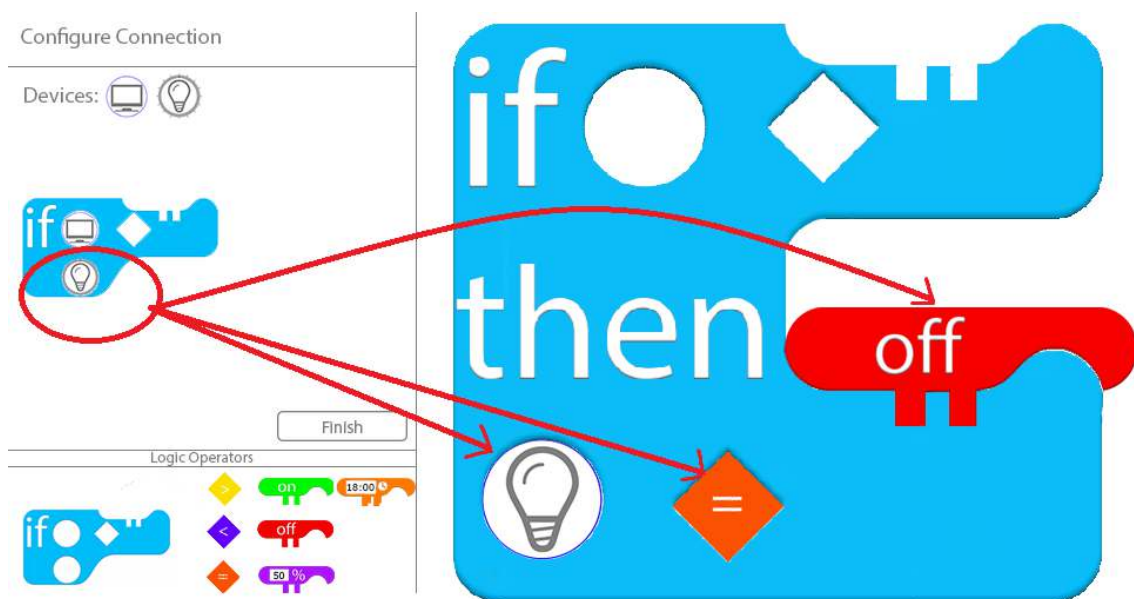


Figure 4.24: Left: The old relationship block used a device node to serve both as "action" and as a representation of itself in the programming logic. Right: The red arrows show how each purpose of the device node has been redistributed in order to make it easier to use, and more intuitive. The new relationship block (right) is not fit to scale.

Having to fill in two additional slots in the if-block is slower than the previous iteration.

But if users never understood the alternative at all, then a minor speed bump is a small trade off for being able to use the application. As Figure 4.24 shows, while providing the same functionality, the relationship block has been considerably expanded.

Since we first started brainstorming ideas in *Controlling the Home*, the concept of incorporating a timer to relationships has been discussed. While the previous iteration contained an idea for this, users said it was not intuitive. A new suggestion arose, as shown in Figure 4.20. The clock device node shown in this figure will be incorporated into the interface as a device node not attached to any room nodes in the graph. It was suggested during the expert evaluation to appear as a contextual menu when dragging a device node, but the implementation would take more resources, and would be utilized for this functionality only. The final evaluation would include a task utilizing this functionality.

Sensors and Device Utilities

While the users gave little mention to the topics of sensors and device utilities during the future workshop, the expert evaluation shed some light on them. During the future workshop, only sensors were mentioned, and the conclusion was to treat sensors similarly to device nodes. Therefore, for this test I would include a task where the user needs to utilize a movement sensor.

As presented in Section 4.4 an alternate design for device utilities was proposed. The suggestion was to allow the user to select device utilities by using the power adjustment gesture. This feature is similar to that of an actual knob, which you would use to select the settings or utilities of the washer. While potentially an elegant solution, there was not enough time to develop and test this functionality.

None of the users noticed or paid any attention to the light illustration on the room node. The room node's light visualization component was explained in Section 4.2.1. I will leave this functionality as is for this prototype, but include an interview question about it in order to evaluate the functionality.

Device States

The state of a device was mentioned briefly by some users, where they wanted floating value devices such as lights and heat elements to display their power in percent. This would make it difficult for the user to misinterpret the floating value state of the device.

The visualization of the state of a device was very similar in the previous iteration, to that from *Controlling the Home*. Floating value nodes were changed, a white board mock up for the new floating value node is shown in Figure 4.15.

The binary nodes however, only received a minor update by making the blue outline slightly thicker. As previously detailed in Section 4.4, most users were able to understand binary nodes. I therefore only changed them slightly.

In summary, I propose a new iteration of design to all types of device nodes. Floating value, binary, and device nodes with children as shown in Figure 4.25.

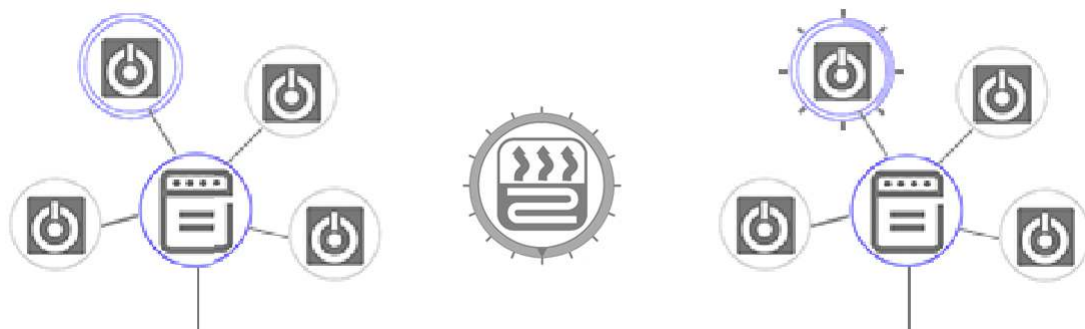


Figure 4.25: Left: Stove during the first iteration, mistakenly a binary device node. Middle: The first iteration's floating value node. Right: The new device node powered on, now also changed to a floating value device node.

Device nodes now show a single grey border if they are off, or a double blue border if they are on. To keep the design consistent, devices with children such as a several hot plates on a stove will show the stove as blue if there are child nodes which are on. If all the child nodes of a device node with children such as the hotplates in Figure 4.25 are powered off, the main device node's outline will also be grey.

4.6.3 Limitations of the High-Fidelity Prototype

The high-fidelity prototype will allow for different testing compared to the low-fidelity prototype used in previous testing. The high-fidelity prototype is developed on the Android platform, and facilitates most of the interactions necessary for completing the tasks in the user test in the same manner one would expect from a finished product.

When developing the prototype, some simplifications were made in order to allow it to be finished in a timely manner.

Users can not select the time or percentage on timer and percent blocks. They are fixed to values which will be correct for the test scenarios.

When dragging pieces, the prototype looks for a drag gesture around the top left corner of each block or device node. Figure 4.26 shows the zones where the users can interact press to select blocks.

Because the device nodes found in the graph view and the blocks found in the toolbox are in fixed positions, their hit box can be more precise. I will explain this behavior to the users so they will not feel like the prototype is unresponsive.

The high-fidelity prototype does not allow for the users to interact with the relationship overview menu. Therefore, it is not included in any of the tasks which the users will complete. The relationship overview menu uses only "tap" gestures. The low-fidelity prototype used previously was able to demonstrate the interaction with this menu well, and the users had no suggestions to the menu which would alter the flow of the application. It is therefore not implemented in the high-fidelity prototype.

The prototype allows for relationships to be created inside the relationship editor. Everything outside of the relationship editor is a static background picture. This means the device nodes can not be switched on or off, and the room nodes can not be interacted with at all. Users can still interact with the device nodes shown on screen in order to pull them into the if-blocks they define in the relationship editor. This will spawn a new clone

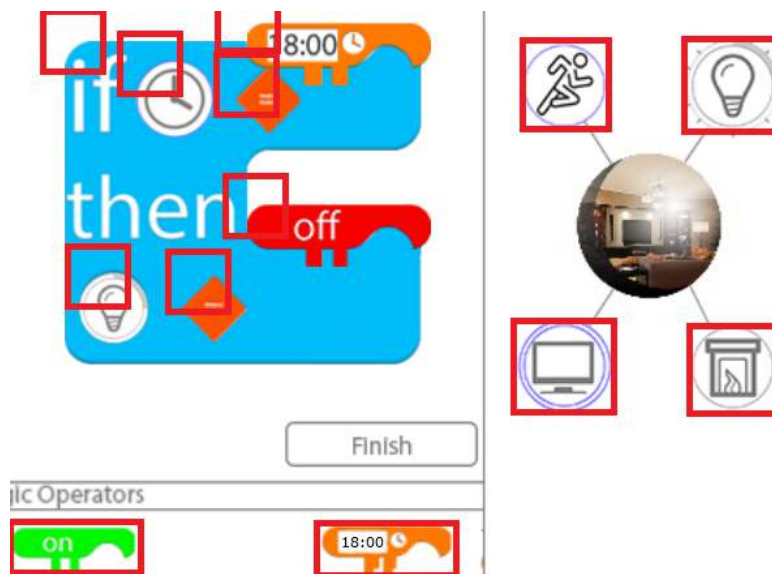


Figure 4.26: An excerpt of the application view. The red circles show where the user can press in order to "select" the adjacent node or device block. Presses not within the red zones are ignored.

of the device node icon, as many times as the user wants to.

Once a user is finished creating a relationship in the editor, the prototype will not create a relationship arrow between the devices in the graph interface.

A final limitation involves a feature discussed during the future workshop. The user should not be able to make mistakes. While the prototype does allow for some control on what the user inputs, by checking if the correct blocks fit into its socket in the if-block, it does not check if the syntax makes sense. With more time, the prototype should discourage users from defining logic such as "if the clock is less than 50%" which is currently legal syntax, but which makes no sense. Maloney et al. shares their thoughts on how to approach error handling in user scripts in Scratch: "A program that runs, even if it is not correct, feels closer to working that doesn't run or compile at all." Without a connection to a working back end solution with actual devices, this will be difficult to measure, and users will only have my guidance for what works and what does not.

4.6.4 Changes to the Low-Fidelity Prototype

The first three training tasks in the final evaluation will use a second low-fidelity prototype. The base of this prototype will be the low-fidelity prototype used during the previous user test. In the previous user test, the low fidelity prototype was used for all the tasks, as mentioned above it will only be used for the first three training tasks and will therefore be less comprehensive for the final evaluation.

In addition to the device node changes detailed in Section 4.6.2 and Figure 4.24 above, the prototype also received a small change by adding symbols to symbolize the presence of the relationship editor and overview menu.

Some inconsistencies between slides were also fixed, namely icons jumping unexpectedly should now stay still when not interacted with. Further, some room nodes had more child node indicators than actual devices. A clock device node was also added. This addition, as well as the icons shown in top left and right in Figure 4.27 will not be used

by users interacting with the low-fidelity prototype, but were added in order to keep the low-fidelity prototype consistent with the high-fidelity prototype which users will be using after interacting with the low-fidelity prototype. Adding the off-canvas menu indicators were also a common suggestion during the expert evaluation, and required little resources to implement.

4.6.5 Strike Team Feedback

Before running a mock test with a peer to ensure the test was well thought out to give me the answers I wanted, and that its estimated duration was correct, I had a small strike team feedback session for the prototype. The purpose of this session was to identify any small bugs which I had not discovered during testing and development. The perceived difficulty of altering or fixing these bugs was the main factor in deciding which feedback to listen to.

The strike team provided the following suggestions:

1. Add more dummy hot-spots to the low-fidelity prototype, so testers do not learn that green spots means correct spots.
2. Select logic blocks based on color. Current implementation looks in a square around a given coordinate, as shown in Figure 4.26. The strike team member suggested using the color of the pixel to determine which block was pressed.
3. Swap the touch-spot for logic blocks. Figure 4.26 shows the old touch spots. The new touch spots would be in the middle of each block, except for the if-block which remained in the top left due to the size of the block.
4. Reset the relationship editor when pressing the finish button.
5. Redesign the if-block. One of the strike team members thought the "then clause" read wrong, and would make more sense to keep more as a straight line, similar to how the "if clause" is structured. This would later be suggested again and is visualized in Figure 6.7 as a proposed third iteration if block in Chapter 6.
6. As the strike team was using the prototype, one user quickly had fun with dragging countless logic blocks into the relationship editor. A trash can functionality was suggested, where the user can drag a piece on top of the trashcan to remove it.

In the end, I implemented suggestions 1, 3, 4 and 6. Adding more hot spots was a small implementation, and would clearly decrease the degree of which users were able to learn that the green square represented hot spots. Figure 4.27 shows this behavior after the user has pressed somewhere where there is no action mapped to.

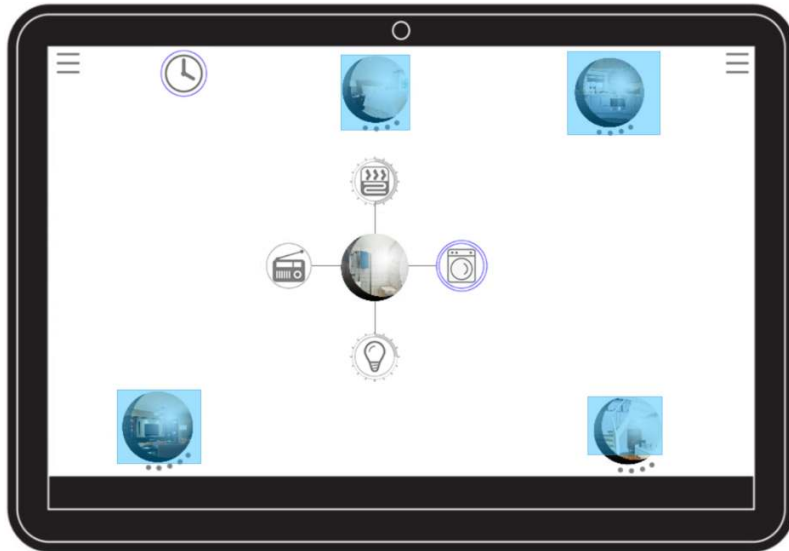


Figure 4.27: An example of what happens if the user presses somewhere in the inversionApp prototype tool if there is no picture mapped to that area.

In the previous version, before the strike test feedback, there was only one square mapped the next image in the low-fidelity prototype. Therefore, if they pressed wrong, you could learn that the square meant that they should press there instead. The new dummy hot spots links to the same image which the user is currently viewing, making the correct hot spot the only way to go forward in the task. If the software had the option to switch these visual cues off, this alteration would not be necessary.

Suggestion 3 made using the touch interface feel more natural. Using the top left corner as a center for touch was a technical simplification early in development, but was easy to offset, making the new touch spots for each logic block the middle of the piece, bar the if block, as suggested. The new touch zones are shown in Figure 4.28. This can be compared to the old touch zones shown in Figure 4.26.

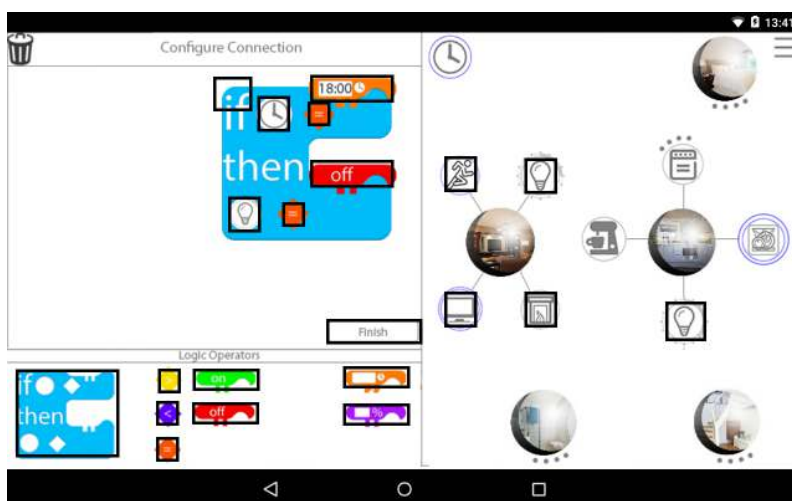


Figure 4.28: The new touch zones for the prototype.

Suggestion four would make the test easier to conduct. The functionality was already implemented for debug purposes during development, and only had to be mapped to the button. Figure 4.28 also shows that the finish button is now mapped to touch.

Suggestion 6 was the largest part to implement, as there was no such functionality already present. The final implementation of the suggestion also allows the user to drag entire if-blocks into the trash can, which will also remove the logic blocks attached to the if block. The trashcan is not marked in Figure 4.28, as tapping or interacting with it will do nothing unless you're dragging either a device node or logic block. The trashcan is located in the top left corner.

Chapter 5

Results

”Everything not saved will be lost”

Nintendo save screen

This chapter is divided into two parts. In the first part, I will look at the details surrounding the evaluations, then how the data were analyzed in order to get to the results which will be presented in the second part of this chapter. In addition to this, solutions and suggestions to the research questions were brought up in earlier testing, which users in the final evaluation did not discuss as much. Therefore, these results from earlier testing will also be presented here.

5.1 Preface and Analysis

This section details the surroundings and execution of the final evaluation. I will present the factors which possibly impact the results, and the choices I made with regards to them.

5.1.1 Considerations and Design

Testing the concept on users who are not computer science students has been suggested since the second iteration of Controlling the Home. During planning of the final evaluation, the opportunity to test the prototype at The Gathering 2015 ¹ presented itself. This would be an chance to reach young adults with an interest in technology, the primary demographic for new home automation systems [7]. The Gathering is a LAN party, where the primary demographic is between 16-25 [17]. Figure 5.1 shows a graph of the demographic data.

The changes to the concept were minor in many areas going from the first to second iteration. The functionality of both prototypes used during the final evaluation is described in Section 3.3.

Therefore, I chose to keep the format of this evaluation similar to the previous expert evaluation described in Section 4.3. This will make the results easier to compare, should it be useful to do so. Some changes needed to be made to the test though, as the first user test was 30-60 minutes long, which I thought was too much to ask from people who did not come to the arrangement with expectations of participating in user tests.

