
CuriO

A multi-sensory approach
to accessible programming

Design of Human Computer
Interface for blind children to
learn programming languages

Master Thesis
Krishna Thiruvengadam Rajagopal

CuriO

A multi-sensory approach
to accessible programming

Design of Human Computer Interface for blind children to learn programming languages

Master Thesis

Krishna Thiruvengadam Rajagopal

Master's Thesis
November 2019

Krishna Thiruvengadam Rajagopal

Master of Science
Integrated Product Design
Annotation in Entrepreneurship
Faculty of Industrial Design
Engineering
Delft University of Technology



Chair
Prof. dr. Ianus Keller
Department of Industrial Design
Delft University of Technology

Mentor
Prof. dr. Wilfred van der Vegte
Department of Design Engineering
Delft University of Technology

External Mentor
Prof. dr. Felienne Hermans
Head of Programming
Education Research Lab
Leiden University

Executive Summary

As the modern world progressing towards digital technologies, there is an increasing need for school students to learn programming skills so that they fit into the labor market now and in the future. Tangible programming toys and block-based languages are popular among younger children to learn to code. Students in middle and high school in several countries have begun to learn textual programming education as a step towards successful careers and educational opportunities. However, due to the visual nature of these materials and tools, they are inaccessible to blind students. Also, the accessibility aids used for interacting with computers in the first place are not accessible when it comes to programming. Braille displays are not accessible by most blind users as the braille literacy rates are falling, while screen readers do not read source code. Also, attempts to teach code using screen readers despite challenges kills the student's enthusiasm. The project proposes a design approach to multi-modal human-computer interaction to reduce sensory and cognitive overload to enhance learning outcomes for this user group. The design goal was to develop an affordable audio-tactile interface, to help blind high school students to learn textual programming languages. Research insights were gathered from literature research, surveys, interviews, observation studies, and co-creation sessions with blind students. The insights were synthesized to form ideas, and concepts were explored. The final concept was then taken to a level of functional prototype and was validated in simulated tests. A sustainable business model was also developed to take the project forward as an open-source hardware venture.

Acknowledgement

The work was made possible due to the support and contributions of many and I would like to thank every one of them

To my supervisors, **Ianus** and **Wilfred**, for guiding me throughout this project and allowing me to stay ambitious and at the same time practical.

To my external mentor **Felienne Hermans**, for letting me work as part of the amazing research group at PERL and providing feedback.

To **Anna van der Meulen** in helping me with research and putting me in touch with experts in the Netherlands

To **Parham, Santiago, Yuma** and **Pawal** for providing valuable insights about your life and work

To **Richard Bekking**, for his time, motivation and technical contributions to this project and be more than a teacher to me.

To **Venkatagiri Ramesh** and **Shashank Roy Choudhary** to stay up days and nights and lend their technical expertise.

To my friend **Jose Carlos Urrea** for guiding me throughout the Open Hardware exploration

To my partner in crime, **Shreyas Prakash**, who was all ready for all world changing ideas to work on

To my mother **Vasuki Rajagopal**, sister **Madhura Rajagopal** and father **Rajagopal Thiruvengadam** for supporting me through tough testing times

To my friends **Naveen Rajasekhran**, **Vinayak Krishnan**, **Prasad Gonugunta**, and **Pranav Suvarna** for being there always like a second family in Delft to fall back on.

Contents

| | |
|------------------------------------------------|----|
| Executive Summary | iv |
| Acknowledgement | v |
| Introduction | 1 |
| | |
| 1. Research | 2 |
| Research | 3 |
| 1.1 Literature Research | 3 |
| 1.2 Expert Interviews | 6 |
| 1.3 User Interviews | 7 |
| 1.4. User Observation | 8 |
| 1.5. Co-creation session | 10 |
| | |
| 2. Synthesis | 12 |
| Synthesis | 13 |
| 2.1 Design Goal. | 13 |
| 2.2 List of requirements | 14 |
| Conclusion | 15 |
| | |
| 3. Ideation | 16 |
| Ideation | 17 |
| 3.1 Functional Analysis | 18 |
| 3.2 Creative Session | 19 |
| 3.3 Morphological Chart | 20 |
| 3.4 Concept Development | 21 |
| 3.5 Final Concept | 23 |
| | |
| 4. Detailing | 24 |
| Detailing | 25 |
| 4.1 Detailed Design: A quick preview | 26 |
| 4.2 Aesthetic Exploration | 27 |
| 4.3. Embodiment | 29 |
| 4.4 Control System: | 34 |

