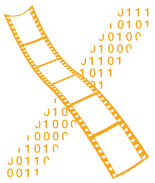


Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Towards a Flow-Based Embedded Programming Environment for School Children

Master's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Ilja Golland

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr.-Ing. Ulrik Schroeder

Registration date: 28.09.2018
Submission date: 28.09.2018

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

_____

_____

_____

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift

Contents

List of Figures

Abstract

Visual programming has been an active field of research for the last years. Research and current movements like the "Maker Movement" suggest that the gap between developers and consumers decreases in some domains. However, young people and novices in technical fields still struggle with introductory barriers regarding programming. In this scope, visual programming as a paradigm provides potential to help them overcome this problem and program projects on their own.

In our work we try to lower this barrier from the perspective of physical computing. Compared to abstract programming, working with sensors and microcontrollers allows direct interaction with the real world and can help to engage young people. We developed a high fidelity prototype of a touch-based app, which provides a basic set of logical functions in order to connect input and output peripherals, by using flow-based programming.

Our studies provide results that young students (aged from 12-15) with no prior programming experience felt confident exploring the different functionalities and solving tasks. Although some participants struggled in the beginning, they were all positive that they could have solved the tasks on their own given more time. We are confident that these results are beneficial for further iterations and systems using visual programming on touch-based devices help young people learn working with microcontrollers and sensors.

Überblick

Seit einigen Jahren ist Visual Programming ein aktives Forschungsgebiet. Forschung und derzeitige Bewegungen wie das "Maker Movement" legen nahe, dass die Schere zwischen Entwickelnden und Konsumierenden kleiner wird in manchen Bereichen. Allerdings haben Jüngere und Neulinge in technischen Bereichen immer noch Probleme anfängliche Barrieren bezüglich des Programmierens zu überwinden. In diesem Rahmen bietet Visual Programming ein Potential, diese Menschen bei Überwindungen solcher Barrieren zu unterstützen und selber Projekte zu programmieren.

In unserer Arbeit versuchen wir diese Barriere mittels der Perspektive von Physical Computing zu senken. Verglichen mit abstraktem Programmieren erlaubt das Arbeiten mit Sensoren und Microcontrollern direkte Interaktion mit der realen Welt und kann dabei helfen junge Menschen dafür zu begeistern. Wir haben einen realitätsgetreuen Prototypen einer touch-basierten App entwickelt, welcher mittels flow-based programming eine Grundmenge an logischen Funktionen zur Verfüng stellt um Eingangs- und Ausgangsperiperie zu verbinden.

Die Ergebnisse unserer Studien legen nahe, dass sich junge Schulkinder (im Alter von 12-15) ohne bisherige Programmiererfahrungen sicher fühlten als sie die unterschiedlichen Funktionalitäten erforscht und die Aufgaben gelöst hatten. Obwohl einige Teilnehmende anfangs Schwierigkeiten hatten, waren sie alle zuversichtlich dass sie die Aufgaben alleine lösen könnten, wenn sie mehr Zeit gehabt hätten. Wir sind sicher, dass diese Ergebnisse für weitere Iterationen und Systeme, die Visual Programming auf touch-basierten Geräten benutzen um jüngeren Menschen zu helfen mit Mikrocontrollern und Sensoren zu arbeiten, förderlich sind.

Acknowledgements

Hereby I would like to acknowledge and thank all the people who supported this work, without them this would not have been possible.

Thanks to my supervisor Simon Völker for his valuable feedback throughout the whole work and by spending time to provide advice when needed, but also for being patient and giving me th freedom for this thesis. And Nadine Bergner for the collaboration and for her feedback, and also for helping us by allowing participants of the school laboratory to partake at our study.

In this matter I want to thank all the pupils for taking time and the willingness to share valuable feedback for our work.

Furthermore I appreciate Aaron Krämer's help when it came to technical issues and also for recommending Unity. Hereby I also want to thank Stefan Greiß for his help when it came to Unity related issues. I am also grateful to Kerstin Kreutz and Oliver Nowak for helping with advice on technical issues and approaches, and further appreciate Kerstin Kreutz' help in particular for proofreading this thesis.

Thanks to Prof. Dr. Jan Borchers and Prof. Dr.-Ing. Ulrik Schroeder for providing me with a working environment which provided everything I needed to work on this thesis and conduct the studies.

A special thanks to my family who always provided support when needed.

Thank you!

Conventions

Throughout this thesis we use the following conventions.

Text conventions

The whole thesis is written in Canadian English and in first person plural form.

We also use the gender-neutral singular pronoun *they* when the gender of a person is not important for the context.

Chapter 1

Introduction

Over the last decades the development of software underwent constant changes. This happened and is still happening in the scope of the different programming paradigms and extending the outreach to everyday users and not only professional software developers. Consumers who use software are supposed to also modify and alter software. Therefore, the distinction between developers and consumers becomes more and more blurry. In his work, Lieberman [2006] coined the term "End-user Development", which describes the evolution from "making systems *easy to use* [...] to making systems that are *easy to develop*". He further emphasizes the importance of "users who do not have background in programming to develop or modify their own applications, with the ultimate aim of empowering people to flexibly employ advanced information and communication technologies."

The gap between consumers and developers became more narrow over time

At the same time, empowering and enabling people is also further supported by Paternò in his work [Paternò, 2013], as well as by the "Maker Movement" which has been growing over the last years. Dougherty [2012], founder of "Make" and creator of "Maker Faire", argues that everyone is a maker because we all create and make things, regardless if it's "fix[ing] your own car [...] or improv[ing] your home or mak[ing] your own clothes". However, today the movement and also the so called Maker Culture refers mostly to the idea of everyone being able to make

Movemenets such as the Maker Movement try to enable more people to create and develop

something at home using technologies such as 3D printing, laser cutting, CNC milling, etc. without having an in-depth knowledge in the technical field. Or, as Fischer [2002] puts it: "It is also a mistake to assume that being a consumer or being a designer would be a binary choice: it is rather a continuum".

Physical computing can serve as an introductory gateway into programming for consumers

However, the consumer market for these technologies mostly consists of consumers who do not want to spend unnecessary time and effort on understanding how these work. But at the same time there are those, who want to gain a higher degree customizing their own creations. A foundation for this, which can act as a good starting point, is the field of "Physical Computing". Originally defined in [O'Sullivan and Igoe, 2004], Booth et al. [2016] summarized it as "Physical computing integrates computing with the physical world, often in the form of electronic devices or systems that interact with the environment via sensors and actuators. These devices can take input from the world, through sensors that measure aspects of the environment, such as temperature, proximity, or light, and respond in some way, for example, through sound, motion or vibration". They further state that platforms "like Arduino aim to lower the barriers [...] to this type of activity, but creating electronic circuits and programming them still requires some knowledge and skill, and troubleshooting physical computing issues can be tricky".

The Arduino and the Raspberry Pi are devices for first attempts but come with potential downsides for novices

The aspects of empowering people and the emerging field of physical computing were part of our motivation for an approach which unifies these. As Drew et al. [2016] phrase it, "[t]he recent proliferation of easy to use electronic components and toolkits has introduced a large number of novices to designing and building electronic projects". When working with such electronic devices as mentioned above, one of the devices novices can try first is an Arduino[1] or Raspberry Pi[2]. They are then oftentimes used together with a breadboard, which is a plastic board with arrays of holes which can be used to plug in cables, sensors, etc. Soldering is not needed, making it easily reusable and suited for quick alterations in the setup. However, while

[1] https://www.arduino.cc
[2] https://www.raspberrypi.org

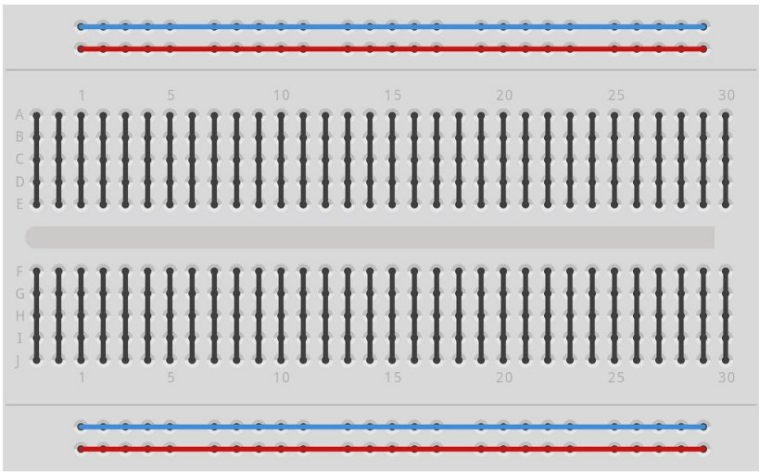


Figure 1.1: The horizontal and vertical connections on a breadboard visually marked

the arrangement of the hole arrays is in rectangular shapes and simple, the conductor paths are - on the most common breadboard - inside of the breadboard and there are no visual clues which show where they are. Figure 1.1 shows these underlying horizontal and vertical conductor paths.

This makes it clear that the usage of the breadboard is easy due to the holes but is also not very intuitive regarding the paths as you only know how they are aligned by looking it up. For novice users this creates a higher barrier for understanding. As Drew et al. [2016] put it, "breadboarded circuits are prone to a host of common issues such as incorrect component placement, faulty connections, and power management problems. The visual complexity of breadboarded circuits makes it even harder to probe and debug circuits".

Issue 1 *Conductor paths are not visible*

This is further enhanced due to breadboards serving as a platform for both input and output. With such on the same platform without a visual distinction, this again makes it not very clear for novices what is used for reading information and what is used for the output.

Issue 2 *Input and output takes place on the same platform*

As we explained above, the usual set up for working with physical components usually consists of having a microcontroller, one or multiple breadboards and a computer to write the code. This creates a disconnect between the area for manipulating and altering the physical set up (the microcontroller and the breadboard with the physical components) and the area for writing the logic (the computer). People then have to switch between those areas every time they change something in one area and have to readjust to the changes in the other area. Changing the physical arrangement of components might require changes in the code and results of those changes can only be seen on the other area. This issue is even more problematic since it creates another disconnect between the interaction and the results as one does not directly manipulate the physical components.

Issue 3 *The need for several components and missing direct feedback and direct manipulation*

Programming has always been the way of how people tell machines what they are supposed to do. While the mental model of the steps is oftentimes naturally formulated in the mind of the programmer, it still needs to be turned into program code. Although there has been an increase in more and more different programming languages and approaches which make code less abstract, code is still text based most of the time. They still require understanding of different syntax' and other specific properties for each programming language. As a consequence students "easily lose their interests in programming, for example, when they need to learn the rigid grammar or when they encounter minor syntactical errors" [Kato, 2010, Maloney et al., 2008]. For this purpose Computer science programs teach programming paradigms and abstract thinking which helps understanding programming in general in order to apply it to the concrete programming language they will use in practice. However, for young people without any of this, this level of abstraction forms another entrance barrier for those without education and sense for abstract thinking.

Figure 1.2: A rough draft of the system consisting of the tablet in the middle and breadboards on both sides

Issue 4 *Hurdle of abstract text based programming*

Our system targets young and novice students without technical background and offers an environment for them to learn and explore physical computing and the fundamental logic for it. In this scope we also try to approach the issues discussed above to ensure this main aspect.

It consists of a smart tablet device and two breadboards on the left and the right side. While our app runs in multiple platforms, for our work the tablet we used was an iPad Air 2. A sketch of this approach can be seen in Figure 1.2. For the metaphor, the breadboards are supposed to be connected to the tablet in the middle. However, due to technical reasons they instead will be connected to an underlying Arduino microcontroller which provides an interface to establish the communication between the breadboards and the tablet.

1. The main purpose of splitting the breadboard into two separate units is to have one for input data and one for output data. This mainly addresses issue 2.
2. In western countries we are used to reading from left to right. Hence we tend to also work in the same direction. Placing one breadboard for input data on the left side and one for output data on the right side supports the visual flow.

The tablet device will run an app, which provides a programming environment for the user. For the reasons discussed above, the paradigm for this environment will be visual based programming. Our work focuses on the concept and design of the correspondent user interface for this app. The idea of using visual programming also directly solves issue 1 as the connections between the physical components are created by the user and they do not need to rely on the underlying hidden conductor paths in the breadboards. At the same time it helps addressing issue 4 since no explicit knowledge about programming languages is needed. We will discuss this in more detail in Chapter 4 and 5

Our flow based approach avoids conductor paths and knowledge about programming

Aside from these issues which are mainly related to the breadboard, our work also attempts to not only teach kids about physical computing in order to understand the basic concepts but to also provide the opportunity to explore. In their work, Resnick and Silverman introduced the concept of *Low Floor, Wide Walls, and a High Ceiling*, which is based on the original work by Papert [1980]. A system satisfying these properties allows novices to easily get used to the system and allows "experts to work on increasingly sophisticated projects". While the latter is not that important for our work, the third property suggests that the system offers "a wide range of different explorations", which supports our idea of a playground for the students. Hence our work also tries to fulfill these properties and apply this concept.

A concept by Resnick and Silverman would allow a playground and exploration for different users

Chapter 2

Related work

During our literature review we analyzed the current body of research and found results of different areas and aspects. It was evident that the area of *Visual Programming* has been a research area for a while now and still is. This also holds for time learning or educating people regarding programming, which has also been an active research area for a long time now. Since our work focuses mainly on the software and how it can help young students explore and learn physical computing, this section will mainly cover literature for these areas.

2.1 Programming without code

As we briefly explained in Chapter 1 "Introduction", programming provides different properties, which create a high introductory barrier for novices, who have no technical background. This is supported by literature reviews done by Papadakis et al. among others. However, in our literature review, developers were also considered to possibly benefit from programming without explicitly writing code [Whitley and Blackwell, 2001].

Due to the abstraction from natural spoken language, there have been different approaches on how to loosen the dis-

Visual programming and Novice Programming Environments as new approaches for beginners

connect between natural and formal languages such as programming languages. One of the recent and more known approaches is visual-based programming or - in short - visual programming. While we could not find studies which provide evidence that visual programming in general is a potential alternative to text based programming, many studies worked with specific environments and provided promising results. Such environments for novice students are called Novice Programming Environments [Papadakis et al., 2014], which "make the programming procedure more friendly and pleasing". More specifically, for the last years the NPE Scratch [Resnick et al., 2009] has not only been used as a basis for studies concerning visual programming but it also served as an idea to build on. According to Resnick "students shift from media consumers to media producers, creating their own interactive stories, games, and animations —- then sharing their creations on the Web" with the aid of Scratch.