¹www.gathering.org

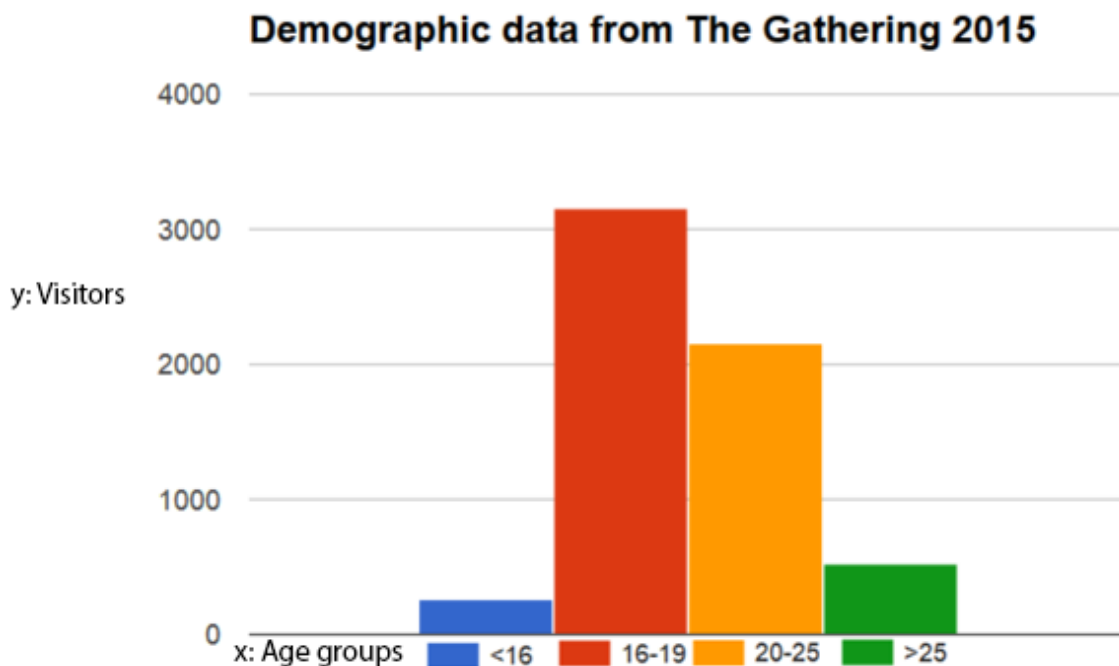


Figure 5.1: The age demographic data from The Gathering 2015.

The first step in reducing the duration of the test was to make the introduction shorter. I would briefly introduce myself, the concept, collect basic personal information before I hand them the tablet and let them start the first training task. The training tasks served more as an introduction and tutorial now than during the first user test.

This left no room for the Controlling the Home prototype, which I did not use to showcase the graph interface as I did in the first user test. For this evaluation, bringing a second application, only to tell the users that they should extract certain elements from this interface, but ignore the rest, could be confusing. The result was introductions between two and three minutes, down from the eight to ten minute introductions in the expert evaluations.

The previous user test consisted of six training tasks, and six6 more scenarios. While I want to keep the final evaluation close to the first user test in structure, I had to remove some tasks in order to reduce the duration of the evaluation. The second user test therefore consisted of four training tasks, and three scenarios. These tasks were designed to test the aspects discussed previously: End user programming and creating relationships between devices as well as sensors, device- utilities and states. The test was conducted with two prototypes - a low and a high-fidelity one, the users had to interact with the low-fidelity prototype until they needed to use the high-fidelity prototype. I aided them during testing, especially during the training tasks.

Finally, the interview consisted of ten questions, where some questions were very broad, and other were narrowed down to one piece of functionality. While this was an increase compared to the previous test, some of the questions were short and concise and did not increase the duration of the evaluation by much. Due to the use of a semi-structured

interview - some interviews were as short as six questions if the evaluator elaborated a lot on some of the broader questions. In the end, I observed that the evaluation durations were between 22 and 28 minutes, which was in line with what I found reasonable to ask from the evaluators.

5.1.2 The Users

The requirements for being a evaluator was to have little to no programming experience. I asked friends at the event if they knew anyone who matched the criteria, and if I could speak to them. The result was 8 testers, four of whom I knew personally, and 4 whom I did not. Two friends I rarely see also participated. Table 5.1 shows an overview of the relevant information regarding the users. My thoughts and reflections on the userbase and its impact on the results are found in Section 7.3.3.

Alias	Sex	Age	Relevant Background	Personal Relationship
Leopold	Male	24	Hardware retailer	Friend
Grant	Male	21	Electrician, familiar with HA	Friend
Melinda	Female	19	None	Acquaintance
Barbara	Female	22	None	Acquaintance
Skye	Female	20	1 programming course in high school	Friend of friend
Calvin	Male	25	Basic HTML	Stranger
Jemma	Female	20	1 programming course in high school	Friend of friend
Phil	Male	20	None	Stranger

Table 5.1: Overview of testers involved in the first user test. The table is ordered based on the order of evaluators testing.

Each evaluator was positive throughout the test, even when the environment got loud and other interviews were conducted around us.

5.1.3 Setting

The evaluations were conducted in a press room in Vikingskipet. The location had a large couch, and was quiet in comparison to the main event hall. The location is depicted in Figure 5.2 below.

The room was also used by other people such as press and crew. This created some elements of distraction, but also allowed the location to be a recruitment source for evaluators as it helped me get in touch with more people.

5.1.4 Execution Summary

In order to facilitate comparison between results from this evaluation and the previous user test, most aspects of the test was unchanged compared to the previous user test. The tasks were modified as detailed in Section 5.1.1. Each test was documented using both a video camera with a microphone, as well as a backup audio recorder device.

For the first three tasks, a low-fidelity prototype was used. This prototype was very similar to that used during previous user testing, but had some alterations to the design. These changes are listed in Section 4.6.4 and minor changes after strike team feedback in Section 4.6.5.

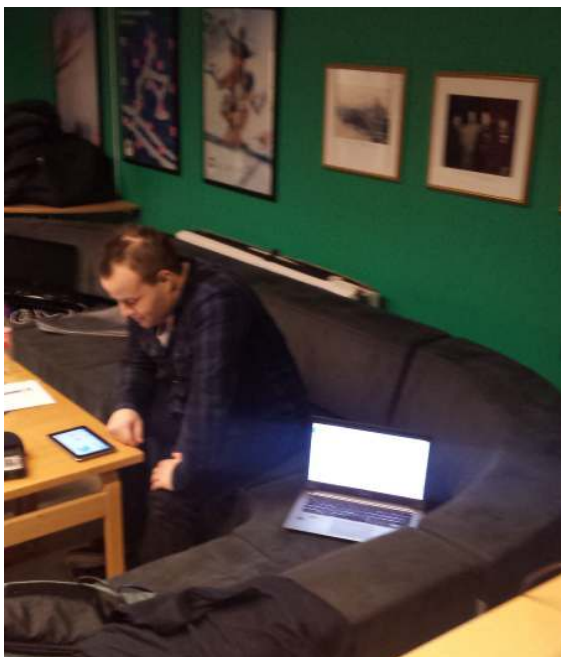


Figure 5.2: The setup for the final evaluation. During the test, a camera was filming over the shoulder of the evaluator to capture the test tablet’s screen.

After the evaluators had completed the first three tasks, I quickly helped them navigate to the high-fidelity prototype also installed on the test device tablet. The process of creating this prototype was detailed in Chapter 4, and the high-fidelity prototype’s functionality was summarized in Section 3.3. In order to make the transition as easy as possible, the high-fidelity prototype closely resembled how the low-fidelity prototype looked after completing the first three tasks. Each task in the evaluation attempted to represent a realistic user scenario with the prototype.

After the tasks were completed, I conducted a semi-structured interview similar to the previous user test. Finally, each user completed a System Usability Scale questionnaire. The System Usability Scale questionnaire is detailed in Section 3.2.6, and the semi-structured interview is described in Section 3.2.3.

5.1.5 Data Analysis

The data to be processed consisted of both audio and video recordings. When transcribing the data, I used the separate audio recordings mainly, as they had a higher sound quality which was valuable given the environmental noise. If the user was describing something which I could not understand without video, I would use the video to aid me as it captured the test tablet’s screen. The video and audio recordings were started close to each other, making it easy to navigate between the two.

During transcription of the data, I used color to highlight interesting conversations. I had used this method before when analyzing the data from the future workshop and expert evaluation. The transcripts were then analyzed using open coding. The initial 88 tags are enclosed in Appendix C.1.

After the first pass of open coding, I pruned the tags. This process is called *Selective*

Coding. The tags were selected based on their frequency, their relevance to the research topics, and how much they would help me in presenting the data in a comprehensible way in this chapter. This methodology is described in greater detail in Section 3.2.4. The resulting tags from selective coding are presented in Table 5.2 below. It is important to note that the tags left after selective coding were not the only ones relevant to the text, but were what I deemed based on my knowledge of the data to be the most relevant and helpful when presenting the data.

(1) Main Topics	(2) Sub topics	(3) Other	(4) Descriptions
Graph Interface	Delay	Inaccurate Touch	Intuitive
Relationships	Target Audience	Learning Curve	Understandable
Then Clause	Clock	Tutorial	Non intuitive
Toolbox	Timer	Pronouncing	Quick
EUP	Device Node	Logic	Reflected
Sensors	Small Icons	Real World	Feedback
States		Exploration	Inaccurate
Utilities			Suggestion
Jigsaw			

Table 5.2: Tags selected through selective coding.

These were topics which would help create the base for the results presented in the next section. Columns 1 and 2 of Table 5.2 contains themes which were commonly brought up during the evaluation. I selected tags for column 1 and 2 during the selective coding process based on their frequency, relevance, as well as my perception of their presence in fruitful discussions. The main topics in column 1 generally had more data associated to the tags than the tags found in column 2.

Column 3 contains other common data which are frequently found in combination with tags from column 1 and 2. These tags usually concern smaller topics which are not the main focus of this thesis by themselves, but which were relevant when combined with tags from column 1 and 2. An example is how I will not be presenting findings regarding "learning curve" by themselves, but rather in the context of and combined with for instance "graph interface" or "EUP".

Column 4 is an example of descriptive tags which helped me group the data correctly when presenting the results. In the same way column 3's tags compliments the tags in column 1 and 2, column 4's tags act as a final filter, describing the type of feedback for the topic. An example of this would be combining a tag from column 1, 3 and 4 as such: "relationships, tutorial, suggestion".

5.2 Results from User Evaluation and Testing

This section details the findings from user testing. Rarely did all users either like or dislike something in plenum, I will therefore refer to "most" users if only one or two users had a different opinion. I observed that the evaluators provided more feedback to what they did not like, than compliment what they did like. If something worked, they found it hard to comment on. This effect has been documented by Firman and Edvardson [15] and Robinson and Ber [43]. This is detailed in Section 3.2.2.

Some users were more reflected than others in their answers. Reflected feedback was generally more fruitful than short comments, this is why "reflected" is a tag left after selective coding. As previously described in Section 3.2.3, Barriball and While lists "willing to reflect" as one of the important qualities in interviewees for semi-structured interviews [35].

Due to the nature of the semi-structured interview, some topics were brought up by some users in interviews, which was not discussed with other users. Another consideration is that if one user suggests functionality or a potential fix to a problem, I could later discuss this with another later user if they too had the same problems or similar suggestions. This means that the first users get less of these kinds of discussions, and the last users' ideas are not discussed with other users.

This section is divided into the core topics emerging from selected coding, where certain categories which are closely related have been merged into one subsection.

5.2.1 Feedback towards the Graph Interface and its Relationship Integration

This section details feedback relating the the first research subquestion:

How can a graph based interface for home automation integrate relationships between devices?

The graph interface is the core of the interface. The prototype's interface facilitates creating logic relationships between devices, and is integrated as a slide-out menu. Figure 5.3 shows the transition which occurs in the interface when the relationship editor is opened.

The questionnaire also presented results relevant to this topic, the questionnaire results will be detailed in Section 5.2.5 below. The tags which were most relevant when compiling the findings in this section were: "Graph Interface, toolbox, small icons, understandable, inaccurate".

What the Users Liked

The users needed to swap from the low-fidelity prototype to the high-fidelity prototype during the training tasks. When making the change, I made a comment describing how the high fidelity prototype had the same view of the graph, but now with the relationship editor open. The users were quick to see the resemblance in the interface, and none felt confused by the addition of the editor, and the now compressed graph view.

The first three tasks, which the users completed by using the low-fidelity prototype were understood quickly and completed by all users with no problems. Pressing nodes and navigating seemed logical to most users, and Skye pointed out how she was used to exploring and that tapping icons to do more was a common interaction which she found fun and intuitive.

Skye liked the way the interface compressed itself, as the functionality was not implemented in the prototype, I had to draw parallels to known software such as the Facebook Messenger application which I drew inspiration from:

Me: *"It's supposed to be like the messenger app on mobile, it slides out"*

Skye: *"Oh that I can agree with."*

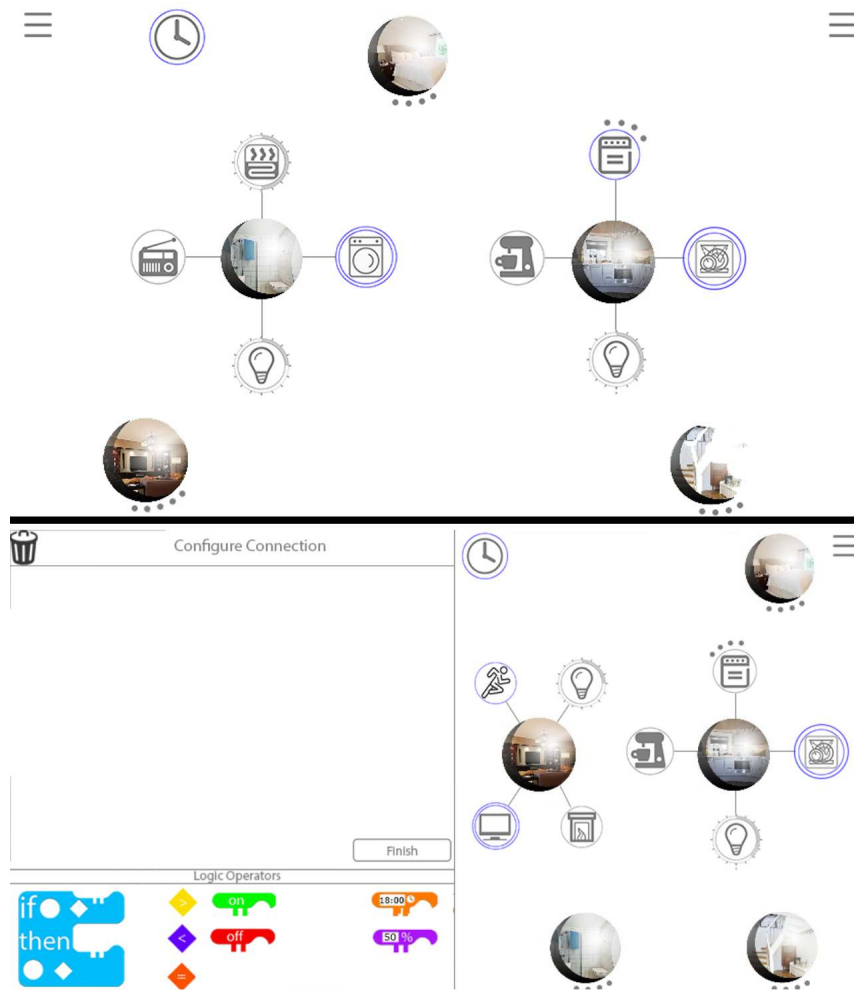


Figure 5.3: The transition screens between low and high-fidelity prototype. The transition is to open the relationship editor, and to open a different room node. Top: relationship editor closed. Bottom: first screen of the high-fidelity prototype, where the relationship editor is open.

She also had some concerns about this topic, which are detailed in the next section below.

What the Users did not Like

A large portion of the changes made to integrating the relationship editor was to remove the "relationship palette" when going into this iteration. The palette, or functionality similar to it, was not suggested by any user. However, one user commented on not being able to tell the difference between different device icons once they were dragged into the relationship editor.

Half of the users found the icons to be too small in the toolbox, which was making selecting the correct blocks difficult.

Skye: *"I think that, perhaps, it was a little small. The buttons, I have quite small hands, but on some formats I can see this being too small. [...]*
I think space is an issue, and it could be a bigger problem for people with larger hands and who's hand-eye coordination isn't as good as us [youth]. What if, this here [graph view] was pressed further together, then you'd have more space on the screen. [...]
That's something which needs testing I guess. Try different sizes, right now it's [the relationship editor] a little small. You can only fit two pieces [if blocks]. And if you drop a piece on top of another, it's 'gone'."

In addition to Skye, other users also had problems either seeing what they were pressing, or pressing accurately enough. The next quotes are from separate interviews, all concerning small icons or inaccurate touch:

Calvin: *"I would try to hit one thing, but end up hitting another."*
 Barbara: *"Once I realized what it said, you see my eyesight isn't that great, then it was easy to solve the tasks. [...]*
I thought the icons for the logic part was too small [..]
The operators were too close to each other."

I also observed that Phil and Leopold had problems with accuracy and selection as well, but did not bring any attention to the problem or comment it.

Suggestions

Going back to the concern of ambiguous device nodes, Leopold commented on revisiting previously defined relationships. An example of this problem is shown previously in Figure 5.4 where 8 device nodes share the same icon in a relationship configuration. He had suggestions to clarifying which device the device node icons in the relationship editor corresponded to:

Leopold: *"I can't tell the difference between the icons."*
 Me: *"No, you can't, there's no difference between them. Do you have a suggestion for some way of distinguishing between them?"*
 Leopold: *"Yes, because as it is, I can't tell the difference between these, if I was to revisit this and look at it later, I wouldn't know which lightbulb is in*

which room. [...] Maybe you could have a different icon 'on' them, depending on which room they come from?"

Me: *"Such as using the room node as an indicator?"*

Leopold: *"Yes!"*

This concern of revisiting old code, or relationships was a new consideration. Currently the way of getting around this problem of forgetting which device icon represents which device in the home was by replacing them with new symbols from the graph interface to be sure. When discussing this with Leopold, he expressed that he found this clumsy.