| | |
|-------------------------------------------|----|
| 4.5 System Architecture | 36 |
| 5. Validation | 39 |
| Introduction | 40 |
| 5.1 Usability Test | 41 |
| 5.2 Aesthetic Perception | 42 |
| Conclusion | 43 |
| 6. Business Model | 44 |
| Introduction | 45 |
| 6.2. Customer Segmentation | 46 |
| 6.3. Open Source Hardware model | 46 |
| 6.1. Market Analysis | 46 |
| 6.4. Revenue Model | 47 |
| 6.5. Cost Structure | 48 |
| 6.6. Promotional Strategy | 48 |
| 6.7. Financial Plan | 48 |
| 7. Conclusion | 49 |
| 7.1 Recommendations | 50 |
| 7. 2. Next Steps | 51 |
| 7.3 Reflections | 53 |
| 7.4 Conclusion | 54 |
| References | 55 |
| Appendix | 60 |

Introduction

Last winter, I had the opportunity to attend SPLASH 2018, a conference on programming education in Boston (USA), along with Dr. Felienne Hermans, Head of PERL and associate professor at Leiden University. One of the talks about Audio Programming Languages for the blind, really fascinated me. Never, I had imagined that visually impaired people took up programming as much as sighted people do. During discussion with some of the participants, I realised the pressing challenges in accessible programming such as navigation and skimming. Little did I know then that this fascination and inquisitiveness, will lead me to complete a thesis on this topic. But I must say the whole journey has been fulfilling so far. A genuine concern and desire to contribute to the visually impaired community, has motivated and kept me going through the ups and down.

I started the project, as a part of the Programming Education Research Lab (PERL) within the topic of 'Inclusive Programming Education: The accessibility of existing programming materials for visually impaired children'. Though initially, the proposal was to investigate programming materials for primary education, I realized how important was it for high school blind students as well, to get access to tools to help in learning programming skills for their prospects in the job market. This way, my journey into developing the project, dubbed as 'CuriO' started.

1. Research

- 1.1. Literature Review
 - 1.2. Expert Interviews
 - 1.3. User Interviews
 - 1.4. User Observation
 - 1.5. Co creation session
-

Research

The project started with an early exploration, starting with Dr. Felienne Hermans' proposal 'Inclusive programming education: The accessibility of existing programming materials for visually impaired children', that her team at the Programming Education Research Lab (PERL) have been researching. The proposal explains about the relevance of programming education for visually impaired children and how most of the current materials are only oriented towards sighted children. Further reading online, out of inquisitiveness, led to blog posts by blind programmers and quora articles, where a number of people had queried how is it even possible for a blind person to code. A blog written by an Amazon professional (Foranzo, 2018), gave the first insights on how a blind person can not only code, but become a professional programmer.

Eager to get to know more about the life of a blind person and to understand where the gap lies, friends were reached out. This led to a casual skype call with Kailash Tandal, a blind PhD (economics) student at the Tata Institute of Social Sciences (TISS), who explained about the aspirations to feel normal among his peers, the various challenges a visually impaired person faces in everyday life, right from inaccessible restaurant menus, to having to depend totally on others while travelling around. When asked about braille letters written in public commute places, the first surprise came, that not all blind people can read braille, which challenged the earlier understanding or personal 'assumption' that every blind person can read braille. Another family friend, Dr. Sriram from the Aravind Eye Hospital, put me in touch directly with blind schools in India. A talk with Gopi, a blind computer science student at the American College (Madurai,

India) revealed how he wished he could learn programming languages, while his friends can code fluently. Around the same time, incidentally an email was circulating in DevCreative, a google group in India, from Youth4Jobs, a training institute that was seeking candidates to teach HTML to blind youngsters in New Delhi (India). A skype meeting with its co-founder, Ankit Bansal gave new insights on how programming education could positively impact visually impaired youngsters in a developing country like India.