Visual programming focuses more on the creativity instead of formalities such as syntax and the need to memorize commands

Rizvi et al. also describe Scratch as "a media-rich programming language and environment that is free from the distraction of syntax while supporting the implementation of complex projects, such as games. It thus enables focus to be placed on algorithm development and the design of projects." At the same time, Federici also asked if "we need really to teach computational skills by using standard programming environments such as [...] Netbeans or Eclipse" as NPEs "focus on the creative potential behind computation by 'lowering the floor' for entry and by allowing students to get interesting results in a shorter time". This and the opinion of many educators that "block languages are too toy-like" lead to their work, which showed that students thought "it can help them in computer programming, especially in avoiding syntax errors and in remembering well the template of the single [...] commands". They also seemed to like that the available commands were all visible, reducing the necessity to know and remember all of them and therefore reducing the cognitive effort.

Several studies show results from using Scratch in introductory courses and suggest that it helped students at-risk to succeed better in exams [Rizvi et al., 2011] (using Scratch 66% CS majors passed the midterm compared to 56% who

failed it the year before) and that it also motivated young people to get into programming [Maloney et al., 2008]. Especially the former work showed interesting results as youths were given the possibility to create media in a Computer Clubhouse using different software like Adobe Photoshop or Bryce. But instead, "Scratch grew to be the most widely used design software available at the Clubhouse" and "was more heavily used than any other media-creation tool, including Microsoft Word". However, the responses from the youths also showed results, which can be both advantageous and disadvantageous. When asked *If Scratch had to be something not on the computer, what would it be?*, "the most common response was 'paper' or 'a sketchbook' because Scratch allows you to 'do anything that you want with it, just like paper' (n = 8)". This suggests that for the most part, the youths did not grasp the connection between Scratch and (computer) programming, as is also further supported by the fact that "[m]ost youth didn't identify scripting in Scratch as a form of programming. In general, when youth were asked, 'What is computer programming to you?' they responded: 'Computer programming? I do not have a clue [what that is]!'" This disconnect can help overcome the barrier of programming in cases when someone has negatively biased associations with programming and might be reluctant. At the same time this missing connection is also a potential threat to the validity since it becomes harder to later apply the gained knowledge to other programming concepts and it can also happen to not solve the associations.

Scratch showed that it both helped students pass exams and also got young students into programming

Most people associated Scratch rather with paper or a sketchbook instead of a programming environment

Lewis compared the performance of students with Scratch to already promising results with Logo, a text based programming language for novice students prior to Scratch. While Logo already showed good results "supporting student development of confidence, interest in computer programming, and understanding of the loop construct", Scratch provided even better results.

Compared with another environment, Scratched provided even better results

While our primary target group are young students without technical background, it is still interesting to see how professional developers perceive visual programming compared to classical text based environments. Whitley and Blackwell studied this with 227 participants and found out

Also professional developers liked the visual language feature in one of their IDEs

that "respondents rated the value of LabVIEW's visuals significantly higher than all other LabVIEW features rated in this survey" and "felt that LabVIEW is a very effective tool." However, some participants "did rate text significantly higher than LabVIEW for expressing conditional logic". In Section 5.1.1 we describe how our system also provides conditional logic.

The metaphor of Scratch was also extended in other works and showed positive results regarding learning effects

As we mentioned above, Scratch was not only used in its original version but also as a basis. Hu et al. used it in their work and extended it using BYOB[1] which allows to create own blocks in Scratch. They used it in order to create a visual notation for *plans* and *goals* "following the metaphor of a network of plans that communicate using dataflow". Their results provide evidence that Scratch helped students to understand programming better. However, although the authors suggest their results provide evidence for the success of goals and plans, it is not clear if that is the case as they also point out a threat to their validity being that they "considered the new approach as a package". Nonetheless the concept of plans and goals is an interesting aspect considering a flow based paradigm with visual based programming.

A study involving App Inventor for Android suggests visual programming can increase motivation and interest of students regarding CS

Another visual programming environment, which is inspired by Scratch, is App Inventor for Android (AIA). In their work, Ahmad and Gestwicki used AIA for an introductory CS course together with an alternative teaching methodology in order to see if students were able to perform better in classes. With interviews, mind maps, and submitted course work by the students, the results suggest that the students found it to be a positive experience and this was strongly attributed to the use of AIA. More precisely, an increase in the "students' motivation, performance, and attitude" could be seen. More interestingly, students "no longer acquainted [the course] with programming alone", appreciated other factors, and subsequently found CS in general "inviting, challenging, and interesting". Other works also support these results that visual programming helped students engage with different topics, such as "privacy and scale in their own terms" and also "privacy and anonymity implications" [Dasgupta and

[1] http://byob.berkeley.edu

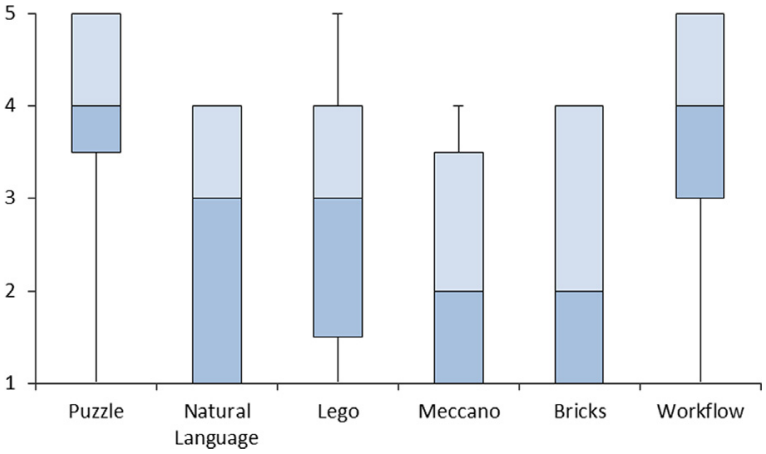


Figure 2.1: Evaluation of a metaphor questionnaire (image taken from Danado and Paternò [2012]

Resnick, 2014]. Even though our target group is younger and they are no CS students, this supports our aim to use visual programming as it potentially lowers the introductory barrier for practical usage and also negative associations with programming.

The presented works make use of the visual programming or flow based programming paradigm, but all of them run mainly on a desktop system such as a desktop PC or a notebook. Hence, the interaction is done via mouse and keyboard. The only work we found, which was using touch input, was done by Danado and Paternò [2012], who developed a system which users without programming experience can use to develop applications on touch-based mobile phones. As Figure 2.2 shows, Puzzle runs on a mobile phone and uses a jigsaw metaphor to create applications. One of their main goals regarding the UI was "to reduce the cognitive effort of end users without an IT background through adequate metaphors and interaction techniques", which is also one of our motivations. Furthermore, their evaluation also followed a similar approach (see Chapter 5 "High Fidelity Prototype") as their way of evaluating the UI was to give no "tutorials or learning steps [...] to end users" to see "if the UI was able to guide them through the creation of a mobile application".

Puzzle is a system running on mobile systems which allows development of apps with no programming background

Figure 2.2: The three different activities in Puzzle: The main screen, the authoring tool, and the execution environment (image taken from Danado and Paternò [2012]

Their work suggests that a workflow and puzzle metaphor was rated as most relevant

Due to their profound evaluation and coverage of different aspects of the system the results provided interesting insight. Initially they evaluated different possible metaphors for programming paradigms using a "five point Likert scale (not relevant=1, residual=2, neutral=3, relevant=4, very relevant=5)" [Danado and Paternò, 2014]. As Figure 2.1 shows, the *Puzzle* and *Workflow* metaphors were chosen as the most relevant. As we discuss in later chapters, our metaphor is closer to a *Workflow*. Their further results suggested that the "jigsaw piece metaphor was easily understood and applied" and the "drag-and-drop interaction technique[...] proved to be effective". Furthermore "the results regarding the UI were positive in terms of easiness to learn, efficiency, effectiveness, easiness to remember, and number of errors". While these are positive results also further support our own approach with our system targeted at the physical computing domain, their target group was different since their group were participants without programming knowledge, similar to ours, but in the first study the participants' ages ranged from 26 to 35 and in the second study from 31 to 59.

Our system is an app which runs on multiple platforms but focuses on touch as the main interaction. This method allows both direct manipulation and a minimum of required devices and peripherals. Past research has suggested that direct manipulation provides many different advantages such as "continuous representation of the object of interest" and "[p]hysical actions or labelled button presses in-

stead of complex syntax" [Shneiderman, 1982]. With these "[n]ovices can learn basic functionality quickly", they can "immediately see if their actions are furthering their goals", and "have reduced anxiety because the system is comprehensible and because actions are so easily reversible".

Learning by doing and creating own things also drives the Maker Movement

Dougherty further supports this in his writing and says that "psychologist and education reformer John Dewey extolled the virtues of learning by doing, and contemporary science of the brain confirms the importance of tactical engagement and of using our hands in the learning process". Furthermore "[w]hen you're making something, the object you create is a demonstration of what you've learned to do, thus you are providing evidence of your learning". While this regards the Maker Movement in general, it also matches the advantage of using touch and direct manipulation.

2.2 Learning Theory

Constructionism Theory suggests that people learn well when creating things in groups and sharing them

The most important foundation on which Resnick and Silverman [2005] based their work Scratch on is the *Constructionism Theory* by Papert. It is formed from the idea when people practically design and build something on their own, as Millner [2012] summarized it in their work, "people learn particularly well while actively engaging in constructing artifacts to share with and be critiqued by others". Their work is also relevant in particular since one of their research questions was how their system can "transform how young people approach design – enabling them to explore design strategies and engineering ideas?". The artifact they proposed offers young people to combine digital sensors with analog material in order to control media and enable them "to act as 'physical computing' designers". Their programming paradigm was also based on Scratch. Through their studies they saw success in engaging young people in getting creative and transform personal ideas into real concepts. However, while it shows potential for the use of physical computing in combination with visual programming, their system was used in groups, where people had experience and helped others who did not.

Previous work combined the visual paradigm with physical computing and saw positive results with engaging young people

Creating projects also engages people in computational thinking

And as Dasgupta and Resnick [2014] mention "the process of constructing these projects offers opportunities to engage in not only computational thinking, but also in 'thinking about thinking' (e.g., any exercise in debugging code becomes an examination of one's own thought process)". These are important properties for our work and our goal which is to offer students the chance to learn on their own by exploring and constructing their own projects.

Carroll and Rosson argue that learning with real-world cases engages students and helps them learn

Case-based learning can also "engage the student in the drama of a real situation" [Carroll and Rosson, 2005] and "provide guidance and encouragement for user action by describing specific activities, events, and problems from real world practice". As our system provides an interface for young students to solve simple but potentially real-world tasks, these works support our approached paradigm. In their work, cases are described as "narrative descriptions of a [...] problem, drawn from the real world of

professional practice" and they can "emphasize the contingency of real practice to novices". Although their work appears to be a theoretical framework for usability engineers working with cases in the HCI domain, it still suggests that using specific cases for learning has potential for novices.

2.3 Physical computing

Splish as an example for visual programming combined with physical computing

There is also research regarding visual programming specifically targeting physical computing. Splish by Kato is a visual programming environment for students who can develop and run their programs on an Arduino. This environment was aimed at non-specialists who "can quickly develop programs which interact with microcontrollers such as Arduino to control sensors and actuators embedded in physical objects in our daily lives". Although this technical aim is similar to ours, our focus of the target group is not only the lack of knowledge but also the age, since we also aim for young students.

An efficient system aimed at non-specialists but lacking empirical studies

Figure 2.3 shows both the environment of Splish and an exemplary program which depicts an actual program. Their metaphor for both the environment and the flow based construction of programs provides a foundation for our own system, as we show in Chapter 4 "Low Fidelity Prototype". Furthermore they explicitly state that, although according to them, visual programming "accelerates the physical programming experiences because the program can be built visually, the program flow can be understood graphically", their system requires an empirical study "to further improve the usability". Our work tries to address this as we conducted a study with young students using our own system (see Chapter 6 "Study").