Grant also discussed the ability to edit and manage previously created relationships, though the conversation took a different turn and we instead discussed the relationship menu and relationship arrows. The prototype tested did not create relationship arrows as a technical limitation. The relationship overview menu was not implemented either:

Grant: *"So it [relationship overview menu] would be able to deactivate the logic then. [...] So you could easily configure what is created here?"*

Me: *"Further, if you connect for instance the light and tv together here, an arrow would appear here [between them]."*

Grant: *"Yeah, but then that could get really cluttering, and make it difficult to see what is connected to what. For instance, the movement sensor to the fireplace, TV to light, a lot between the rooms, then you end up with a lot of arrows, it can quickly become more of a burden than actually helping."*

Me: *"Do you have any suggestions to improving it? You could look at it in a different way, and say that it helps you see that the fireplace is already connected to the timer, and that it shouldn't be connected the light as well. Serving as a warning about the connections, of sorts."*

Grant: *"Or what if they just show the arrows if the connection is active? If something is 'off' or 'on' as a result of a relationship? Like when the timer activates the fireplace, or maybe you could make small icons around, like a little lamp around the TV, then the TV is affected by the light."*

Me: *"Would you still have them [small icons] only when active, or permanently?"*

Grant: *"Only when they were active."*

Only Skye commented that she could find the graph interface hard to relate to, even with pictures on room nodes:

Skye: *"I thought the picture were really nice, cause then it's obviously easier if you can put your own pictures in on your own. Then you'd know which room is which. Otherwise, I'm not very technical, so this is more of a design note, but I think it would be cool if the homescreen showed a blueprint of the apartment. Then you could have the pictured nodes on top of them, so you'd easier relate."*

The suggestion of using a blueprint or top down view of an apartment or room layout emerged previously in the early phases of Controlling the Home [47]. The idea was dismissed however, but will be discussed further in Chapter 6.

5.2.2 Defining Relationships between Devices

The second research subquestion in this thesis is:

How can a design for a non-text-based programming language for a home automation interface on a mobile device be designed?

Most of the evaluation's tasks were centered around using the prototype to define relationships between devices. Comparing to the earlier prototype, this functionality had received the most changes, and the work flow in when creating relationships is different from earlier iterations.

The users completed 4 tasks for defining relationships. These tasks are enclosed in Appendix C. The four tasks concerning relationships are listed below:

1. Training task: Connect the kitchen light with the living room light. Switching either light on or off should trigger the same effect in the opposite light.
2. Scenario: You've installed a movement sensor in the living room. You wish that if it activates, then the light should switch on. Create this logic by using the relationship editor.
3. Scenario: You arrive from work at 18.00, and you wish to come home to a warm home. Create a relationship for this.
4. Scenario: When you switch the TV off before you go to bed, you also want the light to switch off, because the light switch is so far away from the couch. Create a rule for this.

The first task was grouped together with the other training exercises, and I helped the user if any problems arose. After these initial training tasks, all users performed the remaining tasks much faster, with 6 testers having no problems completing the tasks quickly with no issues, finding the correct solution by themselves. The tags from selective coding which were helpful in compiling these findings were: "EUP, then clause, jigsaw, toolbox, timer, delay, learning curve". The tags from the "Descriptions" column were used in order to categorize the data within this topic, for instance: "non intuitive, quick, correct, suggestion".

What the Users Found Intuitive

All users were quick to grasp the drag-and-drop interface concept. With the exception of one user who misunderstood my poor phrasing when explaining the interactions needed to use the relationship editor, all users were able to immediately start drag-n-dropping logic blocks and device nodes into the editor. Some users drew connections to other activities which were similar to defining relationships in the prototype:

Me: *"Do you think an on-screen tutorial could fill my role as a helper?"*

Calvin: *"I think so, because it was rather simple and intuitive. This is not that much different from Lego Mindstorms, and people understand that right away. Like putting together jigsaw pieces and such."*

[...]

Skye: *"This is like a jigsaw puzzle for adults!" [...] "It feels almost like this is testing my math skills."*

[...]

Leopold: *"Oh the round one fits right in there, easy!"*

The three users above also mentioned that the learning curve for the application was noticeably present, but that it was easy to overcome within the first 10 minutes of using the prototype. The topic of a tutorial which was mentioned in the above conversation will be revisited later in this section. I also asked each user if they found the tasks to be logical and to simulate functionality which they would use the prototype for in their own lives. All users responded that the tasks were logical and probable scenarios in a real world setting.

What the Users Struggled with

The first task for creating relationships was to connect two lights together, so that switching one on or off, would switch the other on or off. With the tools available, users would need four if-blocks to complete this functionality:

1. Switch light B on if light A is switched on.
2. Switch light A on if light B is switched on.
3. Switch light B off if light A is switched off.
4. Switch light A off if light B is switched off.

Figure 5.4 shows an example of what this would look like when implemented the prototype. Figure 5.4 also shows the concern of ambiguous device node icons, as half the icons are different light nodes, with no way of distinguishing between them. The users' thoughts on this will be presented later.

None of the users came to this solution by themselves, instead they were satisfied with switching one light on if the other was on, and one light off if the other was off. After discussing their solution and how they envisioned that it would work based on the programming logic created, most users thought that it would be logical to have 4 if-blocks, in order to create this functionality. Some users still thought that their solution using two if statements, one for 'A off- B off' and one as ' A on- B on' should be sufficient, but did not consider this when later tasks required the relationship to not go both ways in the same manner.

When users were having problems with this task of connecting lights, I often asked them "what are you trying to get to work at this point?" some of the testers who figured it out themselves responded saying "I'm trying to set this light on, if the other is on", while evaluators who needed more help would often respond by saying a broader statement, often quoting the task text: "I'm trying to connect these two lights". In general, I observed that most people would start pronouncing logic statements: "'if' light is - 'on', 'then'..." and that this would help them in solving the tasks and using the prototype.

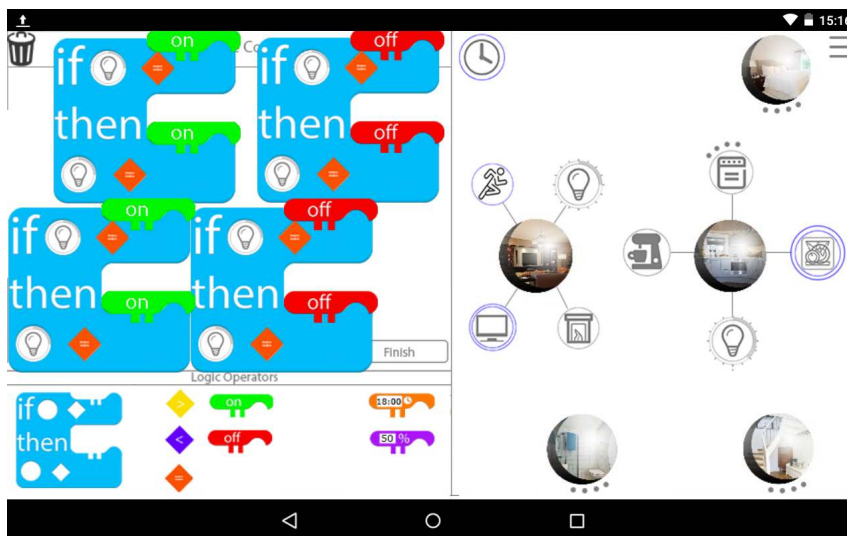


Figure 5.4: The complete solution to training task 3. None of the users completed this, instead they stopped after one "on-on"-if block, and one "off-off" if block.

Suggestions

Melinda pointed out that the operator slot in the "then" clause was redundant and could be replaced by a static element, in the same way the if-block already says "if" and "then". This element is highlighted in Figure 5.5 below.



Figure 5.5: The operator in the 'then clause' is highlighted in red. Melinda identified this as redundant, as all tasks would use the equals operator, and suggested that it should be static instead. A proof of concept for this is found in Figure 6.7 as "third iteration".

Grant also suggested something similar, although he had problems describing his thoughts on the matter. During data analysis it appeared as though he also supported this find, but had troubles putting it into words. None of the other users made this observation, and neither did the strike team prior to testing.

Scenario 2 involved creating a timer:

Scenario: You arrive from work at 18.00, and you wish to come home to a warm living room. Create a relationship for this.

When solving this scenario, Grant recognized that part of this task would be repeated for every timer, and that it could be annoying to have to repeat the timer structure every time. A suggestion to this was to create a new block which would be a timer. Previously, creating a timer was done by using an if-block, then a clock, an operator and a time stamp block. Grant thought this could be simplified by adding a new block which would handle only timer events, a "timer block". This would also eliminate the need for the clock icon, which all users had difficulties finding before I pointed it out to them. The timer-block would be similar to the if-block in size and category in the toolbox.

Grant also suggested a new functionality which he thought would be useful. Grant works as an electrician, and had assisted in installing a home automation system by KNX². He explained how this customer needed help from electrician in order to set up the systems in his own house, how pricey it was, and that a common request was the ability to set a delay. The final task during user testing was a scenario where the user wanted switching the TV off to also dim the lights completely. Grant thought this scenario was not using the technology to its full potential:

Grant: *"Say you want to clean up the living room table, or simply not fall in the dark on your way out of the room. This delay timer could be combined with a condition for the current time, where the user can switch off the TV before 10pm, without switching off the lights afterwards."*

Grant and Melinda were not the only evaluators to suggest new functionality to the relationship blocks. Jemma thought the design of the "if block" as shown in Figure 4.24 was non intuitive. The following conversation takes place while Jemma is having difficulties with the final training task:

Me: *"Do you find it weird having to put the last 'equals block' in there [the 'then' clause]?"*

Jemma: *"No it's just that, I just felt that when the light was under here, then it was more difficult to understand that the 'on button' would work on it, I mean it's right when you consider everything, but it felt like it would be more natural to have it underneath."*

Me: *"How exactly, to be precise?"*

Jemma: *"They should be below, instead of above, I mean it should've been further that way [further away from where the if-statement starts] because the words are in that direction."*

Me: *"You think it doesn't read well? Here you can read 'if the light is on', while here [the then-clause] it says 'if on light equals'?"*

Jemma: *"Yes, because here it's 'light equals on', while here it's 'light equals blank' so maybe the on button should be moved down here [further down alongside and following the device node and operator]."*

This feedback is the same as was gathered from strike team feedback prior to testing as described in Section 4.6.5. In the next chapter, Figure 6.7 will show my concept design for this.

²www.knx.org

Other Solutions, People Think Differently

When tasked with making the fireplace switch on when the user is back from work, Grant wanted to utilize the movement sensor. When I suggested that his solution would not provide a warm fireplace because the fireplace would start when one walked in the room, he instead suggested we used the lights instead, so that walking out of the room would not switch off the fireplace. In this case, he did not see the clock device node, and immediately altered his solution once the clock was discovered.

I also brought up the topic of a tutorial with all the evaluators. These conversations were gathered from the tag "tutorial". Most users were positive of having a small tutorial on screen, suggesting the next step:

Me: *"You mentioned that you got better at this as time went on. How would it have affected your ability to complete these tasks if I wasn't here to help?"*

Phil: *"It would've been slower, but I feel like I would've managed it pretty quickly by myself. Maybe if some tips showed up at the start?"*

Me: *"So swap me out for a tutorial?"*

Phil: *"Yes."*

[...]

Me: *"What do you think about replacing my guidance with a tutorial?"*

Melinda: *"Yes, which would explain where you would put them [logic pieces] in etc."*

[...]

Me: *"Do you think you would've managed the same tasks without my help today?"*

Barbara: *"Yes, but it would've taken more time if I had started from scratch with no tutorial, no guidance in the image, nothing of the storts. Then it would've taken me more time. But I think I would've managed it."*

Me: *"What if you replaced me with a tutorial?"*

Barbara: *"I think that would've been okay. Actually, I think that would've been great."*

Jemma on the other hand, did not feel the need for a tutorial: "I generally don't like tutorials, I like to explore by myself. It wouldn't affect me much."

Leopold was the only user to try to press the empty slots of the if-block. He hoped that this would highlight what he could fit in the empty slot. This is another kind of tutorial compared to the functionality discussed with users above.

5.2.3 Sensors and Device Utilities

The third research subquestion in this thesis is:

How can device utilities and sensors be visualized in a graph based home automation interface?

Data for sensor visualization were gathered from one task during evaluation, where the evaluator was tasked with creating a relationship involving a movement sensor:

Scenario: You've installed a movement sensor in the living room. You wish that if it activates, then the light should switch on. Create this logic by using the relationship editor.

In order to gather more data, interview question 4 enclosed also addressed sensors:

What did you think about the way sensors were included in the interface?

This question was worded very openly, and I would often ask follow up questions in order to get more data if the evaluator replied with a short answer. All interview questions, training tasks and scenarios are enclosed in Appendix C. The selective coding tags I looked towards for gathering data in this section were: "sensor, tutorial, learning curve, non intuitive, reflected".

Device utilities were also involved in a training task:

- Check if the hot plates on the stove are on, if they are, switch them off.

This training task was completed using the low-fidelity prototype.

What the Users Liked

Having lended help quickly to users in the previous task, I wanted to allow them to complete the above mentioned sensor task with little to no help. However, this proved unnecessary as all users were able to complete this task quickly, and correctly.

As the task was completed quickly by all users, most of the feedback for sensor functionality was gathered during the interview. For this iteration of the design, sensors were visualized as a device node which was "on" by using one blue circle, whereas binary device nodes which were "on" would have two blue circles:

Me: *"What did you think about visualizing a sensor in the interface in the same way other device nodes are visualized?"*

Calvin: *"I thought it was logical, but maybe there could've been some small indication that they weren't the same. But at the same time, a sensor is a device, and it's easy to think that "there's a fireplace, there's a light, and this is a TV, I guess the running man is a treadmill." But that could also be fixed by another symbol for the sensor, not necessarily a different indicator."*

While not timed, I would tag the tasks which users completed quickly as such during transcription and open coding. The sensor task was one of the tasks where most users showed that they had a clear idea of what they wanted to do, and executed the task within 10-15 seconds.

The child node indicators, used for visualizing the utilities of devices nodes such as the stove in the prototype received positive feedback by the user once they understood what they meant. The users were split between intuitively understanding the purpose of the child node indicators, and not.

What the Users Did not Like

Two users commented that the icon, as shown in Figure 5.6 below, resembled a treadmill, rather than a movement sensor.



Figure 5.6: The movement sensor icon.

The device utility visualization proposed by a user during the previous expert evaluation became a topic during an interview. An example of this concept is shown in Figure 5.7.



Figure 5.7: A radial menu from the game League of Legends⁴. The player holds a button to show the menu, then drags their mouse towards the direction of the choice they want. The arrow from the center indicates the mouse's position, here going towards the "On My Way" option. This is the same concept as shown previously in Figure 2.9 and Figure 4.23.

I suggested this type of menu for device utilities to Leopold, who did not like this idea. Instead, he preferred the current implementation:

Me: *"A washing machine for instance, has several programs. You have soft fabric, satin, wool and many more. What do you think of using a wheel where you select the washer, then drag out from center in order to select your program?"*

Leopold: *"I would prefer if you could press here, and go into the menu, so you don't have to keep your finger on the screen. Like how the stove works now."*

⁴www.leagueoflegends.com

The topic was not brought up with any other users during interviews.

In summary, sensors were well received by all users. The sensor scenario came after the relationship training task, and all users grasped the concept quickly. The users quickly grasped the device utility visualization, and therefore they did not have a lot of negative feedback to it.

Suggestions

In interviews with the two users who did not like the movement sensor icons, two suggestions emerged. The first was a "sneaking person", and the second was a pair of footprints.

Grant had some ideas regarding the real world applications of this software concept, and the safety around sensors:

Me: What if someone sets up a stupid logic circuit, like switching the fireplace and stove on when the lights are off. That's an extreme example, but do you think that such should give any warnings etc.?

Grant: "Well it's pretty simple, you're pretty skilled if you manage to add errors which switches on the stove, that's well done! 'Power to the people', if people are that stupid, then it's their fault, they would forget to switch off their hot plates anyways [without home automation]."

Me: "You could argue that it would be easier, because you can use the app to see [if devices are off]."

Grant: "Yes, you can just see it there, so you don't have to go into panic mode until you're back from work, or just turn around. I would actually say this is better for fire safety. If you could put fire alarms into this program too, all alarms etc. Maybe not activate or deactivate alarms, but using them as sensors, I mean."

This conversation highlights a few new topics. First, Grant was not concerned over the potential safety concerns of erroneous logic and relationships. Secondly, he identifies that alarms and sensors are similar. An alarm is a binary device which outputs an alarm based on input. This is similar to how the binary sensor device also outputs true or false based on input such as movement or sound. Finally, he recognized the need for integrating alarms into the interface, and suggested that they could be visualized in the same fashion as sensors are.

Barbara did not find the child node indicators to be intuitive, but thought they were useful after I explained how they worked. Calvin had suggestions for alterations to the child node indicators:

Calvin: "I would like to distinguish between the circles for child nodes, and for devices. Maybe use crosses? Or circles without fill color."

None of the other users had any suggestions for alterations to the child node indicators for both room nodes and device nodes. There were other aspects of device node states which received feedback however.

5.2.4 Device States

The last research subquestion in this thesis is:

How can the state(s) of devices be visualized in a graph based home automation interface?

Device state visualization received some changes from the previous iteration. The core issues to shed light on was how well each device node's state was being visualized, including both binary and floating value nodes. Furthermore, one task specifically addressed this. Device states were also a question discussed during all interviews. The tags I used to gather data for this topic were: "states, device node, quick, intuitive, non intuitive".

What the Users Liked

Users found the floating value device nodes to be understandable and convey the device's state well. Furthermore, most users enjoyed and gave positive feedback to switching devices on or off using the interface. The general consensus was that the device nodes were easy to switch on and off, and that with some experience, it was easy to get used to. Most of the evaluators were able to use this functionality without error during the first three training tasks.

Using grey color on device nodes to visualize that they were "off" was well understood by most testers. Most users understood and liked the visualization of "off" devices, but which devices were "on" received some criticism by others.