This early exploration helped formulate the following research question:

Despite the fact that some visually impaired programmers exist, what are the major challenges that still makes programming education inaccessible to larger population of students, towards mainstream career opportunity?

1.1 Literature Research

To get a deeper understanding of the challenges in programming education, for a visually impaired user, literature research was done. First the inaccessibility problems of existing programming education materials for primary and middle school children were explored, followed by the challenges with learning textual programming languages (for employability).

According to the World Health Organization (WHO, 2014), there are 2.2 Billion people with some form of visual impairment (VI) and blindness globally. Of these, 30 million are in geographical Europe and 320,000 visually impaired and 45,000 blind in the Netherlands alone. (Vision 2020, 2005).

With the modern world progressing towards digital technologies, there is an increasing need for students to learn programming skills so that they fit into the labor market now and in the future. Tangible programming education materials like LEGO Mindstorms and BBC Micro: bit and programming languages like Scratch and Blockly are mostly block-based (Glinert, 1986) and popular among younger children, to learn programming concepts. Textual programming languages like Python and Java are introduced at high school. Several countries have already begun to incorporate programming education as part of their mainstream school curriculum. The programming skills they learn enable them to get employment opportunities or pursue higher education in computer science.

However, due to the visual nature of these programming tools and materials, it is mostly inaccessible by the visually impaired and blind students. Most of these materials use visual information to convey cues about problems in execution (Zuckerman et al., 2006) and program output. In the current scenario, these students are largely underrepresented in computer science education (Stefik et al., 2011). Computer programming is an accessible career path for the blind (Siegfried, 2006) because they enable a blind person to work from home. Despite the rapid rise in the number of job vacancies that require programming skills in recent years (Codingjojo, 2019), the employment of visually impaired is far lesser than the sighted population (Goertz et al., 2010).

Education materials like Microsoft Torino Project (Vilar et al., 2019) and Royal National Institute of Blind People's adaptation of Apple's Swift Playgrounds (AppleVis, 2019) are working towards making a difference in teaching coding basics to young blind students. However, a transition to textual programming

languages as part of programming education materials is essential (Moors, Denny, Reilly, 2018) for high school and vocational training students for gearing up for the job market. Accessible textual programming languages like Quorum (Stefik, 2017) and Bootstrap (Schanzer, 2013), are primarily designed for students with visual disabilities. They use audio feedback and belong to the family of Audio Programming Languages (APL). However, these languages act as a starting point for blind individuals to learn textual programming but are not (yet) used in professional development environments. (Quorum, personal communication, 2019). Although large corporations like Google, IBM, and Microsoft employ visually impaired programmers, they do not represent a subpopulation but are the few who have managed to code despite challenges. Many modern IDEs are based on Graphical User Interfaces (GUIs) and include tools like syntax highlighting, variable watch windows, and ability to execute program forward and backward, to improve the productivity of the user (Pothier et al., 2007). Due to the visual nature of such features, they are completely inaccessible to a blind programmer.

To be able to interact with a computer in the first place, visually impaired individuals depend on braille displays and screen reading software. Due to increasing braille illiteracy, screen readers are gaining prominence among the blind community, as the most preferred Human-computer accessibility tool. For instance, of the 1.3 million legally blind people in the United States, fewer than 10% are braille readers (National Federation for Blind, 2009). The primary reason attributed to high braille illiteracy levels is the difficulty faced by any adventitious (late) blind individual to comprehend subtle tactile information as in Braille (Rex, 1989; Schroeder, 1989; Stephens, 1989). Also, reading speeds matters. The students who manage to

learn to read braille (born blind and some late blind) mostly do not attain the reading speeds that are required for employment (Ferrel, Mason, Young & Cooney, 2006).