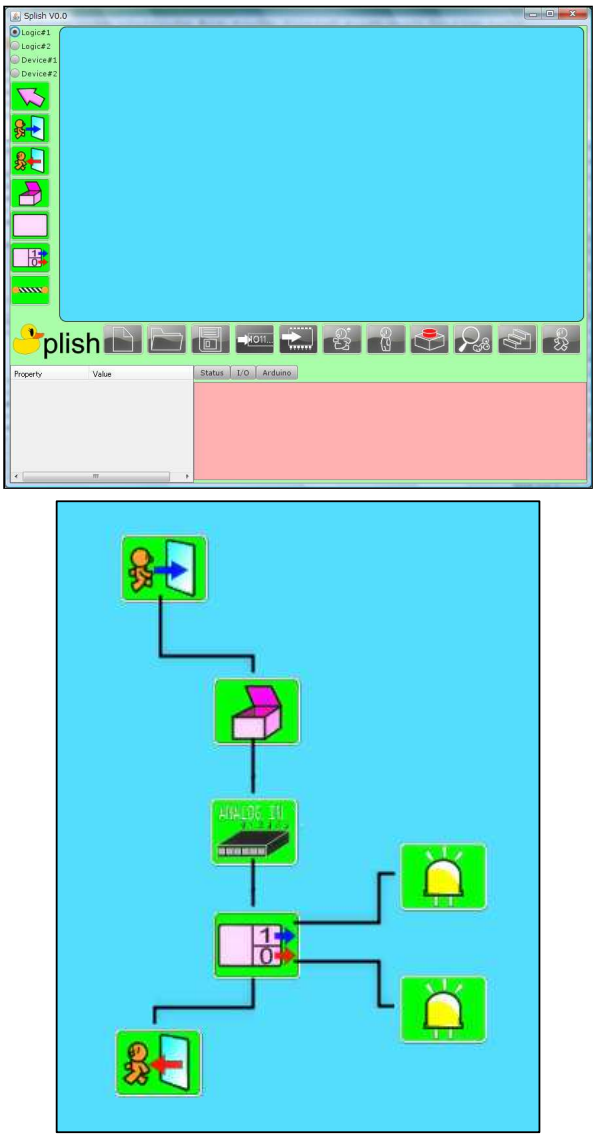


Figure 2.3: The UI for Splish and an exemplary program with read values, analog input, and a "selection block" making a decision (images taken from Kato [2010]

Lawris uses icons and jigsaw pieces but lacks concrete target group and study, and no real-time flow of signals

Arakliotis et al. aim their system at young primary school students. Their approach is rule- and web-based, but contrary to Splish, it uses the jigsaw metaphor of Scratch (see Figure2.4). According to the authors this tackles issues of systems such as Scratch and Splish as they do not think those are appropriate for young primary students. Al-

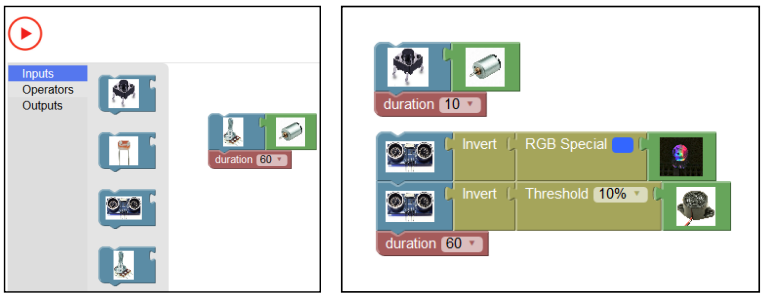


Figure 2.4: LAWRIS system combines icons for elements and jigsaw pieces like Scratch: The left picture shows the play button with the commands palette and a small program to control a motor with a potentiometer for 60 seconds. The right picture shows a program to control an automatic door and its alarm (images taken from Arakliotis et al. [2016]

though their combination of icons and jigsaw pieces looks promising, their work does not specify *young students* more precisely and also lacks empirical data of studies with this system. Furthermore, similar to Modkit [Millner and Baafi, 2011], their system does not provide a real-time handling of the data flow as users put together pieces and then click the play button to check the results. Our idea is based on a constant running system with immediate feedback while interacting with the elements and logic. This is supported by works such as Lieber et al. [2014], whose results suggest that live information for programming interfaces helps "by proactively displaying information about code that is normally hidden" [Drew et al., 2016].

There is a research corpus which suggests that live information helps with programming interfaces

In Millner [2012], the authors introduce the Hook-ups System, which is aimed at young people and based on constructionism (see Section 2.2). It features a combination of physical media, such as sensors, and digital media like images and sounds and use these to create tangible interfaces. Their motivation was based on observations that "too many [...] children do not conceive of computers as being integral tools for the activities they engage in based upon their interests" and that computers as they are used in classes, do not facilitate the creativity of young people. The process of a Hook-ups System is depicted in Fig-

Hook-ups Systems let young people combine physical components with sensors to rebuild their own ideas in new ways

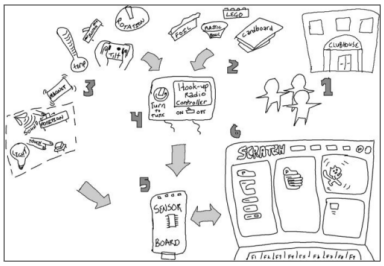


Figure 2.5: The idea of the Hook-ups System: Young students connect physical components with sensors, connect these with a Scratch Board and then use a computer with Scratch to program the control of the sensors

ure 2.5. In summary, young students combine physical components with sensors, which is then called a Hook-up. These are then connected to Scratch Sensor Boards[2], which are plugged "into a computer running a programming environment called Scratch". Their work presents two exemplary projects to illustrate the results as there was no formal or empirical study. According to them, they showed that this system helped young people engage in combining their everyday interests with physical computing and also connect to the "participants' personally meaningful materials" and therefore also help express themselves. In one of the projects they also had to break down a toy and see which sensors they could use to rebuild something new. Although in their work, participants worked together and not all on their own, and the setting and environment was different, the results still indicate that physical computing can help young people engage in new ideas and explore.

[2]https://en.scratch-wiki.info/wiki/PicoBoard

Chapter 3

First Observation

This work has been done in cooperation with the "Learning Technologies Research Group" of the Computer Science Department 9 at RWTH Aachen. This chair holds a school laboratory called "InfoSphere"[1], which they describe as "An Extracurricular Place of Learning for School Students of All Ages". Throughout the year classes with young students come to work with and understand technical concepts and also create applications using methods of computer science. We were told that the age groups and the projects they work with in classes vary a lot, but some classes are about developing applications using App Inventor for Android and other classes work with Arduino microcontrollers.

A school laboratory at our university teaches pupils how to develop applications using App Inventor for Android

For these reasons we scheduled a day to observe the pupils in a class in which they work with AIA. We wanted to see how they use it and what they think of it, as it might have given information on how visual programming is perceived. Furthermore and more importantly it could have shown what causes confusion and problems and when do they get stuck.

Unfortunately we noticed some negative conditions. Pupils in the class were working in groups of about four to five people and needed to solve some exercises in AIA. Al-

[1] http://schuelerlabor.informatik.rwth-aachen.de/en

Since the pupils followed step by step instructions, we could not get any insights into what problems they encounter and how intuitive the app was

though these exercises looked interesting as the end result would be an app to take pictures and draw over them, the learning material the pupils got, did not just explain AIA but guided them through the whole process step by step[2]. This condition not only minimizes the chances of pupils encountering issues, it also lowers the opportunity to explore and create one's own solutions in order to solve an exercise. It is important to note that one of the reasons for this is that AIA is only available in English and most pupils have problems understanding all the terms. The given material explains some aspects of computer programming such as the purpose of variables, but this further suggests, that AIA is hard to comprehend for young people without prior experience with programming. This was also affirmed by the research assistants who held the lab. Due to these observations we could not ask the pupils when they got stuck and what caused problems for them. However, we were also told by the research assistants that many pupils have problems understanding logical constructs such as AND, OR, etc. and also what inequalities mean. In total, however, it was very hard for them to tell what pupils learned through these classes and if they only solved the exercises because they followed the steps or if they also understood what they did while solving it.

Due to the cooperation and because these pupils were so diverse regarding age and prior experience with programming, pupils of these classes were also participants of our study (see Chapter 6 "Study").

[2] http://schuelerlabor.informatik.rwth-aachen.de/en/modulmaterialien/erste-app

Chapter 4

Low Fidelity Prototype

In the very first phase of the prototyping process, the low fidelity prototype sets the base for further iterations. However, even the low fidelity prototype goes through some iterations depending on the thought processes and the resonance. For our case, we used the current research body and current practical applications in order to conceptualize a very first draft. We then used this draft to check how possible users react to it and what they think. In the following we will describe the process of designing this draft and how it was evaluated.

4.1 Design

Our focus for this prototype lied on how technically simple and how easy it should be for the participants to use it. By technically simple we mean that the practical design and construction of the prototype should be kept simple. At the same time, it should also be as close to using an actual touch display and be able to demonstrate our concept. Meaning that we did not want to use a simulation on a computer with a keyboard or mouse. Instead, the user should interact with the prototype via touch interaction in order to increase the internal validity and exclude potential intervening factors. While this is a standard requirement for low fidelity

The focus of the low fidelity prototype was to demonstrate the very basic idea and allow open thinking regarding ideas and critique

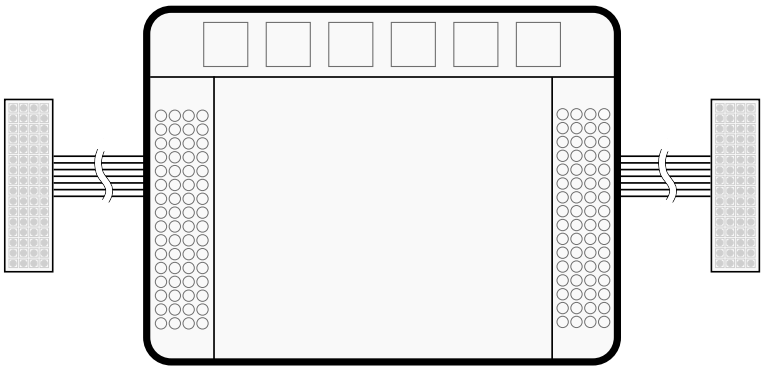


Figure 4.1: First sketch of UI elements

prototypes, in our case with young participants this would also help the students not to think too technically about the concept and lower the barrier to engage with it and be more open. This also meant for our design that it should be easy to use and at the same time changes should be easy to undo changes.

Based on related works we designed a first sketch of the basic structure

Based on works we found in our literature review, such as Arakliotis et al. [2016] and Kato [2010], and openly available software such as Scratch, we first designed a rough structure for the basic UI elements of the application. Figure 4.1 illustrates both the setup of an actual use case for the application and the rough separation of the different areas in the UI. First of all the tablet is positioned in between the two breadboards, which communicate with the tablet. However, the kind of communication - in matter of both hardware and software - is not important in our work.

The rough structure consisted of three main elements, two breadboards and the function block panel

Regarding the UI itself, there are two virtual breadboard areas on the left and right side of the screen, and the top area, which acts as a panel, is a separated area populated with the function blocks. As explained before, the right virtual breadboard (RVB) depicts the breadboard which is on the right side of the tablet and serves as input, analogous for the left virtual breadboard (LVB) serving as output. These areas surround the space in the middle, which is the workspace. This is the main area in which the user creates the flow of the program.

This division of areas was also the static part of our low fidelity prototype, the participants should not change anything in this structure. The dynamic part which we wanted to evaluate was the workspace. We wanted to see what students thought of the concept of visual programming by using function blocks and lines, which would mainly be performed in this area.

The setup was split into a static part and a dynamic part for the participants to play around with

The panel at the top illustrates a row of function blocks (FBs) for the user to use in their program. Those FBs would be generic and could stand for any kind of function. These can be a logic gate such as AND, OR, or XOR, etc. or conditional constructs such as an IF statement. Functions such as these are basic concepts of programming and can be used to describe logical conditions and manipulate input and achieve a certain output. In order to use them for their program, users could use gestures such as touching them to spawn an instance of this FB or drag them from the top panel into the workspace.

The panel at the top would depict function blocks which represent different functionalities in programming

4.2 Apparatus

For the mock-up of the breadboards on the sides, we used two blocks of polystyrene and positioned them to the left and right of the tablet (see Figure 4.2). Colored pins would be used to depict input pins. As for the right breadboard, we used the same but used output terms for the labels such as "LED". The current states of inputs and outputs would be communicated verbally during the session.

Our apparatus consisted of basic material to allow adjustments

As discussed before, we wanted to achieve an experience as close to actual touch interaction on a touchscreen as possible while illustrating our concept. For screen interaction itself, we decided to use a transparency sheet together with overhead markers. Since we had a static area and a separate dynamic area, we used a sheet of paper beneath the transparency sheet, on which we drew the UI, which then could not be edited by the user. In the application, the FBs would be used by using gestures. To achieve this in our simulation, we put different stacks of small post-it notes in the top area and labeled them with function names such as

People could use pins, pens, and post-it notes to draw flows and connect pins

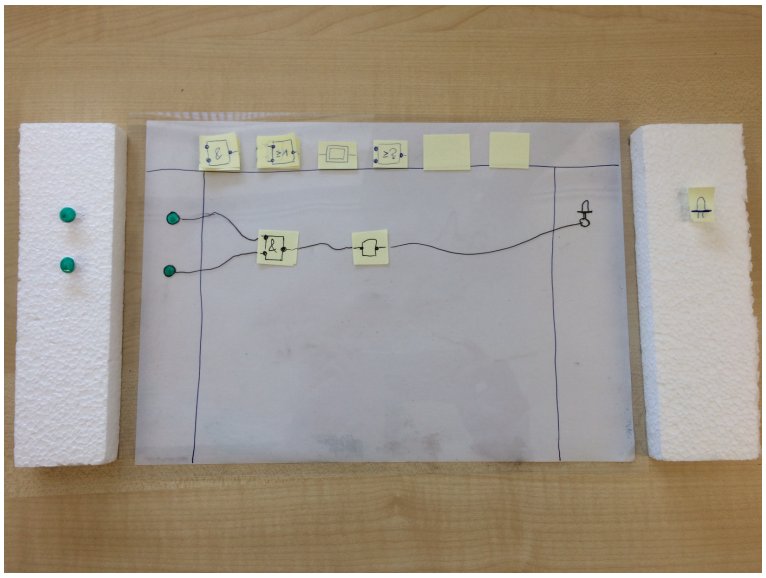


Figure 4.2: Picture of the low fidelity prototype

AND, OR, etc. With this, the user could take a post-it note from the stack and place it somewhere in the workspace to imitate a FB. This is not the same as click and drag a FB into the workspace but in this matter the important part was to provide the ability to create FBs. The user could then connect those by using the marker and drawing lines between the different blocks. Analogous to this, the user could also connect pins on the LVB with FBs and similarly connect FBs with pins on the RVB, resulting in an actual data flow in which input and output would be connected.

With the provided tools it was also easy to edit and adjust the flow

Due to the use of transparency sheets and markers, the drawn lines could easily be edited or removed and placed FBs could be removed by just putting them away. This guaranteed our requirement of keeping it simple and easy to use. Of course, this method of removing objects is not a mechanism which works the same way on touch surfaces, but as we explained, this was not the focus of this study.

It is important to point out that as described in the beginning of this section, the focus of this very first study was to see if the basic principle worked well. Hence, although individual mechanisms such as the removal and editing of

elements would not necessarily be implemented the same way in the application, this was not the goal of this prototype.

4.3 Study design

The goal of this study was to see what people thought of the very basic concept and design sketch

As this was the very first iteration and draft of the concept, there was no clear design and structure for the study. We designed it specifically to demonstrate the concept and let participants try it out and tell us what they think. The participants for the study were two students who did an internship at the chair with the school laboratory, which we mentioned before. Since they were technically versed and showed remarkable results in the programming exercises (compared to the other students at the lab), they could provide valuable feedback.