What the Users did not Like

Leopold thought it was too hard to tell if a device was powered off. He instead wanted a red line across the icon to show that it was "off":

Me: *"You said during the task solving that you didn't like the way devices were visualized as being 'on' or 'off', that you wanted something which was more 'obvious'?"*

Leopold: *"Yes, because if you for instance do a circle with a red line through... But I guess then you wouldn't be able to see the icon quite as well."*

Me: *"I'm liking the brainstorming!"*

Leopold: *"And different colors, I want it to be clearer."*

We then discussed color impairments and how this concern could be a factor when using red and green. We discussed together and found that colorblind mode could be an option, meaning the interface did not have to reserve itself from using red or green.

Leopold was not the only user providing feedback on device node states however. While solving tasks, Barbara did not think it looked like the stove was on at all. She thought a single blue outline was not sufficient in showing that a device was on:

Me: *"Can you check if the stove has any heating elements on, and if it does, switch them off?"*

Barbara: *"Then I would say that there aren't any on, because there is no outline around it [the stove]. Is that wrong?" [...] Because I'd think if it was on, there*

would be outlines. When something is on, or active, it would be indicated in some way."

We continued this discussion during the interview later on:

Me: *"In the first part of the test, the training tasks, you thought the stove was off. Could you elaborate?"*

Barbara: *"It was actually that I didn't think the circle indicated that anything was on. It didn't stand out compared to the circles around it, or those [device nodes] which I had interacted with previously. So there was nothing that indicated to me that if I pressed it, I would get to something which I could switch off. To me, it might as well have been off, and not active at all."*

Barbara also commented that she found the icons to be too small, which she thought was a factor in her initial understanding of the interface before she grew accustomed to it.

Suggestions

Grant commented on the light nodes that the circle around them looked to indicate that they could be dimmed, which was correct. He also thought that some of the device nodes could be an annoyance to have so easily available:

Grant: *"So if I press this washing machine on accident, does it stop, or pause? Because if it stops, that means you have to start over, and that would be really annoying just for a miss click. Same goes for the coffee maker."*

Some users had more of these real life reflections, and it was observable that they thought these scenarios through. Another example of these reflections is Barbara who admitted her apartment was so small there was no need for any home automation at all, as she "could see all her appliances from the couch already".

Leopold commented that certain floating value device nodes needed a unit in order to be useful. Currently, temperature devices such as ovens displayed their power level by using a percentage. Leopold pointed out the necessity for further functionality:

"With percentage, you need to know what 100% is, if I can just select a centigrade, then it's... I mean, you wouldn't have to think about what 50% of 40 degrees is."

In general, all users were very diligent about pointing out inconsistencies in the prototype. Sometimes they were bugs or oversights, but also sometimes larger "details" as the centigrade discussion above is an example of.

5.2.5 Results from the Questionnaire

Following the interview, each participant was asked to complete a Systems Usability Scale questionnaire. The results from each user, as well as the average is shown in Figure 5.8.

From Figure 3.1 we can see that all of the entries receive A or B as their grade, with the average being 82.5, 2.5 points off of an average A grade.

Section 3.2.6 details how to calculate scores from the SUS questionnaire. The score and average score for each question is listed in Table 5.3 below:

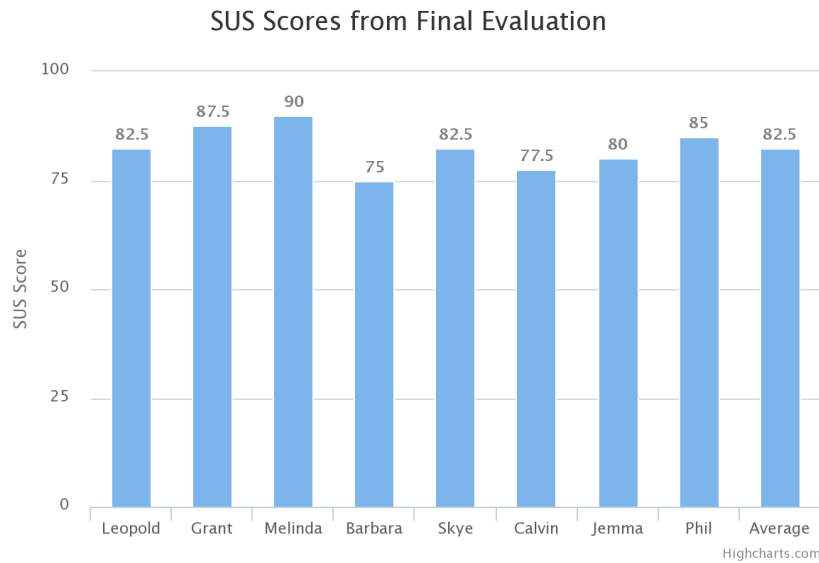


Figure 5.8: Questionnaire results.

	Leopold	Grant	Melinda	Barbara	Skye	Calvin	Jemma	Phil	Average
Q1	4	4	4	2	3	3	3	4	3.375
Q2	3	3	1	4	4	3	3	3	3
Q3	3	4	4	3	4	3	3	3	3.375
Q4	4	3	4	4	3	4	4	4	3.75
Q5	4	3	4	2.5	3	3	3	3	3.1875
Q6	4	3	4	3	3	3	3	4	3.375
Q7	3	4	4	2	4	3	2	2	3
Q8	2	4	4	4	3	3	4	4	3.5
Q9	2	4	3	3	2	3	3	3	2.875
Q10	4	3	4	2.5	4	3	4	4	3.5625
Sum	82.5	87.5	90	75	82.5	77.5	80	85	82.5

Table 5.3: The full scores from the SUS questionnaire used during final evaluation for all users.

Each evaluator filled in their questionnaire after the interview alone, but I would ask if they needed help with any of the terms used. Due to not supervising this process, I did not notice that one user chose to mark between two scores twice in the questionnaire. When calculating the average score, I compensated this by rounding down for one of these cases, and up for the other. In the calculation for question 5 above, I used 3.5 indicating a score between 3 and 4. These forms are enclosed in Appendix C.

Certain questions were more relevant than others in the questionnaire, their results will be discussed in the following chapter.

Chapter 6

Discussion

”I mean, if 10 years from now, when you are doing something quick and dirty, you suddenly visualize that I am looking over your shoulders and say to yourself ‘Dijkstra would not have liked this’, well, that would be enough immortality for me.”

Edsger W. Dijkstra

In this chapter, I will discuss my results by research topic in light of relevant work and research.

Previously, I presented a main research question with four subquestions. The main research question is:

How can relationships between household devices be visualized by using a graph interface for the purpose of home automation?

This chapter is divided into five parts, first discussing each subquestion to the aforementioned research question, and then a summary.

1: *How can a graph based interface for home automation integrate relationships between devices?*

2: *How can a design for a non-text-based programming language for a home automation interface on a mobile device be designed?*

3: *How can device utilities and sensors be visualized in a graph based home automation interface?*

4: *How can the state(s) of devices be visualized in a graph based home automation interface?*

These four questions all impact the main research question. Questions one and two are more closely tied to the main research question however, and therefore have more coverage in both the results and this discussion. Sensors as well as device utilities and states are

all connected to the graph interface, and developing their functionality further impacted the relationship utility.

The first part in providing an answer to the research question was to look at integrating the functionality to create relationships into the graph interface.

6.1 Facilitating Relationship Functionality in a Graph Interface

The first subquestion to the research question is as follows:

How can the functionality to create relationships between devices be integrated into a graph based interface for home automation?

The proposed integration of this functionality involves different modules. These modules were the relationship editor, the relationship overview menu, and the relationship arrows. Each of these modules would all be part of the overall integration of relationships in the graph interface. The relationship relationship arrows and the overview menu were not changed between the expert evaluation and the final evaluation, some of the results discussed here regarding these topics are therefore detailed in Chapter 4, not Chapter 5. My reflections on this testing and data collection is detailed in Section 7.3.

After each module has been discussed, the users' opinions and my own observations on the concept as a whole is discussed in light of other research.

6.1.1 Relationship Editor

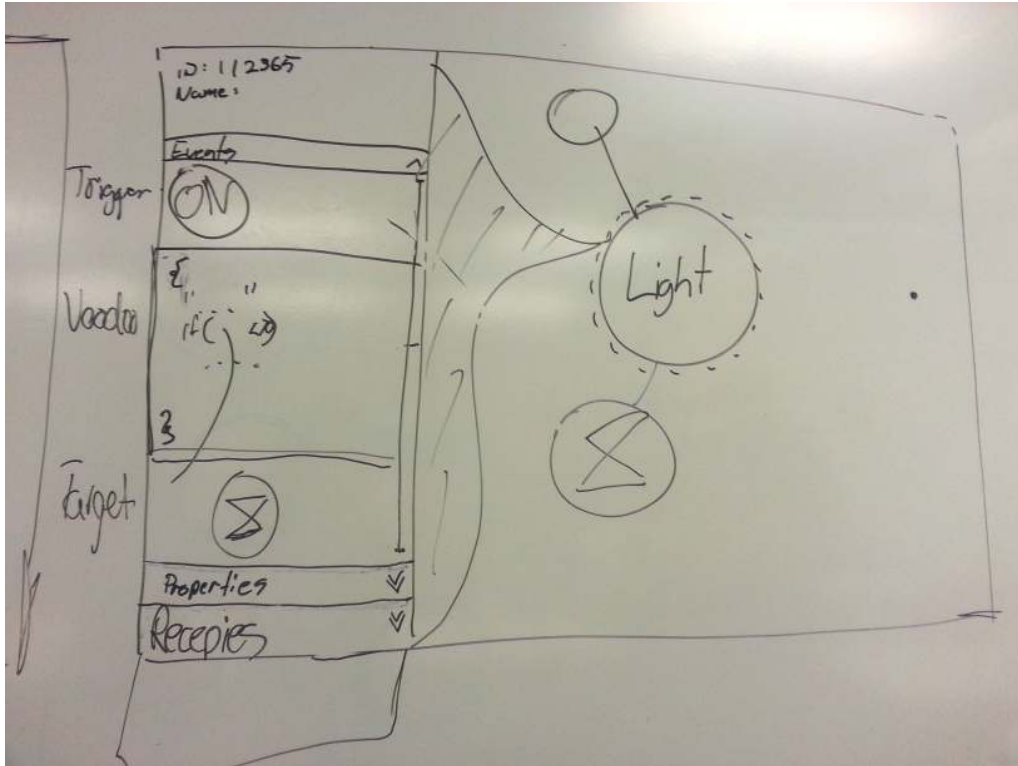
The relationship editor was one of the suggested functionalities emerging from the future workshop. Two interfaces were proposed, these are shown in Figures 6.1(a) and 6.1(b).

After the workshop, I chose to pursue the design in Figure 6.1(b). As shown in the results from the future workshop, the participants did not attempt to pursue a relationship functionality with no programming elements. My choice of continuing with the concept from Figure 6.1(b) was made because I saw more potential in a non-text-based programming solution compared to the advanced editor's text based interface which was described during the workshop as "inspired by Visual Studio"¹. Furthermore, this choice would push the concept in a direction which avoids some of the current challenges of home automation described by Brush et al.: "complex user interfaces, and concerns raised by reliance on consultants". If users can manage the systems themselves, these issues could be less significant.

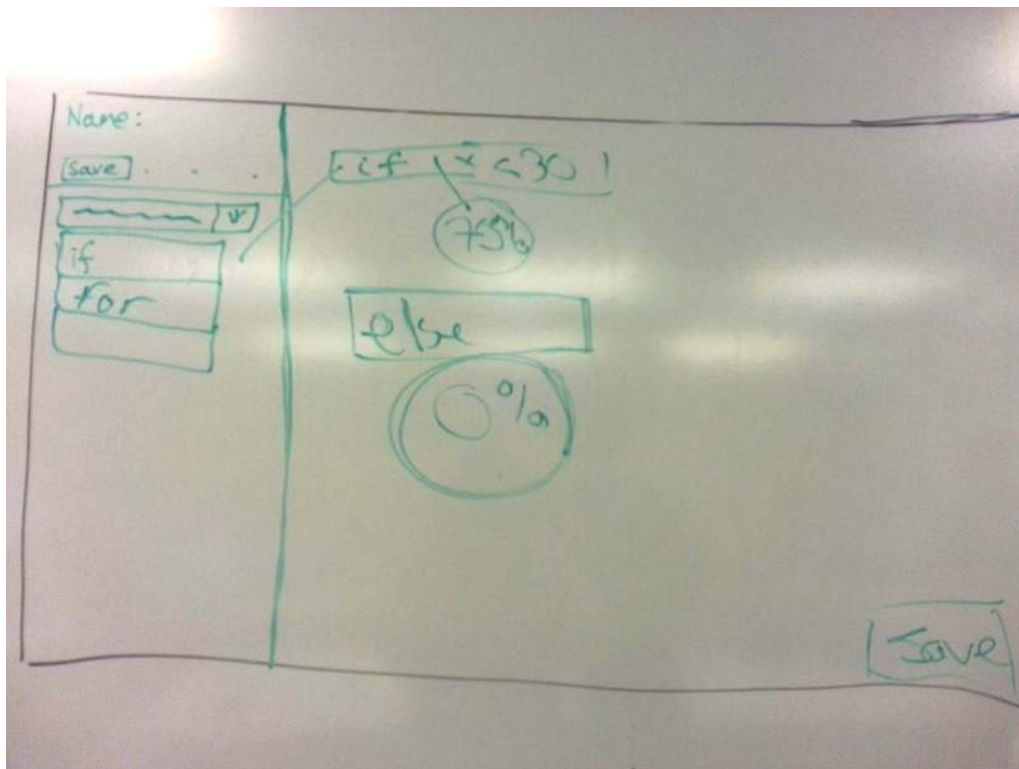
Integrating the interface into the prototype was a large topic during the future workshop, as well as during the expert evaluation. In the future workshop, it was suggested that the interface could slide out, by using a gesture similar to how the Facebook Messenger App works as shown in Figure 6.2.

During the final evaluation, Skye elaborated on the integration of the interface, expressing approval of the concept. However, I do not believe an off-canvas layout to be the best solution. An off-canvas layout is often removed by interacting with or pressing the layer behind it. Pressing the layer behind the menu is a behavior which the prototype relies on in order to select nodes from the graph interface. This functionality is observable,

¹<https://www.visualstudio.com/>



(a) The advanced view editor with text input. The left side of the interface is split into three parts, a trigger device, the code to define the relationship, and the target device. This was inspired by the three-step structure observed in the related work such as HomeBLOX, HomeMaestro and OpenHAB. This structure will be discussed in Section 6.2.1



(b) The simplified relationship editor. Predefined blocks of code are provided for the user to use. This interface would become the design which would be developed further after the future workshop.

Figure 6.1: The two concepts emerging from the future workshop.

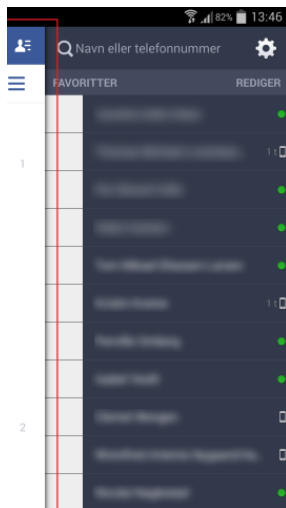


Figure 6.2: An off-canvas layout used in Facebooks Messenger Application. Red: What used to be on top is now slid to the side, making way for the contact list on the right.

and available in frameworks used to create off-canvas layouts². This was not a concern expressed by any of the evaluators, but it is something which could lead to usability concerns if additional functionality is implemented into the high-fidelity prototype. To add to this, the transition and animation of the menu flying out was not implemented into the prototype due to time concerns, instead the user saw the change happen when changing from the low-fidelity prototype to the high-fidelity prototype. This makes it harder to judge, and it is possible that users would have had a different opinion if it was implemented.

The relationship overview menu however, could still work as an off-canvas layout. A simplification done in the prototype, which has not been discussed, was not facilitating the user to select a name for their relationship, or to auto generate a name based on the content of the relationship. This name would be the display name shown in the relationship overview menu.

6.1.2 Relationship Overview Menu

In Controlling the Home, we tried to avoid "cockpit style interfaces" [47]. By cockpit interfaces I refer to several drop down menus, buttons, input fields and lists. An example of this is the MiCasa Verde web interface in Figure 6.3.

²<http://foundation.zurb.com/docs/components/offcanvas.html>



Figure 6.3: An "cockpit style interface". Not scalable, lots of buttons and menus.

In Controlling the Home, the idea of integrating a list interface into the prototype was suggested. The idea was never pursued however, as it could take focus away from the cleaner, more scalable graph interface.

I was able to suggest this list style interface for monitoring active relationships during the future workshop, but while the users were positive, they wanted to discuss the relationship editor more. The workshop taking different directions was to be expected. Apel states this is reliant on the background for the participants [2]. The participants in this workshop were my peers, and I only steered the discussion if it became stale. The relationship overview menu was implemented into the first iteration prototype used for the expert evaluation based on the whiteboard mockup produced in the future workshop, shown in Figure 6.4.

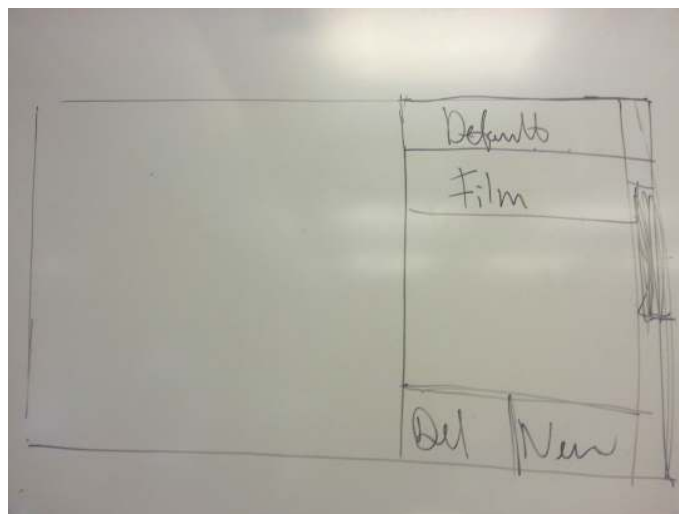


Figure 6.4: Menu showing scenes and active devices. This was thought to be a small addition to the prototype, which could increase the overview of the application. This mockup would become the overview menu shown in Figure 6.5.