For instance, a sighted user scrolls through the code to get an overview, but the blind user needs to read the whole content line by line, with a screen reader. This forces blind programmers to keep track of details such as current brace levels (Armaly, 2016). Also, in programs like Python, the blind user listens to sounds that indicate the number of whitespaces to perceive indentations in a code structure (Potluri, 2018). Screen readers are designed to read natural language, that is words and sentences. Not source code. Blind programmers still find it better to use text-based editors and compile their code using the command line, than using GUIs (Potluri, 2018). However, to work in organizations along with sighted colleagues requires a visually impaired user to use Integrated Development Environment (IDEs).

In recent years, there is an increasing interest in making mainstream environments accessible to visually impaired users. StructJumper, an Eclipse plugin, enables users to access a tree-view structure with respect to a line and help them understand the context of that line (Baker et al., 2015). Another Eclipse plugin features usable forms of tree-view structures, as a non-visual means to navigate through code (Smith et al., 2004). Sodbeans is a plugin that adds audio feedback to the NetBeans environment (Stefik et al., 2009). The Wicked Audio Debugger (WAD) is a debugging tool for Visual Studio, which helps users to comprehend the dynamic program behavior like control flow (Stefik et al., 2007). Musical Layers have been used to represent program structures using metaphorical sounds through an approach called Layered Auralization (Stefik et al., 2006). InfoSound, an auralizer program, features everyday

sounds and speech to indicate abstract events in parallel processing within a program (Sonnenwald et al., 1990).

However, using only a single modality such as audio-based feedback have their share of challenges, especially when it comes to blind students who are new to learning to code. A user receiving sensory information from only one modality and cognitively process it, can overload that modality (auditory), because of the limited working memory of the human brain (Lay-Ekuakille et al., 2010). After a while, users may be limited in the perception of acoustical signals coming from assistive devices. As Cognitive Load Theory suggests, learners should be able to process information elements and their interactions simultaneously before they begin to learn a new concept. (Paas, Renkl and Sweller 2003; Sweller 1988, 1999). The human information processing works by storing new incoming information in their short term working memory and encoding it to permanent memory for problem-solving. However, the short term memory is limited to 7 (+/- 2) chunks of information (Baddeley, 1994) before forgetting the information perceived (Miller, 1956). Since screen readers linearize content (WebAim, 2019), they do not help users to chunk information like phonemes, graphemes, morphemes, and syntactic structures and semantic interpretations of code, making it difficult for the user to comprehend the information. (Sweller, 1988). As cognitive load increases, the user's working proficiency breaks (Moreno & Mayer, 2000), resulting in frustration in learning a new textual programming language. Multi-modal interactions, such as Audio-Haptic feedback, help users better perceive spatial information and aid in their cognitive processing (Rice et al., 2005). The ability of a user to know where the tactile exploration should be directed to when moving through a large piece of text has been shown to increase

with peripheral attention to physical features. (Rayner, Foorman, Perfetti, Pesetsky, & Seidenberg, 2001). Audio guided Tactile maps to teach geography (Brock et al., 2005) and Audio Supported Reading (ASR) of Braille (Jackson, 2012) have successfully incorporated Audio- tactile information processing in educational materials. Hence in order to facilitate a blind student to learn textual programming languages, it becomes essential to represent code structures as audio-haptic tangible interactions.

Insights:

From the literature research, the major challenges in learning a textual programming language were found not only due to a lack of accessible software tools but that there was also a need for a new approach to Human-Computer Interface, in order to facilitate programming education. Some main insights were:

1. Need to switch to a textual programming language is essential for high school blind students to get higher education or employment opportunities
2. Blind users, including professional programmers, find it hard to skim through code, navigate using a screen reader or a refreshable braille display as the only accessibility interfaces
3. Multi-modal (Audio- tactile) interaction is essential for blind users to acquire new information (like a new programming language) without overloading sensory and cognitive modalities.