In the beginning the study conductor explained the concept to the participants

During the study our setup was as depicted in Figure 4.2. Additionally a study conductor was present to demonstrate the prototype and concept to the participants. In the beginning the motivation and idea behind the concept was explained to the participants. Since they were more experienced than our primary target group and could potentially provide more feedback from an experienced perspective, this explanation was more detailed. For the same reason, due to their experience with programming, they were also able to put themselves into the position of our target group and give advise on our UI design and functionalities.

The study conductor then explained the functionality of putting FBs into the workspace and drawing lines in order to connect them and create a flow. After demonstrating a simple use case scenario, the participants were asked to try it out themselves and communicate their ideas during the process and afterwards.

4.4 Evaluation

They liked the visualization and stated they could imagine that it helps lowering the barrier for beginners

In general, both participants liked the concept of visual programming and how it was designed in our application. They liked the mechanism of drawing lines to connect objects and use functions in order to create a certain logic. Furthermore they mentioned that the idea of programming visually might indeed lower the barrier for people and help engage instead while also making it more enjoyable for them. This would also especially break the gap for young students who feel reluctant regarding coding and programming, or tech in general. Unfortunately there was not a lot of feedback regarding how our UI concept and how it could be improved, except for a few remarks for adjustments we would not consider in our work.

Chapter 5

High Fidelity Prototype

With the approval of our low fidelity prototype, we set our goals and guidelines for the more advanced level of the high fidelity prototype. Unlike the previous one, this prototype was meant to be a technical solution running on a tablet. Since at first there was no rationale for a specific platform, we decided to develop this application in Unity, a multi platform solution, which builds for both Android and iOS tablets. As the device for testing we used an iPad Air 2 with a 9.7" display and an item size of 0.2 x 6.6 x 9.4 inches (L x W x H).

The high fidelity prototype was aimed to run on multiple platforms

The primary goal of this prototype was to be able to demonstrate visual programming as a concept to students. With this still being a prototype, it would serve the purpose of a proof of concept. For this reason we wanted to implement different mechanisms to allow different kinds of interaction, and put the focus on these mechanisms instead of covering all the functionality such application would offer in a real world scenario. In the following sections we describe this in more detail.

The main goal was to demonstrate the concept of visual programming and not cover all possible functionalities

5.1 Design

The first sketch was directly based on the low fidelity prototype

Due to the mostly positive feedback regarding our basic concept, this meant that we could fully adopt the basic UI design elements into our high fidelity prototype with the fundamental elements. With the results of our study and how the participants created flows, we also aimed at a similar design for the workspace.

As described in Chapter 4 "Low Fidelity Prototype", the virtual breadboards are positioned on the sides to signify a relation to the real physical breadboards which would be placed on the sides of the tablet. The part at the top serves as a function block panel containing all the FBs the user can use for their work flow.

The high fidelity prototype needed to emulate the behavior of the real pins on the virtual breadboards

However, the virtual breadboards needed to be designed differently for the following reason. Our work focuses on the software part, more specifically the UI. This means that we do not consider connected physical breadboards which transmit signals and therefore need to emulate this part. Considering the emulation, this further meant for the left breadboard, that we needed to emulate the behavior of virtual pins. The right breadboard would normally represent the pins of a connected physical breadboard with output elements. Hence we needed to emulate the behavior of those elements in our virtual concept.

Regarding the workspace and the visualization of the data flow, we also wanted to adapt the results of our low fidelity study, which is depicted in Figure 5.1. Based on the drawn flow in Figure 4.2 it illustrates a possible flow drawn with touch gestures and the use of function blocks and lines.

We will explain the interface and interaction design in detail in the following sections. Furthermore we will describe a practical example to demonstrate how a user could interact with our application.

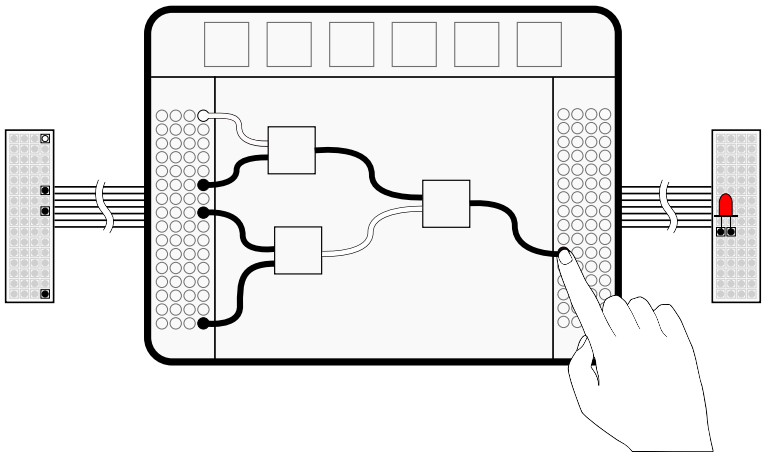


Figure 5.1: More detailed sketch of the UI and the interaction, showing a possible flow with enabled pins and a LED as an output

5.1.1 Interface

The interface builds the core element of our work since this is the component the students would see and use. It contains the UI elements themselves but it also provides the interaction methods and mechanisms allowing the users to create flows and connect elements with each other.

Breadboards

Two breadboards for input and output

As shown in Figure 5.2, we placed the two breadboards on the far left and far right side of the screen (a). We also explained that the left breadboard acts as a collection of input pins and the right breadboard contains pins which are connected to output devices.

Input pin states and values are indicated by different colors

The left breadboard contains a couple of input pins the user can interact with in different ways. These input pins have different states indicated by different colors, which represent the current value and signal of the pin. The default color is gray which indicates the neutral state, in which there is no signal and no value. This is the state when the

Figure 5.2: Screenshot of the app when it is started the first time

app is started and no interaction has been done yet, other states depend on the user interaction. After enabling them, other states are indicated by either a white or green color, representing the lowest or maximum value, or an interpolated colors between these two to represent values in between. In general, the user can tap, click and drag, and long click it.

UI Popup when tapping neutral input pin

If the user taps an input pin, a little UI pops up to the right side of the clicked pin (see Figure on the left). This UI allows the user to choose the input type for this pin, analog or digital and confirm it. An analog pin starts a simulation of a sine wave signal, which computes a value between 0.0 and 5.0, which is then visualized by a gradually changing color from white to green and vice versa. If the input type is set to digital, the value is first set to 0, resulting in a white color.

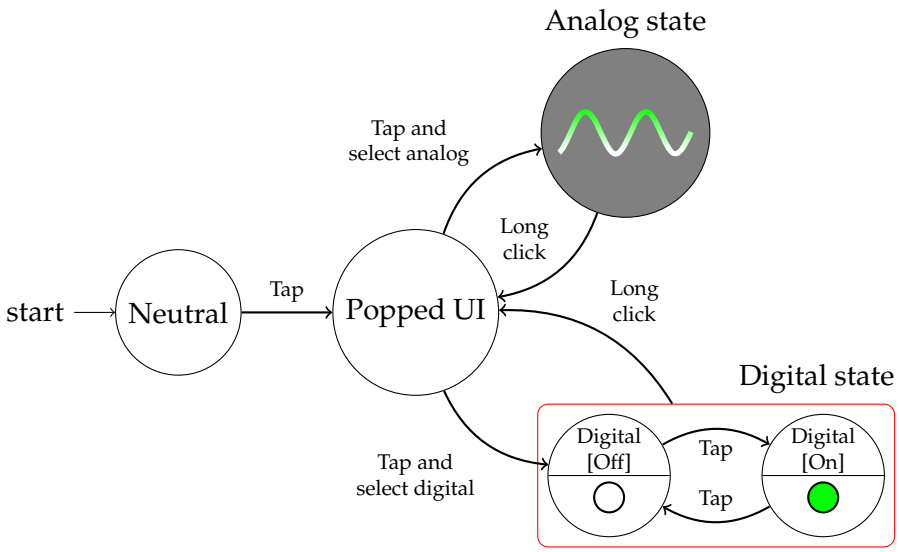


Figure 5.3: State model of the input pin behavior

This also results in a change of the tap behavior, once set to digital. While the analog simulation runs autonomously, the digital state can be toggled between `off` and `on` (or respectively between 0 and 1) by tapping the pin again afterwards.

When the input pin is set to a non neutral state, either digital or analog, the user can then long click it in order to change this setting. Long clicking will open up the UI popup again so the user can toggle to a different state. All the described interactions with the input pin are depicted in Figure 5.3. The UI popup also appears when drawing a line from an input pin to the input of a function block, this will be explained in Section 5.1.2.

States can also be switched by long clicking

Function Blocks Panel

At the top of the screen we placed a panel which contains depictions of the available function blocks. More precisely, they serve as templates, from which actual function blocks can be created in order to be used. Originally, the idea was to visually separate the FBs with respect to their functionality (e.g. those which work with digital signals and those

Due to the size of the iPad and the screen size, function block templates at the top panel could only be slightly separated

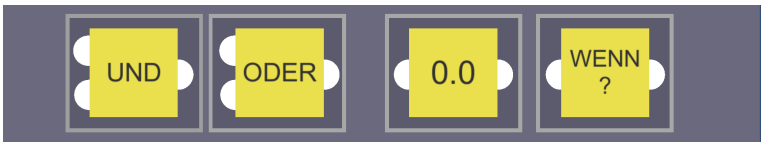
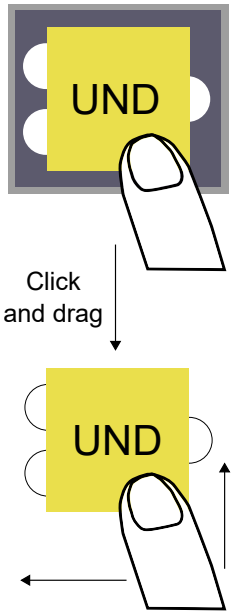


Figure 5.4: Function Blocks Panel at the top with German words "UND" for "AND", "ODER" for "OR", and "WENN" for "IF"

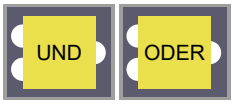
with analog signals). Unfortunately due to the screen size we were constrained in this regard and were only able to group them slightly together, while also considering their size to guarantee full touch interaction. The function block panel therefore functions as a metaphor of a conveyor belt or an inventory, showing and providing items for to use.



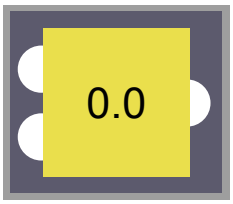
In order to create function blocks, the user can touch the template of a function block, which pops an actual function block, and then drag it to the working space and place it there by letting go of the touch. We restricted the movement of the dragged function block on the horizontal axis, so that the user can not drag and place it on the breadboards on the left and right side. This guides the user and prevents them from creating a semantic constraint.

Compared to the low fidelity prototype in Chapter 4, we wanted to implement real and actual functions. Our focus for this panel further lied in providing a minimal set of different functionalities. In addition to this we were restricted in terms of screen space causing a limitation in the amount of function blocks. This is why we used a small and minimal set of them instead of covering a wide range of functionalities for different purposes and use cases. Hence our idea was to create some function blocks which serve as representatives of a set of functions. We mainly separated the function blocks into digital and analog and split the latter into two separate functions. In the following we will explain these function blocks and their functionality.

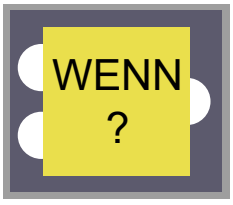
- Logical Function Blocks: When it comes to very fundamental and simple conjunctions between signals, there are logical gates such as AND, OR, NAND, etc. as we have also briefly mentioned in Section 4.1. However, what these have in common is that the function takes in two values (except functions such as NOT) and outputs one value. The only difference is the way the output is computed by the function. For this reason we chose both AND and OR as two representatives for the set of logical functions which work as their names imply. In addition to this we chose two representatives because for our study we wanted to provide the opportunity of a selection for the user so that they had to think about which one to use, depending on the given task (see Chapter 6 "Study").



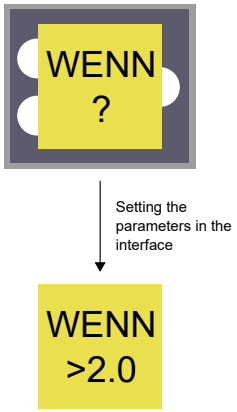
- Function block with dynamic label: While the first two function blocks have a fixed label which does not change while in use, it can be useful to have certain information dynamically displayed on the function block. These can be information about the current input, certain signals or other properties, or even debugging information. To keep it simple and easier to understand for beginners, we chose to display the input value in this function block. If nothing is connected it shows a default value of 0.0 and otherwise the input value is shown. For instance, in our app we provide a so called VALUE-FB which can be used to visualize the analog input value in addition to the changing color of the input pin.



- Function block with parameters: The functions for logical function blocks require two input parameters, but those are passed to the function with the connected lines. Some functions, however, might need arguments or parameters in addition to the inputs and outputs. For example, in programming, a loop construct such as a FOR-loop can iterate through different objects which are passed as input and create a certain output, but in addition it might use other parameters in use with these objects. For our app we decided to use a simple IF-construct with only one case. However, in order to pass the additional arguments, there is also an interface as a part of this func-



tion block. In order to maintain the idea of a simple and minimal interface, it only contains the bare minimum of information and input interfaces, as shown in Figure 5.5. The user has to input two types of information, the type of comparison (a), and the threshold (b) and then confirm it via the "OK" button (c). For the former the user can choose between *equal*, *bigger*, *lesser*, *bigger equal*, and *lesser equal* in a dropdown menu. The default value is equal and since our task (see Chapter 6 "Study") requires a different choice, the user needs to use this dropdown menu in order to solve the task correctly. The threshold is selected by using a slider widget with the selected value depicted on the right side of it. This interface pops up when the user positions a newly generated IF-FB for the first time on the workspace. It also appears when tapping an IF-FB again afterwards in order to change the parameters for that block.



Confirming it closes the interface and accordingly labels the function block. For example, a neutral IF-FBs' label says "WENN ?". In case the user selects "bigger" as the comparison type and 2.0 as the threshold, the label will change to "WENN $<$ 2.0".