The implementation is shown in Figure 6.5

The overview menu received largely positive feedback during the expert evaluation, with only minor design suggestions to take away from the test. The menu was understandable, well integrated, and simple to use. The positive feedback was the reason why this functionality was not tested further during the final evaluation. Seeing how the users

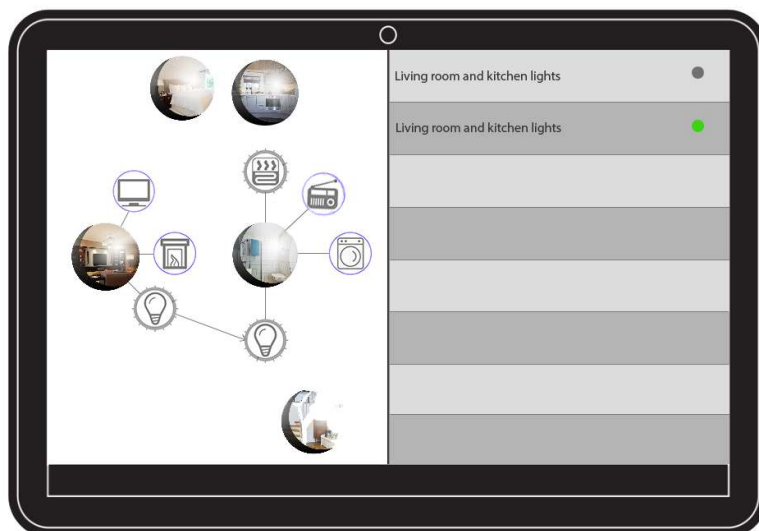


Figure 6.5: The overview menu, an off-canvas menu. Here implemented in the first iteration low-fidelity prototype

in the final evaluation were tackling more difficult concepts such as EUP during the evaluation, I believe this concept would have been understandable for all the evaluators. My perception of what is "more difficult" is based on the amount of errors observed for each task during testing.

The data from the future workshop and expert evaluation were not processed in the same way as the data from the final user test. This could have made an impact on my findings and is reflected upon in Section 7.3.4.

Functionality such as the relationship overview menu and the transition from graph view to relationship editor not being implemented had an impact on the test. It is not a stretch to say that the each questionnaire results would have received different scores if all aspects of the concept were implemented into the high-fidelity prototype. Which parts to implement in the high-fidelity prototype were chosen based on their importance and how costly in time they were to implement. Questions raised during the interview could also have gathered different results if the prototype represented the full concept.

6.1.3 Relationship Arrows

In Controlling the Home, relationships between devices were visualized by an arrow pointing between the devices in the graph interface. I gathered feedback on this topic during the expert evaluation. Some experts thought the relationship arrows tried to do too much, in both acting as an entry point to the relationship editor through the user tapping on the arrow, and visualizing the relationship itself. Other users enjoyed using the arrows as an entry point to the relationship editor however.

None of the applications previously described in Section 2.1 such as the Athom, Micasa Verde, openHAB and Airfy Beacon has functionality resembling the relationship arrows. This functionality was unique to this concept. In the defense of other systems, some of these systems such as the Airfy and openHAB are open to the user, but neither promotes

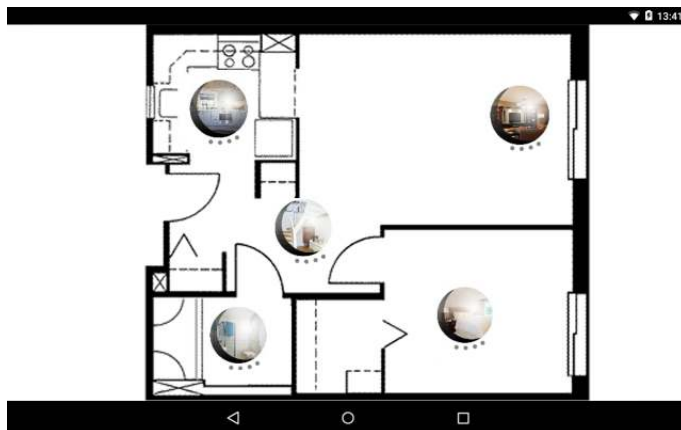


Figure 6.6: The home screen of the application here shows a blueprint of a flat as a background. This could help the interface feel less abstract.

a graph based interface as a solution.

Going into the second iteration and final evaluation, I chose to design without altering the arrows. A contributing factor in this decision was having to implement it into the high-fidelity prototype. The feature was not implemented in the high-fidelity prototype. My reflections on implementation choices are detailed in Section 7.3.1.

6.1.4 Relationship Integration Overall

The relationship editor, relationship overview menu, and relationship arrows all contributed to integrating relationships in the graph interface. This section discusses feedback regarding the integration as a whole.

The final evaluation provided a new opportunity in testing with users which had some level of interest in technology, but no hands on experience with theory or hardware. Skye was the only user to give suggestions to making the prototype more "homey" by using a blueprint of the house. I have created an illustration of this concept in Figure 6.6.

Maintaining the homey feel in the interface was not my first priority during the project. However, revisiting it in the final user test was interesting, as not alienating the user from their own home is important [52]. If this problem persists for several users, it could also be a sign that they are losing their position in the graph interface, or simply that the graph interface no longer appears to reflect their home. A different argument can be made that perhaps feeling defamiliarized from your home for a period can open up new space for design and technology, such as home automation [5]. Personally, I enjoy feeling at home in my house, and I believe it is important to preserve the mental map in graph interfaces for this purpose [16].

As previously mentioned in Chapter 5, the idea of using a blueprint image of a home was discussed in Controlling the Home, but dismissed. This was both because the graph interface showed more potential in terms of scalability, but also because we found no elegant way of allowing the user to generate the map themselves. Solutions such as the Google Tango Project ³ is on the horizon as possible products to create this mapping.

Skye's suggestion of using a blueprint layout on top of room nodes was therefore

³<https://www.google.com/atap/project-tango/>

combining old ideas, into a new idea. The resurgence of this suggestion across users and across projects leads me to believe that this could be something which many users expect or want. As Figure 6.6 above shows, this would only occur on the home screen, which is only visible before any room nodes are opened. The user stated this explicitly, and I agree as this could be difficult to incorporate without cluttering the interface if we were to populate the image with eight to ten device nodes as well.

6.1.5 Integration of Relationships In Summary

Integrating relationship functionality into the interface was a comprehensive change to the interface. Not all aspects of relationship integration in the interface was tested during the final evaluation. However through comments regarding the parts of the interface which were implemented and other remarks made during evaluations - I believe the integration of relationship functionality to be steps in the right direction.

6.2 End User Programming

The second subquestion is as follows:

How can a design for a non-text-based programming language for a home automation interface on a mobile device be developed?

The development of a non-text-based programming language was decided on after the future workshop. This section described the development process chronologically and is divided into the design period and it's impact on the results, then the user's feedback, and finally a discussion around the user's problems with the functionality.

The results which are discussed in this section are from the final evaluation, if nothing else is stated.

6.2.1 Defining the Functionality of the EUP Implementation

The existing work which I looked towards in implementing EUP functionality were Scratch⁴ and Yahoo! pipes⁵ as described in Section 2.3.

Scratch ended up being the most relevant, I liked the block structure more, and the interface looked easier to navigate on mobile devices. I also had prior experience with the environment.

Scratch, homing from MIT has very useful published material [37, 36]. My takeaways from this research are detailed in the implementation of the prototype in Chapter 4. I looked to this research early in the design phase, when the directions and the biggest decisions were made. Examples of these large decisions were types of logic blocks and how logic blocks connect to each other.

I also borrowed the concept of showing example parameter in blocks with input to show their purpose, examples of this is the timer block and percentage block shown in Figure 6.8.

⁴scratch.mit.edu

⁵<https://pipes.yahoo.com/>

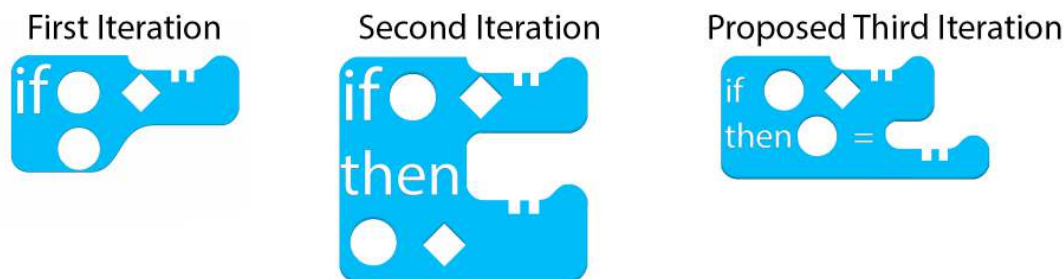


Figure 6.7: The evolution of the if block.

Scratch also overshadowed Yahoo! pipes as Yahoo! pipes facilitated text-based input, I/O operations and web scraping to a greater degree than Scratch, which was not functionality which would benefit this concept at this time. Yahoo! pipes also had a more complex interface, and seeing how I already had seen novice programmers interact with Scratch, I and the users in the future workshop felt confident that it would provide the concept with a base platform making logic "easy to develop" as Lieberman et al. states as the driving force behind EUP [33].

Current applications which facilitate relationships between devices such as homeBLOX [42], openHAB, Jigsaw Editor [23], homeMaestro, Controlling the Home [47], Airfy Beacon⁶ and the Athom⁷ were useful going into the future workshop. One of the common themes in the current solutions presented in Chapter 2 is the three step part of creating a relationship:

1. Select device A
2. Select device B
3. Define what happens with device B when device A gets switched on

This structure was present in all the related software presented in Section 2.

I had this three-step structure in mind when I was designing the first iteration prototype after the future workshop. The initial design for logic statements had only one slot to fill in the "then-clause". The evolution of the if-block is shown in Figure 6.7.

As shown in the first generation if-block found in Figure 6.7, the syntax was "simpler". By "simpler", I mean less blocks required, not usability, as it would show that none of the evaluators found the first iteration if-block logical.

The second iteration added more slots for more logic blocks. The first iteration prototype was low-fidelity. I believe this could be a contributing factor in why the first iteration of the if-block received collective criticism from users. This is because users find it easier to critique something which is less polished than something which feels like it can be altered easily, such as a low-fidelity prototype [20]. The evaluators complained that it was not intuitive, and that it was hard to make sense of. Being a semi participatory project, I

⁶<https://www.kickstarter.com/projects/261147844/airfy-beacon-imagination-meets-smart-home-automati>

⁷<https://www.kickstarter.com/projects/athom/homey-the-living-room-talk-to-your-home?ref=discovery>

decided to go with the feedback from users, but I still feel like the first iteration design had potential through its simplicity and potential speed.

During the final evaluation, I would ask users who were stuck on a task what they were trying to do. The users who responded broadly, often quoting the task text generally struggled more. The users who were able to break down the task into smaller steps generally performed better. This observation led me to look towards other similar findings, and Leigh has found that novice programmers who are able to break down algorithms into smaller abstractions, reflect in a better ability to implement them in code [51].

The change from second to proposed third iteration design of the if-block was motivated by users thinking it was difficult to 'read' the logic. In the final evaluation, I found that the users who pronounced logic out loud while working generally performed better. Being able to put their thoughts into code is hindered by syntax in text-based languages, much less so in non-text-based languages [36]. While Scratch is not a text based language, it still reads as well as a text based language. The attribute of easily readable logic showed its importance during the final evaluation.

As the users who were able to put their programming tasks into words performed better, I believe anything which impedes this should be avoided. Jemma explained her problems with the design which caused problems for her in putting her code into words:

Me: *"Do you find it weird having to put the last 'equals block' in there [the 'then' clause]?"*

Jemma: *"No it's just that, I just felt that when the light was under here, then it was more difficult to understand that the 'on button' would work on it, I mean it's right when you consider everything, but it felt like it would be more natural to have it underneath."*

Me: *"How exactly, to be precise?"*

Jemma: *"They should be below, instead of above, I mean it should've been further that way [further away from where the if-statement starts] because the words are in that direction."*

Me: *"You think it doesn't read well? Here you can read 'if the light is on', while here [the then-clause] it says 'if on light equals'?"*

Jemma: *"Yes, because here it's 'light equals on', while here it's 'light equals blank' so maybe the on button should be moved down here [further down alongside and following the device node and operator]."*

This was feedback I had heard during a strike team feedback prior to the evaluation, but which I did not act upon as the nature of the evaluation environment meant I could not reschedule the evaluation. Upon seeing the importance of a readable language in the results from the final evaluation, I looked towards current research, and how readability affects other EUP applications. Louca compares two novice friendly programming applications, Microworlds, and StageCast Creator. While Microworlds is a traditional text based language, StageCast creator is an EUP implementation where the user saves their animations and logic inside sprites. This makes the code noticeably harder to read, and was giving novice users issues when reading the code [34].

The aesthetics of the first generation logic blocks were designed by myself, not by the users. I used Maloney et al.'s article "The Scratch Programming Language and Environment" [37] as a reference to the base of each design. Basing my design off of Scratch was

an easy decision, as I already had experience with the environment and I knew that it worked well. I also gathered some feedback peers through strike team feedback, which was helpful and worked well.

I attempted to make it clear that each puzzle piece had a slot, and only one slot where it fit. This is why the if-block can be populated by three different shaped pieces:

- Circle shapes device nodes
- Rectangular operator blocks
- Custom shaped on-off-percentage-timer blocks.

Based on Maloney’s choices [37], I used different colors as far as possible, so the blocks were easily distinguishable from each other. I also created the custom shape used in the on, off, percentage and timer blocks myself to resemble a key.

In the next section, my findings and ideas of adding more logic blocks to the EUP language will be discussed.

6.2.2 Additional Logic Blocks

The idea of adding additional logic blocks has been present in the project since the future workshop. In the future workshop, we discussed having a menu for selecting which type of block the user wanted from the toolbox. I decided on not implementing this, as I was in Heim’s words: “most concerned with completeness. We must make sure that users can actually complete their tasks with the functionality provided by the conceptual design” [20].

I decided on not implementing this, as testing the interface with fewer blocks first would be more fruitful as feedback would not be centered around a low-fidelity menu, but instead on the EUP itself. The addition of more blocks would not necessarily lead to higher quality results, but could instead lead to the introduction of too many new variables at once. This would in turn make it harder to find the actual issues and which changes to make in order to improve the concept.

The toolbox where all the logic blocks are located is shown in Figure 6.8.

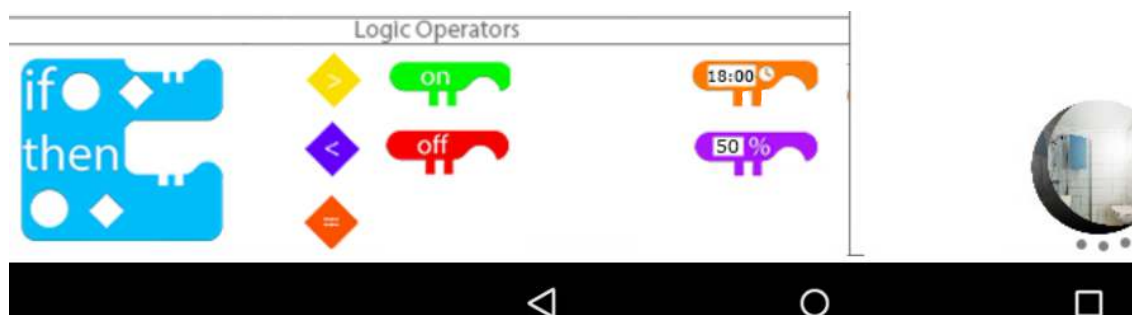


Figure 6.8: The toolbox housing logic blocks. Picture from the second iteration prototype.

The introduction of a timer block is essentially a kind of function block which we would know from most programming languages. This was suggested by Grant within minutes of interacting with the prototype. He identified that the operation of creating a timer was repetitive and something which would likely be done many times, and suggested we merged



Figure 6.9: Left: A common timer structure from the second iteration prototype. A timer relationship always requires a clock -, equals -, and a timer block. Right: Mock up of new timer block. Here the clock and operator is fixed, the user only selects the time. If not for spacing, it could also be more fixed and read "when the time is...".

the steps, making the process faster and with fewer parameters as input. This concept had also been brought up previously during the expert evaluation and future workshop, signifying the need for this functionality. A mockup of this concept is shown in Figure 6.9.

I liked this idea and the reasoning behind the timer block, so I suggested it to other users afterwards as well. This idea generally received positive feedback, and shows that the toolbox needs to be altered with a menu in a next iteration as there is no room for additional logic blocks. A "delay" functionality was also suggested by Grant, but was not suggested by any other users.

The suggestions for additional blocks shows the need for facilitating more logic blocks in the toolbox through a menu system. This menu would categorize different blocks, As the concept presented in this paper consists of four different type logic blocks, I believe Scratch's menu would work also in this smaller format. Scratch's different blocks are shown in Figure 6.10.

Scratch's menu is shown in Figure 6.11. As a suggested design in implementing this menu, the top tabs in this figure, "scripts, costumes and sounds" could be replaced by categories representing each type of block.

The Scratch programming language facilitates the same logic structures which traditional text based programming languages do [36]. The EUP implementation in this project is less comprehensive, and could therefore fit the menu options in as tabs, Scratch's menu is detailed by Maloney et al. [37]. This would be a speed bump to the user when selecting pieces, but it is necessary when adding additional EUP options.

Looking back at the previous section where the code's readability was discussed, readability should be maintained in the introduction of any potential new logic blocks seeing how most users had errors and needed help at times.





	<p>A <i>command</i> block has a notch on the top and a matching bump on the bottom. Command blocks can be joined to create a sequence of commands called a <i>stack</i>.</p>
	<p>A <i>function</i> block returns a value. Function blocks do not have notches.</p>
	<p>A <i>trigger block</i> has a rounded top. It runs the tack below it when the triggering event occurs.</p>
	<p><i>Control structure</i> command blocks have openings to hold nested command sequences.</p>

Figure 6.10: The different blocks in the Scratch software. Figure from Maloney et al. [37].