1.2 Expert Interviews

With the bachelor's research insights as to the starting point, more insights were gained from two experts working with visually impaired children. It was decided to interview them with an understanding that teachers will have first hand

experience in using existing accessibility products to teach students, and their insights could be useful in furthering the research. The experts were from the Royal Visio and Bartimeus, the largest two umbrella organizations in the Netherlands, under whom many special schools for the visually impaired operate. Anna van der Meulen, a postdoctoral researcher with Dr. Felienne Hermans (PERL), was supportive and helped me set up interviews with the experts. A semi-structured interview in Dutch was undertaken, with the support of Benjamin Bosdijk, a then bachelor's student in the PERL group.

Don van Dijk

The first interview was of Don van Dijk, a teacher at the Visio school in Grave. The entire transcript can be found in appendix 1.3. In his words, "The school introduces programming concepts at Kindergarten in the form of block-based programming with micro bits. However, it is difficult to visualize for blind students using such toys. When blind children come to middle and high school, they are taught python language. Some investment is required, but it is accessible to some level. Navigation is still missing. Students can always have workaround strategies, but it kills their enthusiasm. The most common challenge is navigating from one place to another in a coding language." Taking about the behavior of children towards learning, he said they are very eager to explore new technologies, once they are taught where the buttons are and how to get started. He said that the visually impaired community is happy to have so many technologies to provide assistance to them, compared to what existed ten years ago.

Maaïke Meerlo:

The second expert interviewed was with Ms. Maaïke Meerlo, an Innovator and Consultant at the Bartimeus Fablab in Verbrede School (Doorn, Netherlands). She revealed how often design solutions.

do not fulfill user needs as many visually impaired students usually have other (multiple) disabilities as well, and how narrowing down to one type of visual impairment was necessary, as people with different types of visual impairments have entirely different needs. She also explained about the different opportunities for blind students, as the municipalities provide incentives to the employers. She also gave me a tour of the school infrastructure to give an understanding of how it was adapted to needs of a visually impaired child.

Upon observation of how children in that school live their everyday life, it was observed that every place within the premises had been modified to be accessible by all types of visual impairment. Each room was fitted with tiles with different textures to indicate it was a classroom, bathroom, or kitchen. The entire hallway fitted with wooden railings to guide along the path. Doors were opened with an access card and not keys.

Insights:

1. Navigation is a major challenge in textual languages like python, in high schools. Though students managed to find coping strategies, it killed their enthusiasm.
2. Students are open to new technologies and learn them quickly
3. It was essential to narrow down from visually impaired, which is a broader spectrum to design for. So from this point, it was decided to focus only on complete blindness.

1.3 User Interviews

Right from the start of the project, user

interviews were considered necessary as it will reveal the more in-depth challenges and needs of a blind individual. This was a necessary step to make the design solution as human-centered as possible.

Professional blind programmers were reached out specifically to gain more an understanding of how they manage to code in a professional setting despite accessibility challenges, and because their strategies could inspire creative solutions for latter design processes. Though their needs, aspirations, and behavior towards programming would be different from that of high school children, it was assumed that they would still represent the blind community and have similar challenges.

The participants were contacted through blogs, LinkedIn, and through personal connections. Four blind programmers from Poland, Australia, and India, participated in the interviews. All interviews were carried out through skype and were recorded for post-interview analysis. The condensed interview transcripts can be found in Appendix 1.1. The following are quotes from the interviews:

“Braille is very useful for me, it helps me feel specific information especially when reading an e-book, but it takes a lot of time to code with it” - Yuma Decaux, a professional blind programmer (Australia)

“Did you just say skimming? It is not possible in coding. I ask my peers or read, line by line. Also, NVDA makes a different tone of beeps for different indents. This will drive you crazy, especially when you are coding for like 10 hours.” - Pawel Urbanski, professional blind programmer and Employee consultant (Poland)

The user interviews revealed that blind programmers did develop workarounds or coping strategies, to learn programming languages with accessibility aids like Braille

displays or screen readers. The challenges did not make it impossible to learn a new language, but made it harder to believe for themselves that programming is a good career path for the blind community. One participant told how he thinks programming education is just not for the blind, and making them undergo the challenges is making them only frustrated. This answers why professional blind programmers exist, but still largely inaccessible by other blind individuals.