In the next subsection we discuss how the user can create and edit a flow with these function blocks.

5.1.2 Practical scenario

The main activity in our application is drawing lines between pins in order to connect inputs and outputs with each other. This yields different mechanisms such as creating and editing lines and also working with function blocks and removing them if necessary. In the following we will describe a cognitive walkthrough of a user generating a flow in order to demonstrate the different mechanisms in our application.

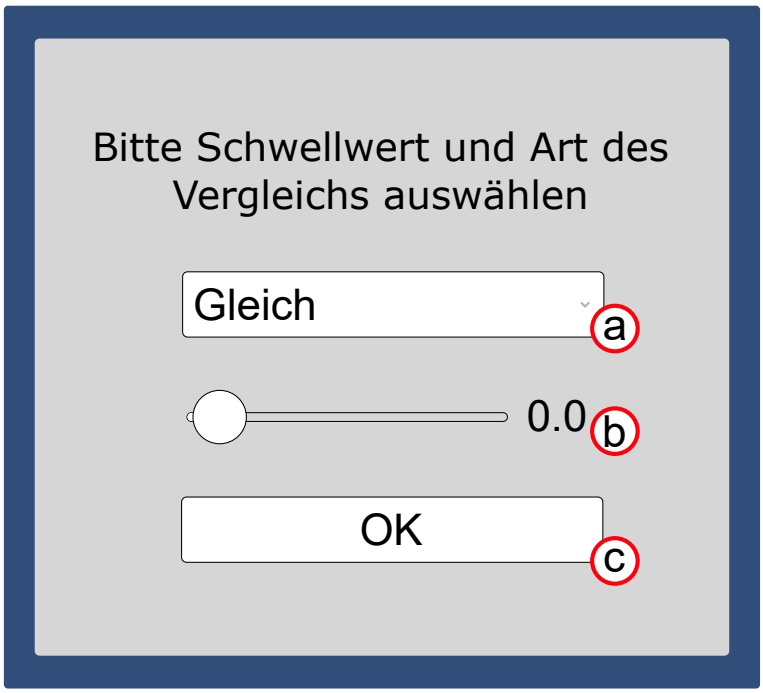


Figure 5.5: Interface appearing when the user interacts with the IF-Function Block, consisting of a dropdown menu to select the comparison type (a), a slider to select the threshold (b) and a button to confirm and close the interface

The user wants to turn on the LED with an AND logic

Let us assume that the user wants to turn on the LED if two signals are "switched on". This requires using two input pins which are connected via an AND-logic. First, the user could click the AND-FB in the function block panel at the top and drag it down to the workspace (Figure 5.6.a). In order to connect the components together, the user would now need to draw lines from the input pins to the function block and then one to the the output pin of the LED.

To draw a line, the interaction is similar to creating a function block. The user touches one of the input pins on the left virtual breadboard and drags - and thus draws - a line with the finger (Figure 5.6.b). The drawn line starts from the input pin and follows the finger. When the finger comes close to the input pins of the function block, it snaps to it, indicating that connecting the line to this pin is possible. If the

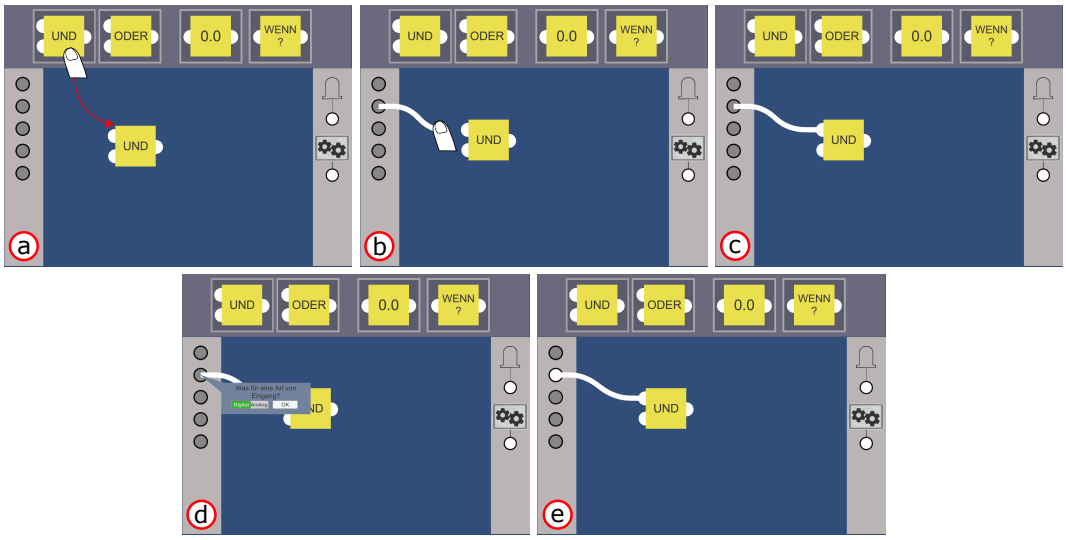


Figure 5.6: The necessary steps when creating an AND-FB and connecting it to an input pin: creating a new AND-FB from the function block panel (a), drawing a connection from an input pin to this function block (b-c), and setting this input pin type to digital (d-e)

A line is drawn by click and dragging from an input pin to the input pin of a function block

line is snapping to the pin and and the user stops the touch, the line is drawn and a connection is established between the input pin of the virtual breadboard and an input pin of the function block (Figure 5.6.c). This also pops the analog-digital interface next to the used input pin. Since the user wanted to have two switched on signals, they choose digital by clicking the switch and confirm it (Figure 5.6.d). This sets the input pin to a digital type and sets the value to off or 0 respectively (Figure 5.6.e).

After another line, the function block is then connected to the LED pin in a similar way

The same procedure then has to be done again with another input pin. Therefore the user repeats the process with different pin and the second input pin of the AND-FB. With this, the connection from the left side is finished and what remains is connecting the function block with the right side, more precisely with an output pin (Figure 5.7.f). Technically, this means that the user would now need to draw a third line from the output pin of the function block to the LED pin on the right virtual breadboard. This is done in the same way as before, but here the user touches the output pin of the function block first and then drags and draws a line from there. Similarly, the end point of the line also

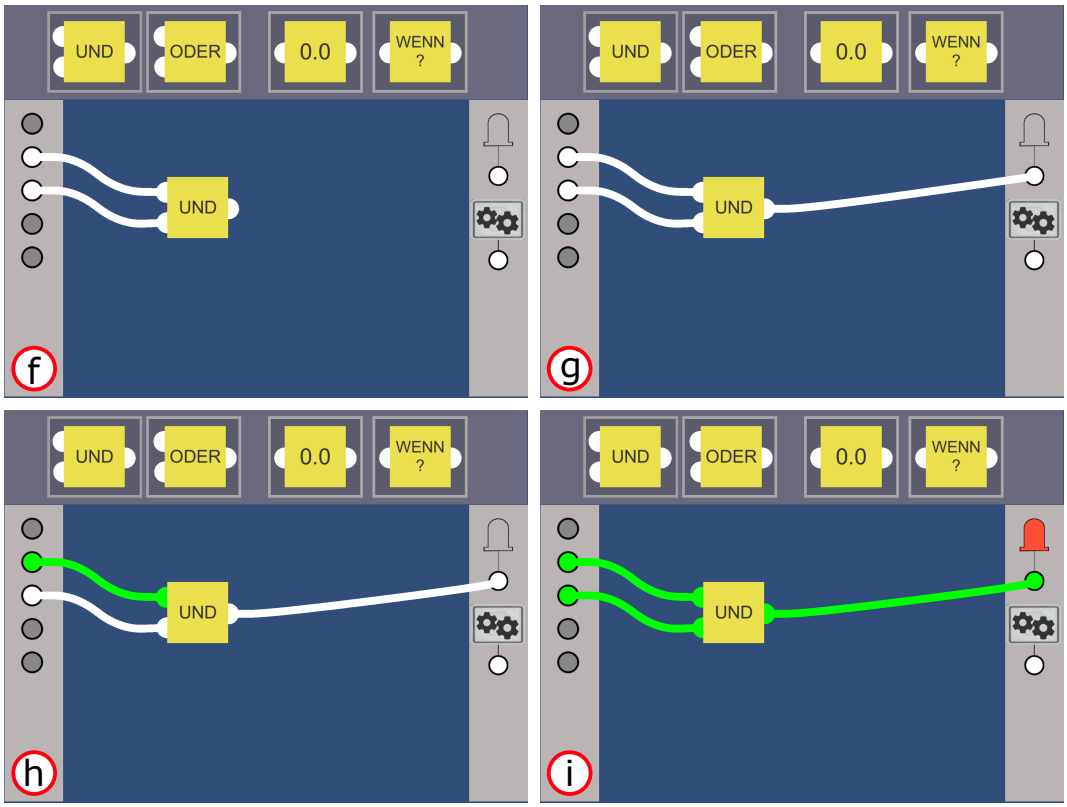


Figure 5.7: The subsequent steps to establish all necessary connections and turn on the LED: adding a second input connection (f) and an output connection (g) and setting the input pins to different states to test the logic (h-i)

snaps when the finger gets close to the LED pin to indicate that it can be connected (Figure 5.7.g) .

The user can test the logic by tapping the input pins and see if the LED turns on or not

After this, the user established a flow by creating the necessary lines and can now test if the logic is working as intended. The user could tap one of the used input pins and switch it to an "on" state (Figure 5.7.h). This should turn the input pin and also the connected line green, visualizing the "on" state. However, as the function block logic is an AND logic, nothing is happening yet and the LED stays off. In order to function as intended, the LED should turn on if both input signals are switched on. To achieve this, the user could then tap the other input pin and turn it on and this should result in the LED changing to red (Figure 5.7.i). To further test this, the user could turn off one the input pins and the LED would turn off as well.

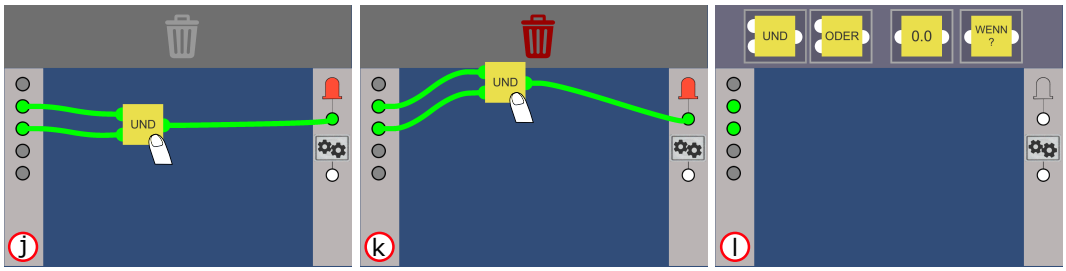


Figure 5.8: Removing the previous AND-FB using the removal overlay by clicking (j) the function block to enable the removal overlay and then dragging it to this area (k) and then letting go (l)

Before connecting the gears, a function block together with its connections can be removed by dragging it to the top panel when it changes to the removal panel

Without further testing we assume that this task is solved and now the user wants to try something else, e.g. to use the gears instead of the LED and turn them on if an input signal is bigger than 3. Since the old solution is still there, the user needs to either remove everything, or adjust and edit the flow. First of all, the AND-FB is not needed anymore and therefore needs to be removed. The user does this by clicking the AND-FB and dragging it around. This interaction is primarily for moving the function block around but it also changes the function block panel to an area where function blocks can be removed. While clicking and holding a function block, the panel changes and shows a trash bin suggesting removal of objects (Figure 5.8.j), a similar mechanism as in Danado and Paternò [2014]. When dragging a function block to this area, the recycle bin changes its color to red which indicates that the object is removed when letting go of the touch (Figure 5.8.k). This does not only remove the FB but it also removes the lines connected to this particular FB. In our case this is the only FB and therefore this results in a clear workspace (Figure 5.8.l).

An IF construct can be created with the IF-FB together with its interface for the parameters

For this new use case the user wants to use a conditioned logic. Therefore they create a new IF-FB using the same mechanism as in the previous case. After positioning the function block, the corresponding interface appears and the required parameters need to be set (Figure 5.9.m). Since the user wanted to turn on the gears if the input signal is bigger than 3, the user selects *bigger* as the comparison type and 3 as the threshold, and confirms it by clicking the "OK" button. The function block is now set, indicated by the change of the label, and can be used to create a new flow

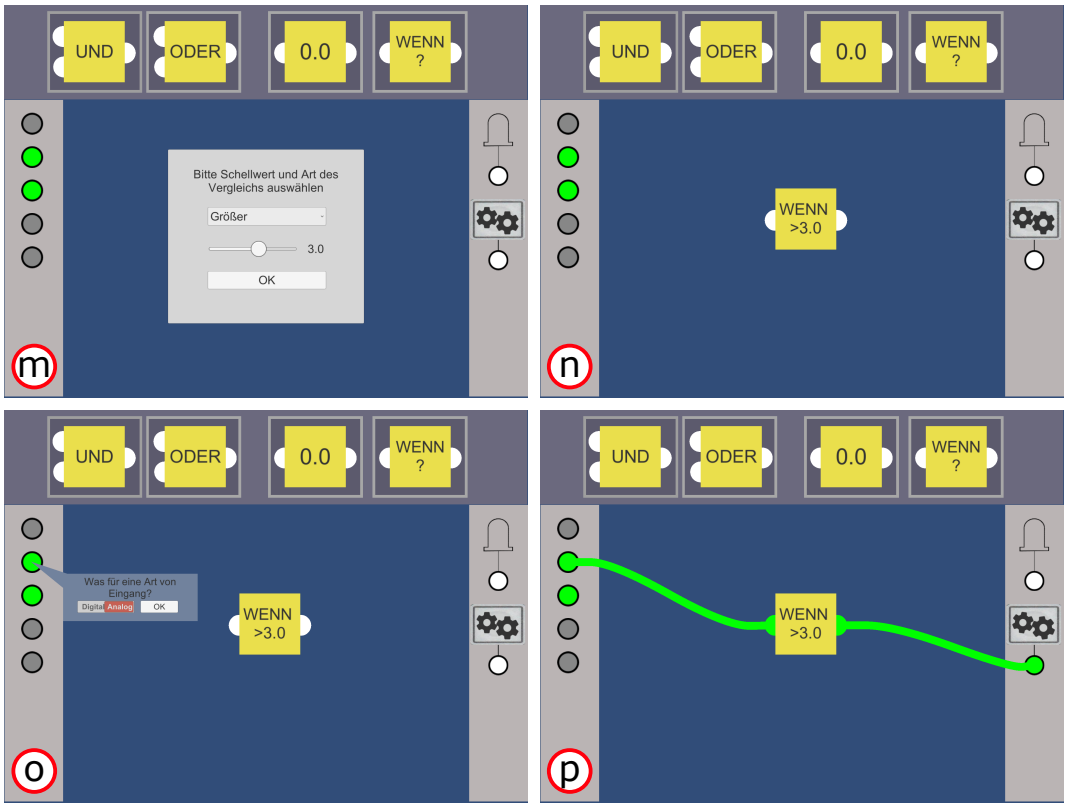


Figure 5.9: Adding an IF-FB to the workspace and setting the parameters (m-n) as well as changing the input type to analog (o) and connecting the function block with the gears (p)

(Figure 5.9.n). As before, the user could now connect an input pin with this FB. Two input signals are already active and could be used, but they are digital input pins and an analog type is needed. A new input pin could be used but the user decides to change one of the already active pins and change its type to digital. They long click one of them to open up the analog-digital interface again and select the analog type instead of the digital type (Figure 5.9.o). This results in an automatically changing input pin signal between the values of 0.0 and 5.0 indicated by the gradually changing color (see Figure 5.3).