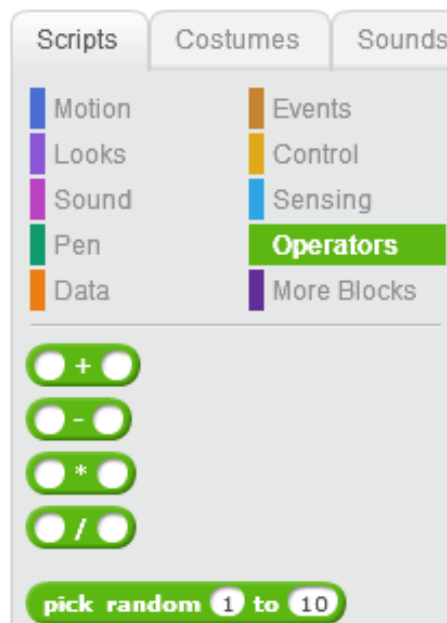


Figure 6.11: The logic block selection menu in Scratch.

6.2.3 Errors

As expected, not all users were able to complete the EUP tasks correctly without help. For the evaluations, I was present at all times and would help the users if they got stuck. This was expected as asking users with no programming experience to program, even simple structures is asking a lot when if they are not to use any help or tutorials.

I asked the evaluators if an on-screen tutorial could replace me in guiding them through the application. Most users were positive, but Skye said she preferred to explore the application on her own.

An interesting interaction occurred when Calvin tried to press the empty slots in the if-block to get more information on what could fit there. This shows a clear expectation from him, this is a suggestion of how a tutorial could be implemented into the application. This type of help implementation was found by Inbar et al. to be the preferred method on mobile devices [25]. This interaction between Calvin and the prototype made me look more into mobile tutorial research, as tutorial integration had been suggested previously during testing in Controlling the Home [47]. Novick and Ward [40] found that the most common causes for users not seeking help in applications were:

"Users cite difficulties in navigating the help systems, particularly difficulties in finding useful search terms, and disappointment in the level of explanation found."

In Calvin's suggested implementation, these concerns are all potentially addressed. The user has a clear expectation of what they want. This is so called "pulled" help. The difference between pulled and pushed help is explored by Jepsen et al. They used user user testing found that the only factor to consider when implementing a help system is the "perceived criticality" of the action [27]. The perceived criticality of Calvin's action is not determined however.

Based on the amount of errors the users had with the first training task, I think there is a clear necessity for an optional tutorial or helper functionality. This is supported by Cao et al.'s findings of how end user programming has different patterns in when users need help when stuck, and what happens when they are stuck. "End User Programming should support other phases than implementation" [8]. While providing support or help during programming is indirectly supporting the implementation phase, I would consider the user to be in another state of mind when they know how to achieve their goal, and when they are stuck.

Question seven in the questionnaire concerned this topic:

"I think that I would need the support of a technical person to be able to use this system"

The users did not think it was necessary to have a personal aide, as they replied on average 1.25 on a scale of 1-4.

Only two users answered '2'. Question 10 in the questionnaire also concerned a related topic:

"I needed to learn a lot of things before I could get going with this system".

The average score was 1.44⁸.

⁸One user answered between the scores of 2 and 3. I calculated the average using 2.5

```

//the prototype's solution, in code:
if(deviceA.state){
  deviveB.setOn();
}

if(!deviceA.state){
  deviveB.setOff();
}

if(deviceB.state){
  deviveA.setOn();
}

if(!deviceB.state){
  deviveA.setOff();
}

//an easier way:
if(deviceA.isPressed){
  deviceA.state = deviceB.state;
}
if(deviceB.isPressed){
  deviceB.state = deviceA.state;
}

```

Figure 6.12: A pseudo code representation of the work flow in the prototype. Left: How it is currently. Right: A simpler version which the prototype does not facilitate.

In question seven, the user is asked to look outside and to how they perceive other users' ability to use the system:

"I would imagine that most people would learn to use this system very quickly"

The average score was 2.75, where 4 is the best. This suggests that the users had more faith in themselves than others. I tried to emphasize that young adults with an interest in technology was the main demographic, but it is possible that the users widened this question to include people not in the target audience such as older people.

The last training task asked the user to connect two lights together, so switching one light 'on' or 'off' would do the same to another light. With the tools given to the users, this required them to use four if-blocks to solve. As none of the users came to this conclusion themselves, it points to the need for either the tutorial mentioned previously, or a way of assigning device states to reflect that of another devices'. The code in Figure 6.12 shows this.

If there was an elegant way of translating option 2 in the code above to the prototype's EUP language, that was be preferable. Completing this task within the prototype was cumbersome, as shown previously in Figure 6.13. While not suggested by users, a possible "connect two devices node" could be an answer. The design of a "connect two devices" node was not discussed with any users. However, given the prevalence of this kind of relationship [7] it could be useful.

The topic of ambiguous device nodes shown in Figure 6.13 will be discussed later in Section 6.4.3.

6.2.4 EUP in Summary

The end user programming implemented in the prototype was successful in facilitating the creation of programming logic between devices in the prototype. The evaluators found the

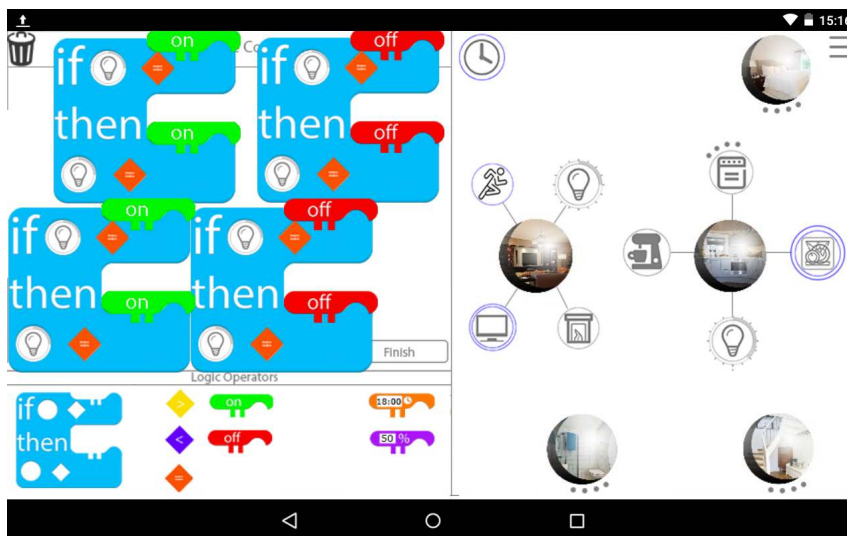


Figure 6.13: The complete solution to training task 3. None of the users completed this, instead they stopped after one "on-on"-if block, and one "off-off" if block. This also highlights the problem of ambiguous device node icons.

interactions to be interesting and fun, and they also suggested new blocks and structure to the language. Certain relationship structures were not facilitated well enough however. Overall, the users were able to quickly grasp the concept, with my help.

6.3 Sensors and Utilities

The third subquestion of the thesis is:

How can device utilities and sensors be visualized in a graph based home automation interface?

This section is divided into two parts, discussing sensors, and device utilities separately.

6.3.1 Sensors

Sensors were a new integration in the interface. Sensors had been discussed as additional functionality since Controlling the Home, but were put on hold as the functionality was outside of the project's scope. In this thesis however, I brought sensors to the table to begin with in the future workshop. This was based on the prevalence of sensors in current home automation solutions [7].

The final evaluation had sensors as a topic both during the tasks, and the interview.

While sensors were liked across most users, there was some criticism to the functionality. Jemma complained about the icon for the movement sensor, she suggested to use someone sneaking, or a pair of footprints instead.

Jemma: "I found it simple, to see the difference in the icons. But the movement sensor, I feel like the icon means 'activity', rather than movement. But that's a bit tricky, maybe feet would've been a better choice for a sensor perhaps? Or footprints?"



Figure 6.14: A suggestion for the new sensor visualization following Jemma and Calvin’s opinions.



Figure 6.15: Device states and sensors. Left to right: movement sensor off, movement sensor on, other device on.

Using a sneaking person as the icon for a movement sensor was suggested in the interview with Leopold. Using icons in the interface will be discussed in greater detail in Section 6.4.3. Using user testing to evaluate how well understood icons in interfaces are was found to be the best way by Schroeder and Ziefle, they state that icon based interfaces will work if users are involved in the evaluation and “if icons represent prototypical semantic knowledge” [45]. This is an example of an icon which based on user feedback did not meet the last requirement.

Only Calvin thought the sensors could use a different visualization, but not in the form of a new icon:

Me: “What did you think about visualizing a sensor in the interface in the same way other device nodes are visualized?”

Calvin: “I thought it was logical, but maybe there could’ve been some small indication that they weren’t the same. But at the same time, a sensor is a device, and it’s easy to think that “there’s a fireplace, there’s a light, and this is a TV, I guess the running man is a treadmill.” But that could also be fixed by another symbol for the sensor, not necessarily a different indicator.”

To show what the users envisioned, I made a mockup of the concept, shown in Figure 6.14. Here the outline color has been changed from blue to green, to help it stand out from device nodes. Green was not a color suggested by the users. For comparison, this new suggestion in Figure 6.14 can be compared to the old visualization shown in Figure 6.15.

This suggestion was unique, but not unexpected. The current design choice for sensors was one I made on my own. This was making the sensor similar to an ‘on’ device, but not fully ‘on’ in the same way the device node is ‘on’. Figure 6.15 shows this.

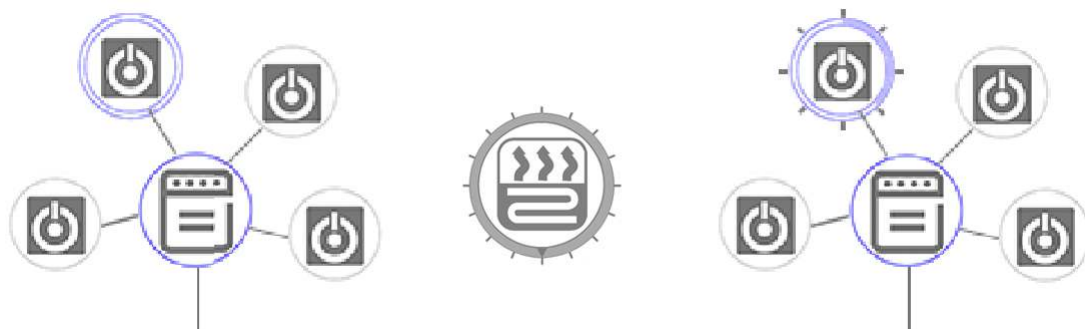


Figure 6.16: The evolution of the device state visualization.

This was a decision which I was unsure of, as the only other device which shared the "single blue circle" visualization was multiple-utility devices. None of the users commented on this however, indicating that it was understandable.

6.3.2 Device Utilities

Device utilities refer to devices which have multiple options. An example of this is a stove, which has multiple hot plates available. The evolution of this visualization is shown in Figure 6.16

The design of device utilities only received minor changes from the first to last iteration concept design. As shown in the figure above, the only change made to the device node after the future workshop was the alteration of the child nodes, to match the new floating value device nodes.

A small addition to the interface prior to the expert evaluation was the "child node indicators". These small indicators would show the users how many children a node has. I applied this concept to room nodes as well for consistency. The concept is visualized in Figure 6.17.

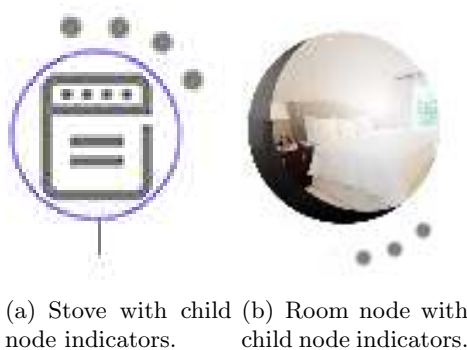


Figure 6.17: Child node indicators on both room- and device nodes.

During the final evaluation, the change was received positively by users, who thought it was logical. They did not comment on whether it was necessary however, and some users did not find it intuitive, but thought it was logical and liked it once their purpose was explained.

A larger concept for device utilities was perceived during the expert evaluation however, it involved using a "radial"⁹ interface.

Radial interfaces are common in video games, as they provide a smaller average distance to menu items, which is beneficial in interactions which should be quick [32, 44]. One of the key benefits of a radial interface is that it takes less space than a comparable list menus. Space is precious in this concept interface. One of the weaknesses of the radial interface is visible if the labels on each item is long [44]. By using icons, this problem is voided, if the icons are understandable. Icons will be discussed later in Section 6.4.3. Several examples of radial menus are shown in Figure 6.18.

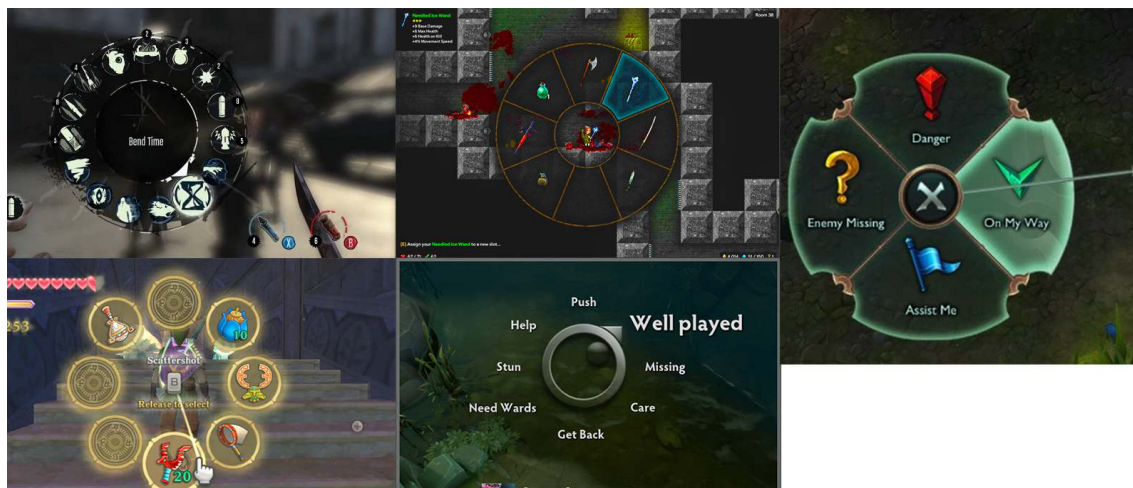


Figure 6.18: Radial menus from several games. Top left to right: Dishonored, Tallowmere, League of Legends, Zelda and Dota2.

I chose not to implement this into the final user test due to a lack of resources.

While I see huge potential in this concept and I think this is the easiest and fastest interaction style for device utilities, there are some problems to overcome. How does this interface work in combination with floating value nodes? For selecting a single item or setting, we can hypothesize that this interaction style would work well. I discussed the radial menu with Leopold during user testing, he on the other hand, did not like it:

Me: *"A washing machine for instance, has several programs. You have soft fabric, satin, wool and many more. What do you think of using a wheel where you select the washer, then drag out from center in order to select your program?"*

Leopold: *"I would prefer if you could press here, and go into the menu, so you don't have to keep your finger on the screen. Like how the stove works now."*

I do not want to read too much into this sole opinion before testing this. Testing interaction styles with low-fidelity prototypes is a difficult concept [20]. This is also something I have observed myself during Controlling the Home testing. During testing in this thesis, I observed that users would play a lot more with the high-fidelity prototype, and that the low-fidelity prototype did not engage the users in the same manner. High-fidelity prototypes provide instant feedback and animations which is important in testing interactions.

⁹Radial interfaces are also known as wave, pie, or wheel interfaces

This is perhaps also a big factor in the questionnaire results, showed an improvement from 76.6 average in the expert evaluation, to 82.5 average in the final evaluation. This is a very similar score to Controlling the Home, where the prototypes were similarly presented to users, and the low-fidelity prototype scored 77.91, and the high-fidelity prototype scored 84 [47].

6.3.3 Sensors And Utilities in Summary

Sensors worked well. The users were able to complete the sensor related tasks quickly and without error. The integration of sensor utility was not a comprehensive integration, by visualizing sensors similarly to device nodes, the sensor nodes and how they work in EUP was intuitive to users. Some minor suggestions were made, and making tweaks to differentiate device nodes from sensor nodes seems like the correct approach. This was shown previously in Figure 6.14.

In regards to device utilities, a lot of feedback was gathered. Unfortunately, the major suggestion of implementing a radial menu which emerged from the expert evaluation was not implemented due to lack of resources. I believe a high-fidelity prototype is necessary in order to test this concept's integration in the interface. The device utility which was implemented and tested worked satisfactory.

Device utilities visualization is tied to device state visualizations, which will be discussed in the next section.

6.4 States of Devices

The third subquestion in this thesis is:

How can the state(s) of devices be visualized in a graph based home automation interface?

This section discusses the development choices and findings of device node state visualizations in the concept, from the basis presented in Controlling the Home, to the final prototype. The section also covers a discussion on the change from text to icons in the interface. This change was not the focus of the research, but being closely tied to the device nodes, I wanted to discuss this here.

6.4.1 Visualizing the State of Devices

As previously mentioned, the visualization of device states is closely tied to the previous research question regarding sensors and utilities. The users were critical of the current solutions in Controlling the Home in this area. This feedback is shown in Figure 6.19

The device node visualizations were split into two categories:

1. Binary Device Nodes: 'On' or 'off'.
2. Floating Value Device Nodes: Hot plates or lights, devices which are not just 'on' or 'off'.