They also change the input pins from digital to analog by long clicking the already enabled input pins

Now that the input is set, the user needs to connect the output of the IF-FB with the gears. Again, this is done the same way as before. With the logic of the function block set and the input and output connected, the gears

If done correctly, the gears should start rotating

should start rotating the moment the connection is established (Figure 5.9.p). More precisely, the speed of the rotation depends on the input value and therefore the higher the input value the faster the gears rotate.

The user wants to test this setup and use an additional VALUE-FB to display the current input value

Let us imagine that the user now wants to test this setup further. The visualization of the input value provides some feedback but the user wants to see it more precisely. This can be done with the VALUE-FB, which is labeled with 0.0 per default in the function block panel. Hence the user could drag this function block to an empty spot in the workspace, leaving the other flow as is (Figure 5.10.q). As the VALUE-FB takes in a value and outputs the same value, indicated by one input and one output, the user needs to add this function block into the flow from before. The user chooses to put this before the IF-FB in the flow. To integrate it, the connection between the breadboard input pin and the IF-FB needs to be split up. This can be done by taking this connection and changing the end point from the input pin of the VALUE-FB to the input pin of the IF-FB. To do this, the user can click the end point of the current connection, which results in the line "unsnapping" and the end point following the users finger, then drag it to the other input pin and let go (Figure 5.10.r-s). If done correctly, the connection should be changed now.

The end point of the first connection is changed by clicking and dragging it to the new VALUE-FB's input pin

The user can then connect everything again and has a new component in the flow

The remaining step would now be to connect the VALUE-FB with the IF-FB. Since this is creating a new line and not changing an already existent line, the procedure is the same. This should now result in a flow which is similar to before but now with an additional component which also shows the current analog value so the user can check it more transparently (Figure 5.10.t).

This cognitive walkthrough demonstrated most of the mechanisms in our application which can be used for different purposes. The tasks also built the basis for the study which involved the high fidelity prototype. In the next chapter we will describe our study regarding its design, the methodology, and the evaluation of the results.

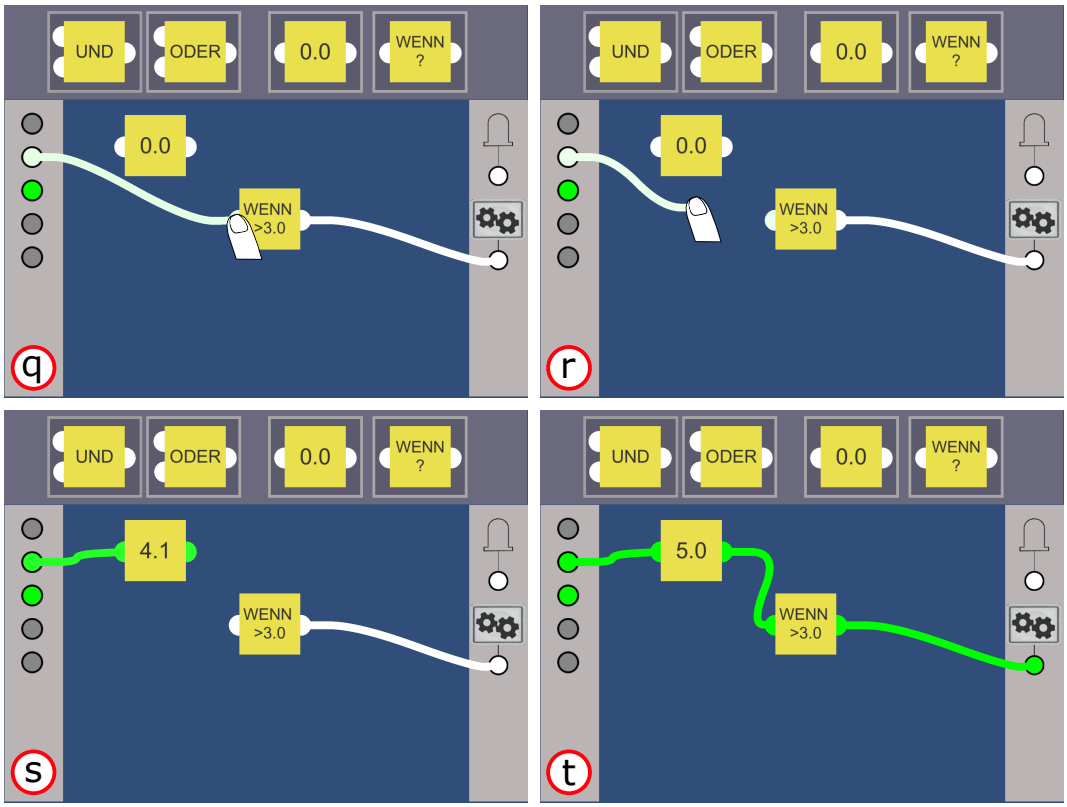


Figure 5.10: Adding a new VALUE-FB into the flow by adding it first (q), adjusting a previous connection (r-s) and then connecting the new function block with the old function block (t)

Chapter 6

Study

In the previous chapters we described our prototype going through the low fidelity phase and how we used this phase to design a high fidelity prototype in the form of a tablet app. This chapter will discuss our study in which students tried our app in order to evaluate how easy it is to use and what they thought of it.

6.1 Design and Methodology

To determine these factors as results, we needed to set some guidelines for our study. First of all, the target group were young students aged from 14-16 years, all with different knowledge of technology and technical skills. In order to keep their thoughts as open as possible, we chose to conduct the study mainly verbally and not use any written forms with questions. For this reason and because we were told that usually the students we would ask tend to have lots of thoughts and can therefore tell a lot, we decided to record the sessions on video. We also designed the study as open as possible and mainly qualitatively, also because it is harder for kids at such age to rate and rank things (e.g. with Likert scales) as they can lack reference values. We will discuss the concrete realization of these guidelines in the following subsections.

Sessions were recorded and kept informal and verbal to allow open answers

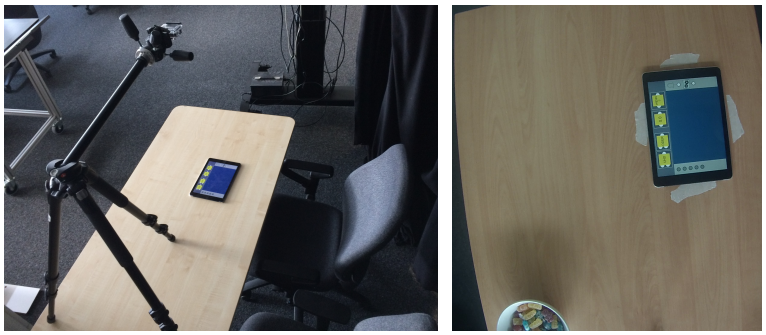


Figure 6.1: Setup for the study with a GoPro recording audio and video from above, capturing how the participants use the app (Left picture shows the setup and the right picture shows the camera perspective

6.1.1 Apparatus

As noted before, the sessions were recorded, which required a setup with the camera and tablet running the app. This setup can be seen in Figure 6.1 and it was used for the whole session.

Setup used a top down view with a GoPro camera to capture how the app was used

As depicted in the figure, we used a tripod with a GoPro camera which would record the student using the app from above. Since we were mainly interested in hearing the participants' answers and seeing how they used the app, this angle allowed to record them while they use it but hide their faces to preserve anonymity. However, the recordings were only used for the discussed data and were deleted after the evaluation. At the beginning of each session the participants were informed and asked about this.

6.1.2 Procedure

We split the study into three phases, similar to Danado and Paternò [2014], starting with a few questions about demographic data and their technical knowledge, practical tasks to test the app, and finally some questions about what they think of the app. The demographic data concerns their age and if they have any experience so far with microcontroller

and programming of such. If they had any experience we asked them which software they used. This would show the diversity of different knowledge levels and it might give insights between previous knowledge and how they used our app and what their opinion would be. The video recording would also start at the beginning of this phase and therefore also record the respective questions and answers. Giving the students a particular task serves the purpose of giving the participants a both narrow but still open specific scenario. It is narrow as there is a certain goal we want them to achieve but it is also open because the way to achieve this goal does not follow a straight set of steps as we will see in the following. Solving such tasks in particular is also useful to us as we want to see how students perform when they need to solve certain problems using our app. Furthermore it provides insight into issues and what might confuse the participants.

Study was split into three phases with the first asking about age, grade, and about their technical knowledge

The practical phase contained tasks for the participants to solve on their own

We designed three tasks in total for the participants and they would solve them consecutively without resetting. The tasks were designed to be as simple and as realistic as possible, because we wanted to leave enough room for many possible solutions. This way the participants can explore the application when thinking and trying out the possibilities and different mechanisms the app provides. Furthermore a simple task design can also help reduce the cognitive workload. This is also a reason we kept the task design implicit. Explicit task formulations as those who may appear similar to exams or school assignments could cause additional stress for students (similar to the Hawthorne effect in studies), compared to task design which is more open and also realistic, in the sense of being as close as possible to an actual task or cognitive goal in the student's mindset when working with microcontrollers and processing of signals.

Task design was kept simple and realistic to allow broad range of solutions and alleviate the Hawthorne effect

After finishing the questions from the previous phase, the study conductor turned on the tablet and started the app. They would then explain that there are three tasks in total and that they are free to try out the app and solve the tasks in any way they want to and can. Also they would be able to see if they achieved the goal and did not need to ask for that. In addition they were told that they should try

Students were told to try and solve the tasks on their own and if stuck they would get hints

to solve the tasks completely on their own and only ask if they are stuck and need help. This would serve as our metric to measure how easy it was for the participants to use our app. We were interested in how many times they got stuck and what caused their confusion. In the following we will give an overview of the three tasks. As described in Section 5.1.2, the scenario provided a basis for the tasks we would assign to the participants.

Task 1: Turn on the LED
A simple task in which input and output is connected. Because of that it also allows many ways of solving the task. It also ensures the basic understanding of inputs and outputs and the functionality of the application, which is necessary for the following tasks.

Task 2: Use two input signals to use the gears and turn them on if one input signal is set to on
The first task which introduces a condition. In this case the student shall again connect input and output but this time with another output and with a condition under which the output is enabled. This forces the student to explore and use other mechanisms aside from connecting ports with each other directly. More precisely, they must use function blocks to create a logic for this task.

Task 3: Turn on the LED if the input value is bigger than 2
Similar to Task 2, the student is again forced to use new mechanism. In this case, the "IF" function block must be used, which then introduces the concept of the UI to the student, which they need to interact with.

As described before, all these tasks were given to the participants verbally by the study conductor and if they got stuck, they were given advice. During a task the study conductor observed them and told them when they achieved the goal and explained the next task.

After finishing this phase, the final phase was conducted in form of a verbal questionnaire. This yielded general insight into the understanding of the participant regarding the app. After the participants performed the tasks from before, they would have gotten a certain understanding of the app and how it can be used. This serves as a base for us, from which we can draw conclusions about the usability and functionality.

In the final phase the participants were asked some questions about the app

In correspondence to our set guidelines, the questionnaire was designed open, qualitative, and general. We put focus on the open aspect because we did not want to restrict the answers and allow a broad range of those For the same reason we designed it to be general and qualitative. Since we were recording the session, the study conductor asked the questions similar to an interview and the answers were recorded instead of being written down. This also helped to capture their thoughts and ideas more precisely. The questions were as follows:

These questions were also designed open and qualitative to not constrain the thoughts of the participants

1. What are your thoughts on this [concept]?
2. What did you like about this [concept]?
3. What did you not like about this [concept]?
4. Can you imagine using this again [to experiment and play around]?

While the first question was meant to gain a general idea of what the participants though of our app, question 2 and 3 were meant to narrow it down to particular good and respectively bad aspects. Question 4 was intended to hear how balanced the answers to the previous questions were and if it was good enough that the students felt they could imagine using it again.

The questions aim to give insight into what the students thought about using the app and their critique

It is important to note that we tried to reduce bias in the phrasing of the questions as much as possible. For this reason, and similar to the technical question in the first phase, the questions were formulated as simple as possible without any technical terms. Since our focus with this work also lies in breaking the introductory barrier for young students

Phrasing and wording was chosen specifically to be as simple and as least irritating as possible for the students

to get them interested in working with microcontrollers and programming, we did not want them to think too technical and therefore risk negative associations due to prejudices. For example the wording "What are your thoughts on this app?" or "What are your thoughts on this concept?" could direct the thoughts of the participants to more technical thinking than needed. This holds true for the term "app" but is even more important with abstract and technical terms such as "concept". Hence the study conductor did not use these words and if it was not clear, referred to the app non verbally or using other words.