No concrete suggestions were made during the future workshop in improving these issues. This meant that the first iteration design was drafted by myself, where the main

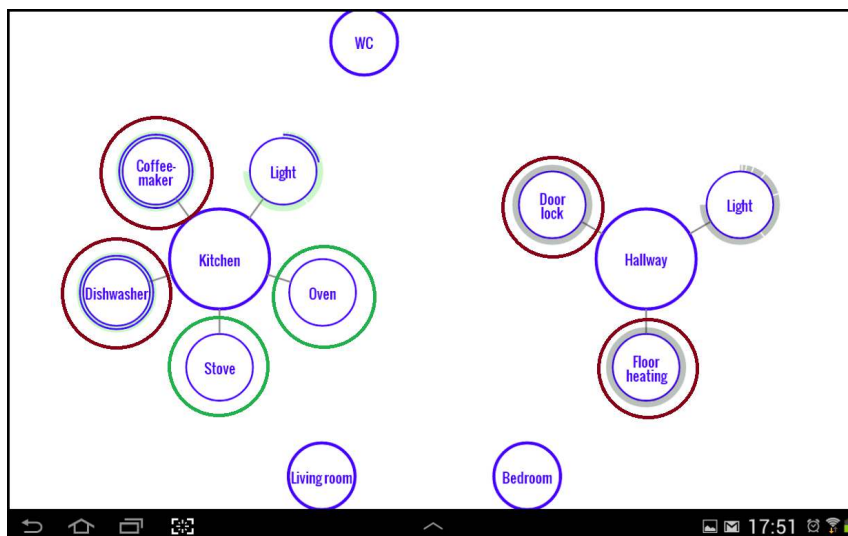


Figure 6.19: Inconsistencies in the Controlling the Home prototype. The red circles are devices which should behave similarly, but the pair in the kitchen node has double blue lines, while the hallway nodes have one blue line and a thick grey outline. The grey outline is intended to be 'off', but this was not clear to users. The green pair has only a blue line as they have child nodes of their own similar to rooms, but the users did not understand why they had no border.

goal was to remove ambiguous visualizations from Controlling the Home. The previously shown Figures 6.15 and 6.16 shows the evolution of both the binary and the floating value device nodes. During the final evaluation, I gathered feedback on how the user felt devices being 'on' or 'off' was conveyed through the interface. Most users understood which devices were 'off', in addition, some had issues seeing which devices were 'on'. Some of the users who struggled telling which devices were 'on', also complained that the icons were too small. This leads me to believe that there is a connection between size and difficulties in seeing the color difference showing the state of the device. Suggestions were made to overcome this, both Leopold and Skye wanted to use red and green to indicate if a device is off or on. A reflected conversation with these users led to the solution of keeping the blue interface as an option for color blind or otherwise vision impaired users. Using red or green instead of blue was the same feedback and suggestions as gathered during Controlling the Home. I chose not to alter this as I wanted to see if redesigning the nodes, and making thicker borders would be sufficient in order to overcome the problem.

These reflected discussions were possible because of the evaluators both in the expert and the final evaluation. Going back to the open coding process, there is a clear pattern in conversations tagged "reflected" and their fruitfulness to the research topic. Reflection is one of the qualities which Barriball and While lists as important for users when conducting semi-structured interviews [35]. Similarly, the same users also did not possess the negative qualities described by Oppenheim [41].

6.4.2 Floating Value Device Nodes

The floating value device nodes were changed from Controlling the Home. This design change was prompted based on the future workshop, where users thought the device nodes were overall too difficult to understand. The subsequent design alterations is shown in

Figure 6.20 and was based on results from the expert evaluation. These findings were from users thinking out loud, as described by Someren et al [1]. It was noticeable that users with more experience in user testing were able to utilize think aloud to a greater extent than users who had not [28, 53].



Figure 6.20: The evolution of the floating value device node. Left to right: Controlling the Home, first iteration design after future workshop where the node is 'off' at 50%, then 'off' and 'on' at 50% of the second iteration design tested in the final evaluation.

During the final evaluation, users still thought the state of floating value device nodes was hard to make out. I believe there are other factors which could be tweaked in order to make the state of device nodes to be easier to tell. Mainly, this entails making the icons bigger, as was also the case with binary device nodes.

6.4.3 Icons

The device nodes were changed to icons from text in this prototype.

In training task 3, the users were tasked with connecting two lights together. These two lights were found in different rooms, but had identical icons. Some users commented that it was hard to keep track of which node corresponded to which device in the rooms. Figure 6.13 above highlights this problem. Leopold suggested a solution:

Leopold: *"I can't tell the difference between the icons."*

Me: *"No, you can't, there's no difference between them. Do you have a suggestion for some way of distinguishing between them?"*

Leopold: *"Yes, because as it is, I can't tell the difference between these, if I was to revisit this and look at it later, I wouldn't know which lightbulb is in which room. [...] Maybe you could have a different icon 'on' them, depending on which room they come from?"*

Me: *"Such as using the room node as an indicator?"*

Leopold: *"Yes!"*

Using images introduces new problems however. A new concern which was not considered is multiple devices of the same type in the same room. Both during the expert evaluation as well as the final evaluation I observed that the users were very diligent in finding inconsistencies and minor concerns in the design and concept. It was therefore unexpected that none of the users identified the problem of having two devices with the same device node icon.

My implementation of device nodes as icons was based purely on user feedback. Seeing the evaluators' response led me to look towards relevant work. Using user testing to evaluate how well understood icons in interfaces are, was found to be the best way by

Schroeder and Zieffle [45]. For ambiguous devices, or devices which is hard to visualize with a icon, combining an icon and a label will still be more descriptive than the label alone [14].

6.4.4 Device States in Summary

Changing the descriptive attribute of the device nodes from text to icons was successful. The icons were understandable to all users. There were however some concerns of distinguishing similar devices from each other. Some evaluators thought this was quite cumbersome, and suggested incorporating the room node into the device node in order to distinguish between them. The size of the device nodes were an issue for some users, and my observations suggest that the second iteration floating and binary node design, in combination with an increased size should be sufficient to overcome these problems.

6.5 Summary

In this chapter I have discussed findings from the final evaluation, as well as findings from earlier design iterations and testing.

The end user programming solution developed in this project was well received by the novice users. Some shortcomings were unveiled during testing, and the cumbersomeness of connecting two devices to one another was one such case. New programming structures which would benefit the interface was also suggested.

The integration of relationships into the graph interface worked well. There is however some uncertainty as the relationship editor was a static element in the high-fidelity prototype, instead of implemented. This means there are possibilities of problems in the integration of the menu. The relationship overview menu was not implemented in the high-fidelity prototype, however the expert evaluation showed positive feedback to the menu to work in the interface.

Sensors were implemented in a satisfactory way, and the functionality was understandable and intuitive to users.

Device states received some criticism, but were overall understood as well. Using more traditional green or red colors to indicate 'on' or 'off' could work, but the blue interface needs to remain an option to ensure universal usability. In addition, device nodes should be made larger so the visualizations are easier to see.

No major redesigns to device utilities were tested. The concept of using a radial menu was presented however, and while the users believed this could be cumbersome, I believe this change needs to be implemented in order to be tested sufficiently.

Chapter 7

Conclusion

”In a few decades time, computers will be interwoven into almost every industrial product”

Karl Steinbuch

This chapter is divided into three parts, first I will conclude my research questions, then I will look towards new directions for future work with this project, finally I will list my reflections on the project.

7.1 Concluding the Research Questions

In this thesis, I have explored different aspects of integrating relationship visualization into a graph based interface. The thesis is based on previous work in the Controlling the Home project [47].

Changes to sensors, device states, utilities and end user programming implementation and integration were evaluated by eight users in the target audience for the concept.

The users were tasked with using the concept’s tools to complete tasks simulating every day life by using two prototypes, a high and a low-fidelity prototype. The evaluators expressed that the tasks in the evaluation were logical. Complex relationships caused problems for the users, and I had to explain the solution each time. After the first end user programming focused task, the users found the next tasks to be easier. All users instantly began to solve the new tasks, with little to no thinking before trying a new solution. There were some errors, but the simpler relationships caused no problems for most users. The difficulties with complex relationships such as connecting two devices and timers can be aided by introducing new control structure blocks to the end user programming interface. This shows that the target audience for home automation is capable of end user programming if it is facilitated well.

A menu providing the user with an overview of relationships created within the prototype was evaluated in an expert evaluation by six testers with programming experience. The list menu was simple and all users found it easy to use. This shows that a list of active relationships is a good choice in providing an overview in the concept.

Visualizations for a device node’s state and utilities were tested. Device states and utilities received positive feedback, but still require changes until they are unanimously

approved by testers. New concepts for device utility visualization emerged and will require testing in future projects.

Sensor visualization was implemented and tested. Sensors are one of the most common tools in home automation today [7], and their place in the concept was evaluated by testers and found to be logical and well functioning.

In conclusion, this project has provided a new direction for the graph interface developed in Controlling the Home. The introduction of EUP to the interface was well received and tested to be found understandable by users who possessed no prior programming experience. The final evaluation in this thesis shows that this interface and its EUP implementation is understandable and a valid direction for future development.

7.2 Future Work

During this research, several ideas and concepts emerged which were not pursued, or could have been pursued differently. These findings and thoughts are detailed in this section:

7.2.1 Further Development Of the EUP Implementation

The EUP implementation needs several changes in order to accommodate and facilitate the needs which users will have when using the system. The most urgent additions are:

1. The "AND" and "OR" operator.
2. A "time between" block. This could also be managed by using a AND block. The redundancy here needs to be evaluated.
3. The "connect device" logic block. As shown in Figure 6.12, this could simplify a common structure in the programming language.
4. A delay functionality. The ability to say "do this in 5 minutes" was suggested by users as something useful.

7.2.2 Room Nodes

While the users liked the room nodes being pictures, and spherical, I believe the first strike team's feedback during design of the first iteration to be correct. The room nodes should follow the same design guidelines as the rest of the interface, and be flat. This is not a usability concern, but an aesthetic change.

7.2.3 Light Indicators in Room Nodes

As shown in Figure 4.8, the room nodes had a shade which indicated the total light level in the room. During the final evaluation, I asked the users what they thought the shade meant. Half the users thought the shade was just to make the room node appear more as a sphere, while the other half thought it had something to do with lights. I did not present these findings in Chapter 5 as they ended up not being relevant to the research topic.

There is potential in this visualization, but in order to test this, it needs to be implemented into a high-fidelity prototype, I discussed this with Skye:

Me: *"Did you think anything about the shadow around the room node?"*

Skye: *"No I thought that was for a 3D-effect. It's lights!"*

Me: *"Do you think that is useful, or messy?"*

Skye: *"Well it took me 2 seconds to figure out, but it is logical."*

Me: *"Would it have been easier if it was implemented, to see that the shade adjusted itself?"*

Skye: *"Yes I would think so"*

There were other findings which needs to be implemented in order to be tested satisfactory such as new device utility functionality.

7.2.4 Device Utilities

I see potential in implementing a radial menu for the application. This menu should also facilitate floating value input at the same time. This could be done by making the mother node a floating value node, and select children based on the radial menu.

An alternative to the radial menu, or one which can be combined, is the home automation operating system suggested by user 5 during the future workshop. If the user can download an app to control the TV, or the dishwasher or any other device with several utilities, then the need for a generalized utility visualization is less pressing.

7.2.5 Device States

Device states did not receive as much feedback as the other topics, which I attribute to the observation that users had generally little feedback on things which they liked. As presented in Chapter 5, this effect is known [15, 43], and people would often look for something which they did not like when commenting. In turn, this means that the issues which device states had, were perhaps overshadowed by other elements in the concept and interface.

For future work in regards to device states, there are several options for tweaking:

1. Make the icons larger
2. Make the icons red or green if the device is on or off, add colorblind mode.
3. Make the border of device nodes thicker.

All of the options above can be changed in order to ensure device states being understandable for all users. Based on the results from this project, it is unlikely that all these three factors needs to be used in order to achieve this goal.

Another concerns which was never addressed in this project is the need to distinguish between ambiguous device nodes. For future work, I suggest using the room node as an indicator when the device node is used in the relationship editor.

7.2.6 Relationship Arrows

The final iteration did not test this functionality. It is a big tool in the graph interface, and while its functionality overlaps with the relationship overview menu, their means of conveying information and overview are different. This should be implemented and tested. Alternatively, the relationship overview menu could be implemented by itself with no relationship arrows, in order to evaluate if the arrows are redundant.

7.2.7 Implementation

The prototype is very rigid in its hit detection. Determining what the user has pressed needs improvement.

Currently, the prototype only checks coordinates within a rectangle based on the piece's size for what the user has selected. In a next iteration, a better way which does not rely on a rectangle to work is to test the pixel color, then test for position if there is ambiguity.

7.2.8 In-app Tutorial or Help

The final evaluation showed that the prototype could need a help menu implemented. For how to implement this help menu, a user suggested a very non-intrusive method of tapping empty slots in the if-block to get some kind of feedback or indication of which blocks can fit in there. Implementation concerns which I think are important are detailed in Section 2.2.2. User thoughts and my observations on the topic are found in Section 6.2.3. There was also an early idea during the design of the first iteration to have the slots in the relationship editor light up if the user was dragging a node which would fit there. This was not implemented however, both due to resource constraints and how it was not suggested during the expert evaluation.

A similar observation is which stance the interface should take to relationships which "do not make sense". Some kind of auto-detection of relationships which are odd, such as "Set the stove to max when the lights are off" should be warned for. A key element here is heating elements, and other sources of heat.

7.2.9 Energy Conservation

Reducing energy consumption has not been a topic in this thesis. I believe there to be potential in integrating energy conservation into this interface. This was not discussed directly by any users, but a comment was made during the future workshop that being able to monitor circuits to make sure the user would not overload them. This could be the start of being able to create rule based behavior based on current energy prices, such as "start the machine after 1 am if the power is less than a certain amount per kWh".

This would require a back-end connection to devices, which was simplified in this thesis and not considered. This is an obstacle which needs to be overcome in order for the front end interface presented here to be useful.

7.3 Reflection

During this work, there are a some things which I am satisfied with, and some things which I am less satisfied with. This section will detail my thoughts on some parts of the methodology and execution.

7.3.1 Implementation

When development started for the high-fidelity prototype, I needed to decide on a scope. Being inexperienced with the Android platform¹ was my first issue, but using this platform was still the easiest way to a satisfactory high-fidelity mobile prototype. This was due to the devices I had available, and the only major alternative, the iPad² being something I had no experience at all with.

Before development started, I brainstormed ideas of creating several high-fidelity prototypes which would each demonstrate a piece of functionality. This would entail creating one prototype where the user could only drag-and-drop logic blocks into an if-test, as well as another high-fidelity prototype simulating the graph interface. Due to time constraints however, and how the relationship editor ended up tying into the graph interface by using

¹<http://developer.android.com/tools/studio/index.html>

²<https://developer.apple.com/ipad/sdk/>

the graph as a part of the toolbox for fetching device nodes, the solution I chose was to reuse most of the old low-fidelity prototype for simulating the graph interface, and to create the relationship editor without implementing the functionality to navigate the graph interface.

Relationship arrows were planned to be implemented in the high-fidelity prototype, but were not due to time. This leaves a hole in the question of how relationships can be integrated in the interface, but can still be picked up as future work.

7.3.2 Execution

This project has had four subquestions to the main research question. Each of these four questions concerned a specific piece of functionality, and some covered more in combination. In hindsight, the scope larger than necessary. While I maintain that updating device states and utilities in order to match the rest of the interface, this could have been done without as much focus and pressure on it. The broad scope made it harder to focus as much as I liked on each topic.

Two examples of this is not gathering enough feedback on sensors during the expert evaluation, or lacking the resources for implementing and testing relationship arrows during the final evaluation. The final evaluation's format was good, but also meant that I could not interview the users on every conceivable topic in the same way which I could with users under different circumstances. The overview menu was also not tested during the final evaluation. And while I believe that the final evaluation would have shown that the menu was understood by the testers, I am not able to be completely certain.

In hindsight, it would have been interesting to do a second EUP centered workshop after the future workshop where the users would have created their own suggestions for these designs. This way, I would not have dictated the direction of the design quite as much, and the background of the participants was more than sufficient to have opinions on EUP implementations. It is possible that this would have progressed the EUP concept further within the project time frame, such as adding more logic blocks and facilitating their place into the interface. A concrete example of this is that the topic of "code reading well" could have surfaced earlier in the development.

7.3.3 Testing

Ever since Controlling the Home, the experts have suggested to test the concept on non-tech-savvy individuals. When the opportunity presented itself to evaluate the concept using not only people who had little to no programming experience, but who also were young adults with interest in technology I was excited to try.

Testing on The Gathering turned out much better than expected. The reflective conversations with the evaluators provided better quality data than I had imagined. For future projects, I will be more positive to testing against the target audience, even if it requires more resources. Using testers whom had some personal relationship to me, being mainly friends of friends seemed to make them care more for their response and feedback. I included one completely random user³, and his feedback was not as fruitful as the others'. While one user is a small sample size, it was not so much that the random user's feedback was disappoint, but rather that the other users' feedback exceeded my expectations.

³There were two random testers, but I chatted longer with one of these subjects before testing, and he was overall more prepared for being part of the evaluation.

Including a questionnaire during the expert evaluation was not as useful as I hoped, as the tests ended up being more different than I had initially planned in order to cover all topics in the final evaluation. I believe this made individual results less valuable to compare, the results were therefore not compared for individual questions.

7.3.4 Data Analysis

The data gathered during the final evaluation were transcribed, and then underwent open and selective coding. This was a thorough process which made processing the data easier. As a whole, the process seemed to make it easier to present and structure the data, compared to the simpler analysis used for the data gathered during the expert evaluation and future workshop.

In hindsight, I believe the time investment in open coding was similar to the time saved afterwards when presenting the data, comparing to the simplified analysis. In the end however, I felt more confident that I had not missed any topics when using open coding, due to the overview it provided.

7.3.5 Relevant Work

Towards the delivery date of this thesis, themes and topics in the work began to present itself which were previously not apparent to me. These topics were generally things which were already present in *Controlling the Home*. This includes the icon and mobile tutorial presented in Section 2.2.4 and 2.2.2, as well as the radial menu interface detailed in Section 2.2.3.

Another case was the icon theory presented in Section 6.4.3. The users provided detailed ideas regarding all aspects of the icon implementation. Therefore, it did not occur to me to look to previous work. It was not until problems with the icons were discovered during the final evaluation. There were also completely new topics which arose during testing which I had previously not considered such as the theory behind code "reading" well. This could have brought better solutions to the table quicker, making the findings more refined.

By telling peers about my thesis, sometimes relevant work and ideas would be suggested. An example of this was an older bachelor thesis which involved home automation interfaces. This work in particular proved to be not very relevant, but I would continue to tell more people about my work to gather ideas from all sources.

Bibliography

- [1] The think aloud method - a review of literature. <https://calebjc.files.wordpress.com/2010/05/litreviewthinkaloud1999.pdf>, 1999.
- [2] Heino Apel. The future workshop. http://www.die-bonn.de/esprid/dokumente/doc-2004/apel04_02.pdf, 2004.
- [3] Debasis Bandyopadhyay and Jaydip Sen. Internet of things - applications and challenges in technology and standardization. <http://arxiv.org/pdf/1105.1693.pdf>, 2011.
- [4] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. http://uxpajournal.org/wp-content/uploads/pdf/JUS_Bangor_May2009.pdf, 2009.
- [5] Genevieve Bell, Mark Blythe, and Phoebe Sengers. Making by making strange: Defamiliarization and the design of domestic technologies. <http://doi.acm.org/10.1145/1067860.1067862>, 2005.
- [6] John Brooke. Sus - a quick and dirty usability scale. http://cui.unige.ch/isi/icle-wiki/_media/ipm:test-suschapt.pdf, 1996.
- [7] A.J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: Challenges and opportunities. <http://doi.acm.org/10.1145/1978942.1979249>, 2011.
- [8] Jill Cao, Yann Riche, Susan Wiedenbeck, Margaret Burnett, and Valentina Grigoreanu. End-user mashup programming: Through the design lens. <http://doi.acm.org/10.1145/1753326.1753477>, 2010.
- [9] Rodney Carlisle. *Scientific American inventions and discoveries all the milestones in ingenuity—from the discovery of fire to the invention of the microwave oven*. John Wiley and Sons, Hoboken, N.J, 2004.
- [10] S. Casey. Women invent: Two centuries of discoveries that have shaped our world. <http://books.google.no/books?id=911RAAAAMAAJ>, 1997.
- [11] N. J. Clifford. *Key methods in geography*. Sage Publications, Thousand Oaks, CA, 2010.
- [12] David Culler, Deborah Estrin, and Mani Srivastava. Overview of sensor network. <http://www.computer.org/csdl/mags/co/2004/08/r8041.pdf>, 2004.

- [13] K. Dunn. *Qualitative Research Methods in Human Geography*. Melbourne: Oxford University Press, 2005.
- [14] C. Egido and J. Patterson. Pictures and category labels as navigational aids for catalog browsing. <http://doi.acm.org/10.1145/57167.57188>, 1988.
- [15] Margareta Firman and Bo Edvardsson. A content analysis of complaints and compliments. <http://search.proquest.com/docview/197988146/fulltext/C90C1B7F599F4FDFPQ/1?accountid=43205>, 2003.
- [16] Manuel Freire and Pilar Rodriguez. Preserving the mental map in interactive graph interfaces. <http://doi.acm.org/10.1145/1133265.1133319>, 2006.
- [17] Tarjei Garsjø. Personal communication with press crew at tg15, 2015.
- [18] Barney G Glaser and Anselm L Strauss. The discovery of grounded theory: Strategies for qualitative research. http://www.sxf.uevora.pt/wp-content/uploads/2013/03/Glaser_1967.pdf, 1967.
- [19] Alfonso Gárate, Nati Herrasti, and Antonio López. Genio: An ambient intelligence application in home automation and entertainment environment. <http://dl.acm.org/citation.cfm?id=1107609>, 2005.
- [20] Steven Heim. *The resonant interface : HCI foundations for interaction design*. Pearson/Addison Wesley, Boston, 2008.
- [21] Rashina Hoda, James Noble, and Stuart Marshall. Grounded theory for geeks. <http://www.hillside.net/plop/2011/papers/E-13-Hoda.pdf>, 2011.
- [22] Joel Houghton. Improvement in machines for washing table furniture. <http://www.google.com/patents/US7365>, may 1850.
- [23] Jan Humble, Andy Crabtree, Terry Hemmings, Karl-Petter Åkesson, Borianna Kolva, Tom Rodden, and Pär Hansson. "playing with the bits" user-configuration of ubiquitous domestic environments. https://www.sics.se/~kalle/published/UbiComp03/Playing_with_the_Bits.pdf, 2003.
- [24] Andrew Hunt and David Thomas. *The pragmatic programmer : from journeyman to master*. Addison-Wesley, Reading, Mass, 2000.
- [25] Ohad Inbar, Talia Lavie, and Joachim Meyer. Acceptable intrusiveness of online help in mobile devices. <http://doi.acm.org/10.1145/1613858.1613892>, 2009.
- [26] Hiroshi Ishii. Tangible bits: Beyond pixels. <http://doi.acm.org/10.1145/1347390.1347392>, 2008.
- [27] Kathrin Jepsen, Gregor Glass, and Roman Englert. When 'one fits all' does not fit: Study of visualization types for mobile help systems. <http://dl.acm.org/citation.cfm?id=1671011.1671062>, 2009.
- [28] Patrick Jordan. Revising computer documentation for comprehension: ten exercises in method - aided revision. Technical report, Carnegie Mellon University, 1998.

- [29] Azusa Kadomura, Cheng-Yuan Li, Yen-Chang Chen, Hao-Hua Chu, Koji Tsukada, and Itiro Siio. Sensing fork and persuasive game for improving eating behavior. <http://doi.acm.org/10.1145/2494091.2494112>, 2013.
- [30] Thomas Karagiannis, Elias Athanasopoulos, Christos Gkantsidis, and Peter Key. Homemaestro: Order from chaos in home networks. <http://research.microsoft.com/apps/pubs/default.aspx?id=70594>, May 2008.
- [31] Mohammad Khatami, Yaser Heydariyan, and Vahid Bagheri. Investigating the effect of think aloud strategy on reading comprehension ability among efl university students. <http://www.ijllalw.org/finalversion5131.pdf>, 2014.
- [32] Arun Kulshreshth and Joseph J. LaViola, Jr. Exploring the usefulness of finger-based 3d gesture menu selection. <http://doi.acm.org/10.1145/2556288.2557122>, 2014.
- [33] Henry Lieberman, Fabio Paternó, Markus Klann, and Volker Wulf. End-user development - an emerging paradigm. http://hiis.isti.cnr.it/attachments/publications/liebermanpaternoklannwulf_enduserdevelopmentan_emergingparadigm_2006.pdf, 2006.
- [34] Loucas Louca. The syntax or the story behind it?: A usability study of student work with computer-based programming environments in elementary science. <http://doi.acm.org/10.1145/1054972.1055092>, 2005.
- [35] K Louise Barriball and Alison While. Collecting data using a semi-structured interview: a discussion paper. http://www.researchgate.net/profile/Alison_While/publication/234055873_Collecting_data_using_a_semi-structured_interview_a_discussion_paper/links/0fcfd50ea96fa548fd000000.pdf, 1994.
- [36] David J. Malan and Henry H. Leitner. Scratch for budding computer scientists. <http://doi.acm.org/10.1145/1227504.1227388>, mar 2007.
- [37] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. <http://doi.acm.org/10.1145/1868358.1868363>, November 2010.
- [38] Sarah Mennicken and Elaine M. Huang. Hacking the natural habitat: An in-the-wild study of smart homes, their development, and the people who live in them. http://dx.doi.org/10.1007/978-3-642-31205-2_10, 2012.
- [39] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. <http://www.sciencedirect.com/science/article/pii/S1570870512000674>, 2012.
- [40] David G. Novick and Karen Ward. Why don't people read the manual? <http://doi.acm.org/10.1145/1166324.1166329>, 2006.
- [41] Abraham Naftali Oppenheim. *Questionnaire design, interviewing and attitude measurement*. Bloomsbury Publishing, 1992.

- [42] Michael Rietzler, Florian Schaub, Julia Greim, Bjorn Wiedersheim, Marcel Walch, and Michael Weber. homeblox: Introducing process-driven home automation. https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.100/institut/Papers/Prof_Weber/2013-homeSys-homeBLOX.pdf, 2013.
- [43] Larry M. Robinson and Robert L. Berl. What about compliments a followup study on customer complaints and compliments. <http://lilt.ilstu.edu/staylor/csdcdb/articles/1979/Robinson%20et%20al%201979.pdf>, 1980.
- [44] Krystian Samp and Stefan Decker. Supporting menu design with radial layouts. <http://doi.acm.org/10.1145/1842993.1843021>, 2010.
- [45] Sabine Schröder and Martina Ziefle. Making a completely icon-based menu in mobile devices to become true: A user-centered design approach for its development. <http://doi.acm.org/10.1145/1409240.1409256>, 2008.
- [46] Ben Shneiderman, Maxine Cohen, and Steve Jacobs. *Designing the User Interface, 5th edition*. 2010.
- [47] Anders Skaalsveen, Sofie Olsen, and Martin Eskerud. Controlling the home. 2014.
- [48] Percy Spencer. Method of treating foodstuffs. <https://www.google.com/patents/US2495429>, October 1945.
- [49] Julie Steele. *Beautiful visualization : looking at data through the eyes of experts*. O'Reilly, Sebastopol, CA, 2010.
- [50] A. Strauss and J.M. Corbin. Basics of qualitative research: Techniques and procedures for developing grounded theory. <http://books.google.no/books?id=7BGtQgAACAAJ>, 1998.
- [51] Leigh Ann Sudol-DeLyser. Expression of abstraction: Self explanation in code production. <http://doi.acm.org/10.1145/2676723.2677222>, 2015.
- [52] Leila Takayama, Caroline Pantofuro, David Robsen, Bianca Soto, and Michael Barry. Making technology homey: Finding sources of satisfaction and meaning in home automation. <http://doi.acm.org/10.1145/2370216.2370292>, 2012.
- [53] Maarten W Van Someren, Yvonne F Barnard, Jacobijn AC Sandberg, et al. The think aloud method: A practical guide to modelling cognitive processes. http://www.researchgate.net/profile/Maarten_Someren/publication/215439100_The_think_aloud_method_A_practical_guide_to_modelling_cognitive_processes/links/00463517ae0d3c12c0000000.pdf, 1994.
- [54] Jeffrey Wong and Jason I. Hong. Making mashups with marmite: Towards end-user programming for the web. <http://doi.acm.org/10.1145/1240624.1240842>, 2007.
- [55] Feng Xia, Laurence T. Yang, and Alexey Vinel. Internet of things. <http://onlinelibrary.wiley.com/doi/10.1002/dac.2417/abstract>.

Appendix A

Form of Consent - Future Workshop

Form Of Consent - Future Workshop

Hello and thank you for attending this Future Workshop.

By participating, you consent to the gathering of data regarding your age, education, relevant background and past involvement with home automation. You also consent to having your picture taken, which could be included in a final report and/or subsequent papers. This session will also be filmed, but the video will not be viewed by anyone but Martin Eskerud, and will be transcribed, describing both your actions and speech.

The purpose of this test is to gather ideas and feedback from you, and is **not** a test of the participants. I encourage all testers to think aloud.

By signing this document, you agree to the terms detailed in this document, and you will participate in this future workshop. Your name will not appear anywhere in public material published as a result of this work, but your picture could be included (without names).

Date

Signature

Appendix B

First User Test

Training Tasks

1. Hent frem toalett-rommet.
2. Øk styrken på lyset på toalettet.
3. Skru ned varme på gulvet på toalettet.
4. Skru på oppvaskmaskinen
5. Sjekk om stekeplatene står på, skru de evt. av.
6. Koble kjøkkenlyset sammen med stuelyset.

Oppgaver for relationships

1. Du ønsker at baderomslyset skal skru seg på når du skrur på lyset på soverommet. Lag en regel/connection for dette.
2. du ønsker å komme hjem til varm stue kl. 18.00 Opprett en regel for dette.
3. Du har fått flexitid! Nå er du hjemme en time tidligere, men du ønsker fortsatt å komme hjem til en varm stue. Endre regelen du akkurat lagde.
4. Du har fått deg en hund som sover på badet, den liker ikke lys om morgenen. Kan du fjerne den regelen du lagde i stad?
5. Når du skrur av TV'n ønsker du at lyset også skal skru seg av, kan du lage en regel for dette?
6. Kan du skru av de connections du har opprettet?

Interview

1. Var det noe du syntes fungerte bra i interfacet? Hvis ja, hva?
2. Var det noe i oppgavene du gjennomførte som oppførte seg annerledes enn du forventet?

3. Har du noen ideer om ting som kunne blitt gjort annerledes eller forbedringer til grensesnittet?
4. Hva skal til for at du skal kunne ta i bruk et slikt system hjemme?
5. Hva var helhetsinntrykket ditt til konseptet og hvordan den lot deg løse oppgaver?
6. Hvordan tror du ikke-tekniske personer ville håndtert de oppgavene du har løst i dag? Eller familie etc.

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree					Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	1	2	3	4	5	

Appendix C

Final User Test

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6. I thought there was too much inconsistency in this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently					X
2. I found the system unnecessarily complex		X			
3. I thought the system was easy to use				X	
4. I think that I would need the support of a technical person to be able to use this system	X				
5. I found the various functions in this system were well integrated				X	
6. I thought there was too much inconsistency in this system	X				
7. I would imagine that most people would learn to use this system very quickly			X		
8. I found the system very cumbersome to use	X				
9. I felt very confident using the system				X	
10. I needed to learn a lot of things before I could get going with this system	X				

unge mennesker
ja.

System Usability Scale

© Digital Equipment Corporation, 1986

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

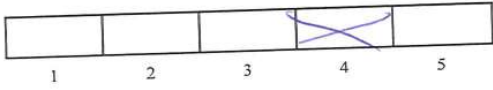
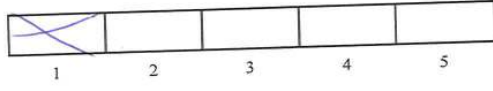
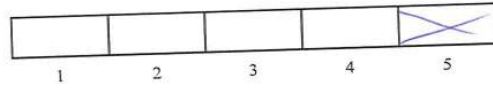
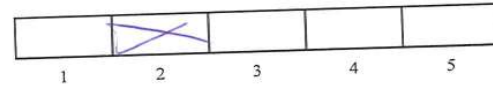
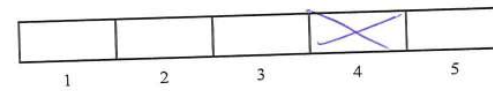
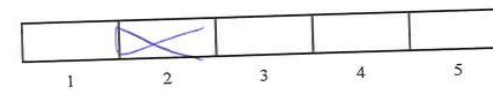
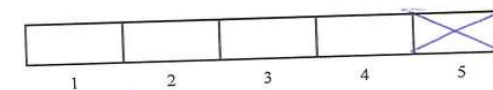
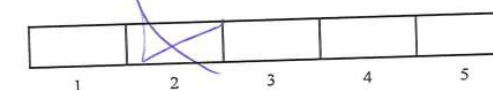
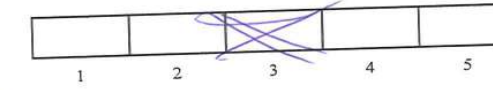
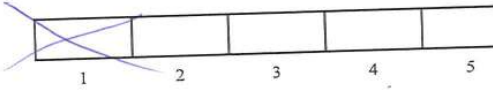
System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree						Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree	Strongly agree
1. I think that I would like to use this system frequently		
2. I found the system unnecessarily complex		
3. I thought the system was easy to use		
4. I think that I would need the support of a technical person to be able to use this system		
5. I found the various functions in this system were well integrated		
6. I thought there was too much inconsistency in this system		
7. I would imagine that most people would learn to use this system very quickly		
8. I found the system very cumbersome to use		
9. I felt very confident using the system		
10. I needed to learn a lot of things before I could get going		

System Usability Scale

© Digital Equipment Corporation, 1986

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. I found the system unnecessarily complex	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
6. I thought there was too much inconsistency in this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	1	2	3	4	5
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	1	2	3	4	5

System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
8. I found the system very cumbersome to use	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Training Tasks

1. Hent frem toalett-rommet.
2. Skru på oppvaskmaskinen
3. Sjekk om stekeplatene på komfyren står på, skru de evt. av.
4. Koble kjøkkenlyset sammen med stuelyset. (high fidelity prototype)

Oppgaver for relationships

1. Du har montert en bevegelsessensor i stua, du ønsker at dersom den er på, skal lyset skrus på. Opprett logikk for dette ved relationship-editoren
2. Du ønsker å komme hjem til varm stue kl. 18.00 Opprett en regel for dette.
3. Når du skrur av TV'n ønsker du at lyset også skal skru seg av, kan du lage en regel for dette?

Interview

1. Var det noe du syntes fungerte bra i interfacet? Hvis ja, hva?
2. Var det noe i oppgavene du gjennomførte som oppførte seg annerledes enn du forventet?
3. Tenkte du noe på skyggen rundt romnodene? Hva tror du dette betyr? //Forklar - hent feedback.
4. Hva syns du om å inkludere sensorer i interfacet på den måten det ble gjort?
5. Forslag til forbedringer?
6. Var det tydelig å se hvilke ting som var av og på i interfacet?
7. Hvordan likte du å definere programlogikk på denne måten? Var noe vanskelig?
8. Var det logisk å gjøre de oppgavene vi gjorde i dag? Kunne du se for deg at dette var en vanlig dag med denne funksjonaliteten?
9. Hva skal til for at du skal kunne ta i bruk et slikt system hjemme, forutsatt at det var ferdig og fungerende?
10. Hva var hellhetsinntrykket ditt til konseptet og hvordan den lot deg løse oppgaver?

C.1 Tags Produced from Open Coding

experienced user	mobile	remote control	relationships
too advanced	understandable	graph interface	Ambiguous
intuitive	device node icons	exploration	non intuitive
surprised	correct guess	device states	Re-usable code
misunderstanding	open minded	not happy	Idea/Suggestion
utilities	EUP	pronouncing logic	error
clarifying	information overload	Correct	reassuring
first try	tried alone	failed	quick
helping	clock	concerned	average user
positive	negative	target audience	reflected
restrictions	problem	shadow room node	sensors
icon	feedback	unfamiliar	learning curve
tasks	new usages	improvements	centigrades
problem solving	price	possible problems	room nodes
sellable	think aloud	real world	timer
alternative solution	safety	function	alarm
automation	seeking help	expansion	inaccurate touch
interval	global	delay	less and more than
conditional timer	tutorial	logic	fun
then clause	clean design	shortcut	toolbox
comprehension/understanding	small icons	relationship arrows	off-canvas layout
potentially cluttered	navigation	adaptability	jigsaw