6.1.3 Participants

Seven participants with similar ages and one half having diverse technical experience and the other half having none

We conducted the study with seven participants aged from 12 to 15 ($SD = 1.069$, $M = 13.8$) with two girls and five boys. All but one participant were between the 8th and 9th grade (the study was conducted during the summer break). Regarding the technical knowledge we determined that there was a broad range with four participants having no knowledge at all to 3 participants who told us that they already worked with some programming languages and some of them also used IDEs such as Lego Mindstorms, Scratch, or mBlock[1]. One also mentioned that she had experience with Arduino and microcontrollers.

All the participants of our study were originally participating at a student lab at our university[2] in which they use App Inventor, a software similar to Scratch, to build apps. We conducted our study at the same time and asked if some of them wanted to participate in our study as well.

6.1.4 Results

In this subsection we will briefly show the results of the study consisting of the answers to our questions and how they performed the given tasks. In advance, all participants

[1] http://www.mblock.cc
[2] http://schuelerlabor.informatik.rwth-aachen.de/

were able to solve the tasks. Differences could be found in their approach and some of them were given hints.

Task 1

Three participants were able to solve the first task completely on their own without asking for advice. Those were also the participants who had previous knowledge. Other participants needed at least one hint, one needed three of them. The latter seemed to have problems identifying all the possible touch gestures with the elements, which they could make use of. For example they tried tapping on the input pins but used the dragging gesture only on the function blocks. Hence they were not able to understand the concept of connecting elements. However, when the study conductor told them to try and do more with the input pins aside from activating them, and that they could use the dragging gesture on those as well, they had more success in the task. Finally they were given a hint to tap the input pins again to turn them on.

The first task was solved without issues by three participants, others needed at least one hint

One of the participants turned on the LED by using an IF-FB and connecting it with the LED, therefore she did not use any input pins. While this is also a correct result, she was also given the hint that she can also connect the input pins with the function blocks and not just function blocks with output pins, as she seemed to have problems with this concept.

One participant solved the task correctly without using input pins

The other participant who was given a hint got stuck because they only used the tapping gesture at first. When the study conductor told them that they could also try other gestures they were able to finish the rest on their own and solve the task correctly.

Another participant had problems identifying alternative gestures

The last participant, who had no previous knowledge, had technical issues as they tried to draw lines but the system did not recognize it correctly. However, also in this case the study conductor got involved, but instead of a hint as in the other cases, they just told the participant to try it again, which led to a correct solution.

Technical issues by the system hindered a participant from solving it directly

Task 2

Most of the participants solved the second task without complications

While two participants needed hints, five of the participants were able to solve the task without complications. However, one participant used the analog input type to connect to the OR-FB. Although in our vision, we aimed at digital input pins which could be turned on and off again to control the gears manually with these input pins, considering the formulation of the task ("if at least one input signal is on"), this is a correct solution because at a certain point one input signal would be 0.0 and the other input pin would be > 0.0 and the gears would be running, which satisfies the requirement.

Two participants required hints concerning logic and use of function blocks

One participant, who had no previous knowledge, almost solved the task correctly but used an AND-FB instead of an OR-FB. Another participant struggled more and needed a hint that they could use a function block and then another hint which one they needed to use.

Task 3

For the final task we saw that except for participant, there were no issues solving the task correctly. The participant who had problems, seemed to be confused by the VALUE-FB and IF-FB as they needed help with using the latter and not the former.

How did they like it

Most found it easy to play around with and some were confused in the beginning

All but one participant stated that they liked it because it was "easy to play around and experiment" with it. Two of the participants stated explicitly that they found it "a little bit confusing in the beginning" but after that phase it was easy to use and one, who had problems in the first task, said that "in the end it was all logical".

One of two participants, who stated that they were confused by the input pins and how to interact with them, also said that they would think that this interaction can be figured out by playing around because "with touch there are only so many gestures like tapping, double tapping, and long tap, so it needed to be something with dragging or swiping".

Although some had problems with the input pins, they said they would have found out eventually

What did they not like

Overall it seemed that no participant had major issues to criticize about the app. Three participants explicitly said they did not find anything bad at all about the app. One of these and another participant mentioned that the reason would be that it all makes sense and "people would surely be able to figure out everything on their own". The remaining criticism concerned design issues.

In total the participants did not mention any aspects of the app they did not like

For example the coloring could be adjusted as the participant suggested that the colors red and green are often associated with erroneous or correct and therefore the colors in the analog-digital interface could be irritating because they show analog with a red background and digital with a green background. At the same time digital and analog connections could be colored differently as well.

One mentioned that the coloring could be improved to make it more distinctive

Another participant suggested to use toggle widgets as they can be found in mobile applications or other types of switches for the input pins to toggle between off and on states.

What did they like

Most of the participants considered the visualization to be the best feature of the app. They said that it helped them "understand it better", one liked the visual feedback of the outputs and the gradual color change of the connections in particular, and another one stated that they liked that the "feedback of the results was visual and immediate".

Visualization was the most liked feature by most students who participated

Some liked the simpleness and others the experimental part

Two participants also said that they liked how simple it was and found it also interesting. At the same time four participants liked that they were able to find out things on their own while using this app. One mentioned that they found it easy to "orient themself" using the breadboards on the sides and the panel at the top.

Would they use it again

The majority would definitely use the app again for different reasons

Five out of six participants stated that they would use the app again. This includes those who said they could imagine using it to "try out [logical] circuits beforehand" and one would use it if it was more complex and provided more functionalities with the function blocks. Both these participants had previous experience with coding in Scratch and using Arduino.

Both experienced and non experienced participants seemed interested

One more experienced participant mentioned that they might use it on their own but they could imagine others would benefit from it if used for learning in classes or other scenarios. At the same time one participant with no previous experience seemed to enjoy it and played around during the last questions and asked questions about particular elements and some mechanisms.

Chapter 7

Evaluation

In our study seven participants with different technical knowledge levels tried to solve different tasks using our app and gave us insight into their thoughts on it. In this chapter we discuss the results of the previous chapter in general with respect to our motivation for this study and the relation between performance and the answers of the survey.

As stated before, four participants had no previous experience when it comes to programming and working with microcontrollers while the other three already programmed with Arduinos and Scratch. For the following this will also be our main dichotomization, those with at least some experience and those without any experience, for convenience we name the former group A and the latter group B. This was very important to us as we wanted to see if the approach and their thoughts would vary depending on this aspect. Hereby this holds true for both ways as we wanted to see if in some aspects those with experience judge differently than those without and if, for example, those without experience struggle a lot with the tasks compared to those with experience.

Participants are categorized into two groups A and B depending if they have prior experience or not

We saw that through all tasks group A had no issues solving them and therefore needed no help or hints. At the same time the other group needed at least some hints in each task except for the final task, in which all participants

Group A solved all tasks without complications while group B needed some help at first

had no problems except for one from group B, in which case a hint was needed. For this reason the first two tasks are more interesting and important to analyze as they caused the most confusion.

One participant who needed the most help in the first task, also stated in the survey that they were sure they could have done it without help after some time.

In the first task, two members of group B needed a hint regarding the drawing of lines. One participant did not know that they could connect input pins with function blocks and the other one struggled figuring out that more gestures than tapping can and need to be used. Aside from these issues, they were able to solve the tasks alone. One participant had bigger issues and required help with both interacting with input pins and also with gestures they did not use. However, that participant also told us in the survey afterwards, that although they were "confused by the input pins", they did not find anything bad about the app and were also sure that "if given some time, [they] would have figured it out on [their] own". Furthermore, it was interesting to see that the same participant was able to solve the other tasks without any further issues. They also enthusiastically told us that they liked the simple idea and that they felt quite happy to see the immediate positive results when done right.

The same participant solved the other tasks without needing any hints

Other participants also noted that after overcoming the first barrier it was easier

We observed this phenomenon also for the other members of group B. Although they needed some help, they all said that at first they found it complicated or irritating but after this first barrier it was good and "logical" and "made sense".

Only one participant used the wrong logical function block, but there is no evidence that the others fully understood logic

We can therefore note that the first task was the most complex when considering group B. Regarding the second task, again group A had no issues but also two participants of group B solved the task on their own. We were told beforehand that the students in this lab have most problems when it comes to understanding logic. One participant of the latter group used an AND-FB first and then needed the hint that this is not correct and that the task was aiming at a different logic, which immediately led them to use the correct OR-FB. While this seems to indicate that all participants did understand the logic of at least AND and OR, we do not know if this is the case. One also needs to consider that the phrasing of the task formulation contained the word *or*

which might have lead some of them to use the function block with this word on it, which does not necessarily implicate their understanding of the actual logic.

One participant who had general problems with function blocks was also the one most curious about the app

Furthermore one member of group B also got stuck and needed hints to think of using function blocks in the first place and then another hint to think of which one might be suited for this task. Again it was interesting to notice that this participant also stated to "like that you could figure things out on your own" and did not find it too difficult. The same participant also asked the most questions about the app and its features after the survey and kept playing around with it for some time.

In general, both groups did not have many differences regarding their thoughts on the app

Considering the answers of the survey afterwards, there were not many differences between both groups. In general, members of both groups seemed to have liked using it. This makes it interesting to see that even those with experience also thought well of it. One emphasized that they liked the illustration and noted that they thought it was "easy to understand" and they thought it would allow easy experimenting. The only difference we saw between the groups was that one member of group B based their opinion on imagining what it would be like for others as they themself did not see any use for their own, while the other members provided similar feedback as members of group A.

Members of group A criticized more about the app than group B

The question about what the participants found bad was particularly interesting since one could imagine that those with experience have more to criticize as the system could be too simple for them. Vice versa those with no experience could address many issues since they encounter more problems due to their lack of knowledge. As we saw it turned out to be the former as members of group A addressed some issues and members of group B except for one who mentioned that one might have "troubles in the beginning which is not so good but [...] it would still be easy to figure out on your own and those are not issues of the app". The same critique was also mentioned by a participant of group A. Aside from this the other members of group A said that the colors could be improved to make it more distinctive and design the IF-FB more helpful.

Only one member of group A mentioned the possibly irritating barrier in the beginning

When asked what they found bad, most positive feedback came from group B

While those were only some minor points, group A provided more critique than group B. The opposite, however, could be seen in the answers when asked what they thought was bad. Except one member of group A, who liked a broad range of features such as the coloring and the colored feedback of the input values as well as the flow based paradigm and who could also imagine that it can save work, the most positive comments came from group B, those who had no prior experience.

Their feedback mostly concerned the visual aspect of the app, although they could have been subject to the Hawthorne effect in this matter

As previously stated in Section 6.1.4, according to them the visualization and the simpleness of the app, together with the playful aspect to experiment, were the most liked factors. We can see from this that not only was group B very verbal about what they liked, but they also seemed to think about it in some detail and understood what made it good for them. However, it could be criticized that they mentioned some things only due to the fact that they were asked and they felt under pressure, therefore being subject to the Hawthorne effect. But as we tried to keep the interview as informal as possible and therefore reduce the pressure one could have in an interview situation, we presume that, based on how the participants answered and not just by the statements, the participants did mean it.

Independent of group, most participants stated they would use the app again

To some degree this is also backed by the consensus of the last question, in which five out of six participants, independent of the group, said that they would use it again for further experimenting or trying out circuits. The study conductor also noticed that especially the members of group B put emphasis on their positive answers.

Chapter 8

Summary and future work

In this chapter we initially summarize our own work and then provide perspectives for future work in these domains and research areas.

8.1 Summary and contributions

Our work attempted to provide a user interface for an app, which lowers the introductory barrier for young students in order to become acquainted with programming. Initially, our literature review and visiting a school laboratory suggested, that both mobile development for young students and physical computing using a visual programming approach are research areas which have not considered aspects which we try to address. The related research we presented regarding visual programming was mostly using the desktop metaphor and not direct touch. Works on physical computing aimed at different target groups, did not run the applications in real-time, or lacked empirical studies.

Our first low fidelity prototype provided elemental features of the concept and received positive feedback

Based on previous research we built a first low fidelity prototype and had two young students evaluate it. Due to the positive feedback we used the results as a basis for a high fidelity prototype in form of an app running on a tablet. This app featured basic elements regarding functionality in terms of programming. It provides logical functionalities for both digital and analog signals and allows connecting input and output signals in order to manipulate output devices such as an LED. Along with these elements, we also implemented interaction mechanisms to provide a simple workflow for users which would allow them to create flows without requiring any instructions or guides.

The high fidelity prototype featured a simple interface with engaging interaction mechanisms

Our aims were to keep the interface and the interaction simple, and also users to explore the app on their own and enjoy learning about the different possibilities of how to solve problems. Furthermore we wanted to motivate and engage more young people regarding programming and also making and designing applications [Fischer, 2002, Dougherty, 2012].

Except for some confusion for some participants in the beginning, most would use the app again and gave mostly positive feedback

We then conducted a study with pupils from a school laboratory using our high fidelity prototype. Seven participants initially told us about their diverse prior experience with programming and working with microcontrollers, in the end we had three participants who had some experience and four who were not technically versed in these domains. Our results indicated that except for some issues in the beginning some participants with no prior experience had, most participants had no major problems solving the tasks on their own. Furthermore, six participants stated they would use the app again and the overall feedback regarding the app itself was also mostly positive. Some pupils mentioned that parts could be designed differently regarding colors and the look of input pins as well as some function blocks.

Logical thinking seemed to be no problem but there is no clear evidence that participants fundamentally understood logic

In Chapter 3 we discussed how most pupils in the school lab had problems understanding logic and mathematical inequalities. While we did not explicitly ask or know for certain that all our participants understood the logical FBs they used, almost all of them were able to use the correct one after being given the task assignment. At the same time all of them used the correct inequality given in the task. We

assume this is mostly due to the formulation of the task which used the same wording as in the interface.

In summary, the positive results indicate that we were able to provide a simple and mostly intuitive user interface for development of physical computing applications on a tablet device. This approach can be seen as a first step towards a complete system which can teach young students with bare programming experience to program and interact with sensors and output devices. In the next section we will explain potential next steps.

8.2 Limitations and Future Work

As we explained in Chapter 1 "Introduction", our work focused on the user interface of a more complex system. This system will feature actual physical breadboards connected to the tablet so that users can use real sensors and real output devices instead of the virtual entities we used for our system. We explained in the previous chapter that our results can be used as the first steps from the software perspective towards such system. Our positive results can lay the foundation regarding both the general programming metaphor used for the app and also for the app design.

If connected with hardware peripherals, the interface could be dynamic and adapt to the used hardware components

Adding the corresponding hardware to this system does not only imply a connection to real breadboards, it can also affect the interface. For example the amount of input and output pins could also be dynamic and depend on the used breadboards and pins. Also, the depicted FBs in the top panel in our interface were static and independent. They could, depend on the actually connected devices on the breadboards. Particular hardware could lead to only certain FBs being displayed and signifying that these can be used with this particular hardware setup. This would result in a dynamic function block panel, which could also guide the user and physically constrain them. In a technical scope this could be achieved by either displaying only certain FBs or display the whole palette but only allow the interaction with certain FBs, and additionally use coloring to signify which can be used for the current hardware setup.

Only certain FBs could be displayed or others could be greyed out

A dynamic panel could also help with the screen size

This approach could also solve the problem of a limited area to display the FBs. As we discussed in Chapter 5 "High Fidelity Prototype", we were also limited with the screen size of the tablet and could not separate the function block types more distinctively. If only certain FBs would be displayed, more space would be gained and they could be separated more clearly visually. Another possible mechanism is to use an expansion menu like in Danado and Paternò [2014]. With this, there would be no concrete FBs anymore but instead abstract blocks representing certain groups of FBs. Clicking such abstract block would open up a menu from which the user could then choose the desired FB and drag it to the workspace. A different idea could be to allow users to customize the top panel and its elements but in our opinion this could lead to too many options for an inexperienced user who wants to learn.

Function block groups instead of concrete blocks could also help with space issues

A possible feature would also be the combination of function blocks to create new ones

One of the experienced participants stated that they would use it again if it was more complex and had more function blocks. In its current state, additional function blocks can easily be added by the developer. However, while creating new blocks on their own with new logic seems overwhelming for new users, we can imagine a feature of combining blocks, thus creating new blocks with already present logic. This also increases the ceiling [Myers et al., 2000, Papert, 1980], allowing experts to use advanced methods. Due to the amount of positive feedback regarding how the pupils liked being able to play around and find out things on their own, we can also see that the interface provides a low floor.

Further iterations could also evaluate redesign of input pins and different colors for the connections to avoid a *green* and *correct* association

Considering the criticism of some participants, some design and usability aspects could also be redesigned through further iteration cycles. Since the input pins caused trouble for a few participants, they should be redesigned to provide more affordance. One participant of our preliminary user study mentioned that they could also be semicircles, similar to the pins of the function blocks, to indicate that those are also pins. Due to time restraints we kept the design as it was for the study but this could be analyzed in next iterations. Furthermore, as we mentioned above, one participant stated that the coloring of the connections could be improved to distinguish more easily between digital and analog signals and also possibly refrain from using green

for connections as people can associate green rather with correct than with signals.

Studies could also be conducted with more pupils and pupils from other classes and schools

Regarding the evaluation phase, we think it would also help increasing the validity of our work if the studies would be conducted with more participants and different classes. In addition to this, our participants were all participating in a school laboratory which they attended due to their classes. Although we tried to reduce a potential Hawthorne effect, we think it could be further alleviated if the pupils would participate without a background activity in which they are forced to partake.

An additional code view could also help pupils to potentially learn code if they wanted to

We can also imagine that a code view could be implemented, similar to Millner and Baafi [2011], in which users edit the code and see the results in the graphical view, and vice versa. In their work, users can toggle between "representations as they see fit". One of our experienced participants also stated that they prefer code and although they are experienced, they also liked the concept and could imagine using it to try ideas. A code representation can also support the transfer knowledge of pupils. For example they could create a project in the graphical representation and toggle to the code view and try to understand what they did. A quick switch between the views can then help learning what code does and how it affects a program. Furthermore it could help them transfer their knowledge to code based systems in the long run.

Bibliography

Khuloud Ahmad and Paul Gestwicki. Studio-based learning and app inventor for android in an introductory CS course for non-majors. *SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, pages 287–292, 2013. doi: 10.1145/2445196.2445286. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84876262426{&}partnerID=tZOtx3y1.

S. Arakliotis, D. G. Nikolos, and E. Kalligeros. LAWRIS: A rule-based arduino programming system for young students. In *2016 5th International Conference on Modern Circuits and Systems Technologies, MOCAST 2016*, 2016. ISBN 9781467396806. doi: 10.1109/MOCAST.2016.7495150.

Tracey Booth, Simone Stumpf, Jon Bird, and Sara Jones. Crossed Wires: Investigating the Problems of End-User Developers in a Physical Computing Task. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 3485–3497, New York, New York, USA, 2016. ACM Press. ISBN 9781450333627. doi: 10.1145/2858036.2858533. URL http://doi.acm.org/10.1145/2858036.2858204{%}5Cnhttp://dl.acm.org/citation.cfm?id=2858036.2858533.

J.M. Carroll and M.B. Rosson. Cases as Minimalist Information. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, volume 00, pages 1–10. IEEE, 2005. ISBN 0-7695-2268-8. doi: 10.1109/HICSS.2005.135. URL http://ieeexplore.ieee.org/document/1385313/.

Jose Danado and Fabio Paternò. Puzzle: A visual-based

environment for end user development in touch-based mobile phones. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7623 LNCS, pages 199–216. 2012. ISBN 9783642343469. doi: 10.1007/978-3-642-34347-6_12. URL http://link.springer.com/10.1007/978-3-642-34347-6{_}12.

Jose Danado and Fabio Paternò. Puzzle: A mobile application development environment using a jigsaw metaphor. *Journal of Visual Languages and Computing*, 25(4):297–315, 2014. ISSN 1045926X. doi: 10.1016/j.jvlc.2014.03.005. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84904745412{&}partnerID=tZOtx3y1{%}5Cn{%}3CGotoISI{%}3E://WOS:000338482400004.

Sayamindu Dasgupta and Mitchel Resnick. Engaging novices in programming, experimenting, and learning with data. *ACM Inroads*, 5(4):72–75, 2014. ISSN 21532184. doi: 10.1145/2684721.2684737. URL http://dl.acm.org/citation.cfm?doid=2684721.2684737.

Dale Dougherty. The Maker Movement. *Innovations: Technology, Governance, Globalization*, 7(3):11–14, 2012. ISSN 1558-2477. doi: 10.1162/INOV_a_00135. URL http://www.mitpressjournals.org/doi/10.1162/INOV{_}a{_}00135.

Daniel Drew, Julie L. Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. The Toastboard. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*, pages 677–686, New York, New York, USA, 2016. ACM Press. ISBN 9781450341899. doi: 10.1145/2984511.2984566. URL http://dl.acm.org/citation.cfm?doid=2984511.2984566.

S Federici. A minimal, extensible, drag-and-drop implementation of the C programming language. *SIGITE'11 - Proceedings of the 2011 ACM Special Interest Group for Information Technology Education Conference*, pages 191–196, 2011. doi: 10.1145/2047594.2047646. URL http://delivery.acm.org/10.1145/2050000/2047646/p191-federici.pdf?ip=137.226.73.135{&}id=

2047646{&}acc=ACTIVESERVICE{&}key=
575DA4752A380C0F.4D4702B0C3E38B35.
4D4702B0C3E38B35.4D4702B0C3E38B35{&}CFID=
950613158{&}CFTOKEN=
56019362{&}{_}{_}acm{_}{_}=
1497965116{_}2593fa54b1a7562dcf64c6611ba6ed0b.

Gerhard Fischer. Beyond "Couch Potatoes": From consumers to designers and active contributors. *First Monday*, 7(12):20, 2002. ISSN 13960466. doi: 10.1109/APCHI.1998.704130. URL http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1010/931.

Minjie Hu, Michael Winikoff, and Stephen Cranefield. Teaching novice programming using goals and plans in a visual notation. *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123*, pages 43–52, 2012.

Yoshiharu Kato. Splish: A visual programming environment for arduino to accelerate physical computing experiences. In *8th International Conference on Creating, Connecting and Collaborating through Computing, C5 2010*, pages 3–10. IEEE, 2010. ISBN 9780769540290. doi: 10.1109/C5.2010.20. URL http://ieeexplore.ieee.org/document/5476940/.

Colleen M. Lewis. How programming environment shapes perception, learning and goals. In *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*, page 346, New York, New York, USA, 2010. ACM Press. ISBN 9781450300063. doi: 10.1145/1734263.1734383. URL http://portal.acm.org/citation.cfm?doid=1734263.1734383.

Tom Lieber, Joel R. Brandt, and Rob C. Miller. Addressing misconceptions about code with always-on programming visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2481–2490, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557409. URL http://doi.acm.org/10.1145/2556288.2557409.

Fabio; Wulf Volker Lieberman, Henry; Paternò. *[Human-Computer Interaction Series] End User Development Volume 9 —— End-User Development: An Emerging Paradigm*, volume 10.1007/1-4020-5386-X. 2006. ISBN 978-1-4020-4220-1,978-1-4020-5386-3. doi: 10.1007/1-4020-5386-x_1. URL http://gen.lib.rus.ec/scimag/index.php?s=10.1007/1-4020-5386-x_1.

John Maloney, Kylie Peppler, Yasmin B. Kafai, Mitchel Resnick, and Natalie Rusk. Programming by choice: urban youth learning programming with scratch. *SIGCSE '08 Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 367–371, 2008. ISSN 0097-8418. doi: 10.1145/1352135.1352260. URL http://dl.acm.org/citation.cfm?id=1352260.

Amon Millner. Computer as Chalk: Supporting Youth as Designers of Tangible User Interfaces. *Constructionism 2012*, pages 339–348, 2012. URL https://llk.media.mit.edu/courses/readings/Millner-Constructionism.pdfhttp://llk.media.mit.edu/courses/readings/Millner-Constructionism.pdf.

Amon Millner and Edward Baafi. Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects. *Proceedings of the 10th International Conference on Interaction Design and Children*, pages 250–253, 2011. doi: 10.1145/1999030.1999074. URL http://portal.acm.org/citation.cfm?doid=1999030.1999074http://dl.acm.org/citation.cfm?id=1999074.

Brad Myers, Scott E. Hudson, and Y Pausch. Present and Future of user interface software tools. *ACM Transactions on Computer-Human Interaction*, 7(1):3–28, mar 2000. ISSN 1073-0516. doi: 10.1145/344949.344959. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.9491.

Dan O'Sullivan and Tom Igoe. *Physical Computing: Sensing and Controlling the Physical World with Computers*. Thomson, 2004. ISBN 159200346X. URL http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20{&}path=ASIN/159200346X.

Stamatios Papadakis, Michail Kalogiannakis, Vasileios Orfanakis, and Nicholas Zaranis. Novice Programming Environments. Scratch & App Inventor. *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments - IDEE '14*, pages 1–7, 2014. doi: 10.1145/2643604.2643613. URL http://dl.acm.org/citation.cfm?doid=2643604.2643613.

Seymour Papert. *Mindstorms: Computers, Children, and Powerful Ideas*. Basic Books, 1980. ISBN 0-465-04629-0. URL http://dl.acm.org/citation.cfm?id=1095592.

Fabio Paternò. End User Development: Survey of an Emerging Field for Empowering People. *ISRN Software Engineering*, 2013:1–11, 2013. ISSN 2090-7680. doi: 10.1155/2013/532659. URL http://downloads.hindawi.com/journals/isrn.software.engineering/2013/532659.pdf{%}5Cnhttp://www.hindawi.com/journals/isrn/2013/532659/{%}5Cnpapers3://publication/doi/10.1155/2013/532659{%}5Cnhttp://www.hindawi.com/journals/isrn.software.engineering/2013/532659/.

Mitchel Resnick. Sowing the Seeds for a more Creative Society. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, page 30, New York, New York, USA, 2009. ACM Press. ISBN 9781605582467. doi: 10.1145/1518701.2167142. URL http://dl.acm.org/citation.cfm?doid=1518701.2167142.

Mitchel Resnick and Brian Silverman. Some Reflections on Designing Construction Kits for Kids. In *Proceeding of the 2005 conference on Interaction design and children (IDC '05)*, pages 117–122, New York, New York, USA, 2005. ACM Press. ISBN 1595930965. doi: 10.1145/1109540.1109556. URL http://dl.acm.org/citation.cfm?id=1109540.1109556.

Mitchel Resnick, Brian Silverman, Yasmin Kafai, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, and Jay Silver. Scratch. *Communications of the ACM*, 52(11):60, 2009. ISSN 00010782. doi: 10.

1145/1592761.1592779. URL http://portal.acm.org/citation.cfm?doid=1592761.1592779.

Mona Rizvi, Thorna Humphries, Debra Major, Meghan Jones, and Heather Lauzun. A CS0 course using Scratch. *J. Comput. Small Coll.*, 26(3):19–27, 2011. ISSN 1937-4771. URL http://portal.acm.org/citation.cfm?id=1859159.1859166{&}coll=GUIDE{&}dl=GUIDE{&}idx=J420{&}part=affil{&}WantType=Affils{&}title=JournalofComputingSciencesinColleges{&}CFID=102938954{&}CFTOKEN=39012726{%}5Cnhttp://portal.acm.org/ft{_}gateway.cfm?id=1859166{&}type=pdf{&}coll=GU.

Ben Shneiderman. The future of interactive systems and the emergence of direct manipulation†. *Behaviour & Information Technology*, 1(3):237–256, 1982. ISSN 0144-929X. doi: 10.1080/01449298208914450.

K.N. Whitley and Alan F. Blackwell. Visual Programming in the Wild: A Survey of LabVIEW Programmers. *Journal of Visual Languages & Computing*, 12(4):435–472, 2001. ISSN 1045-926X. doi: DOI:10.1006/jvlc.2000.0198. URL http://www.sciencedirect.com/science/article/pii/S1045926X00901988.

Index

Typeset September 27, 2019