Insights:

1. Users find a code abstract when a screen reader, goes through it line by line, without being able to skim.
2. When there are many classes and functions in a program, it becomes necessary for a blind user to mind map all of them.
3. Another challenge with screen reader is its incompatibility with many IDEs and text editors. In IDEs it does not provide live feedback, but only on compilation.
4. Braille displays are useful to feel the code (tactually), but does not convey information on hierarchies, making it essential to create mind maps instead.
5. Because braille displays are expensive it becomes difficult for users especially in developing countries to access them.

1.4. User Observation

The literature research and the skype interviews provided an understanding of the challenges of a blind programmer. However, to get a more in-depth and holistic understanding of how they interact with the tools and accessibility aids and its usability problems, users were observed in their context. The observation of their interaction with other products like the coffee machine, smartphone, vacuum cleaners served as a design research tool and as an inspiration for ideation later on.

Initially, Visio and Bartemeus schools were reached out to perform observation studies of blind high school students and to set up interviews with the computer science teachers. However, due to time constraints and confidentiality aspects of this vulnerable group of children, the requests were turned down.

Parham Doutsdar, an Iranian blind programmer, working at Booking.com in Amsterdam, accepted the request for the observation study. The different coping strategies in everyday life were observed, right from navigating around his home, making coffee, filling hot water in a vessel, cleaning the house, cutting vegetables, and cooking. Tactile indicators such as temperature, moisture, landmarks within a room were used to identify its relative distance from another object of reference. The participant preferred to place objects in the house in a specific place. This way, the relative positions of other objects were not missed. Other stimuli such as smell were used to identify different rooms and sound of cracking to determine if a piece of bread has been toasted, for instance.

Then, the participant explained how code is written using his accessibility tools. Voice Over screen reader (in Mac) was used to write code. During this, the participant paused to check if any WhatsApp messages had been received and showed how smartphones are used. The Voice Over app also worked on the iPhone, apps were navigated with right swipes, then double tapped to go into the selected option. The app reads out all the information on the screen. It was observed how every line had to be read to understand who had sent the messages. It was interesting to see how he had his laptop screen turned off initially before starting the demonstration. The screen reader sounded like gibberish. It was too fast to comprehend. The reading

speed was then slowed so that it was easier to understand what is going on.

blind students to reuse code blocks from the internet to learn a new programming language or develop projects.

A sample python program for Fibonacci series was provided through a git repository. The plan was to observe how the participant uses a screen reader to skim and navigate through the code to get an overview. The file was cloned into Emacspeak, a text editor with a speech engine that reads out lines on the screen. The participant was asked to read aloud whenever the context of the line was understood. Keyboard shortcuts were used to navigate around the code, and the lines read one by one. It was necessary for the participant to go back and forth to understand the whole logic. Six minutes later, the logic behind the code was explained. The key challenges in the workplace as a professional programmer was explained. Reading what other sighted colleagues wrote was difficult to read. It was necessary to remove camel casing and abbreviations so that the screen reader can read what function, for instance, is written in the code. Since it is difficult to skim through longer codes, assistance is sought from other sighted colleagues to explain to him what the code is about and how the functions have been called.

Insights:

1. Reading others code is difficult, because how others call the function, for instance is too abstract for a blind user to understand, or the screen reader is going to read it strangely. For instance, nth term is difficult to be read by a screen reader.
2. It is important to remove camelcasing and abbreviations before a code can be read by a screen reader.
3. It is difficult for a screen reader to read code blocks from online platforms like GitHub, where not all code is properly documented, or written with a standard coding style. This means it is difficult for

1.5. Co-creation session

To get to know the nuances of how blind people, especially adolescents and young learners, process large pieces of information cognitively, co-creation session was conducted at a blind school under the National Institute for Visually Disabled (NIVD) in Chennai, India. Co-design processes allow the end-users (blind students) as the experts of his/her own experiences and make them key providers of knowledge for a designer (Sanders, Stappers, 2008)

21 blind and low vision students between the age group 16 to 22 years participated in the session. This group had intermediate experience in using NVDA screen readers and basics in HTML programming. A novice group in programming experience was selected to simulate the experiences of students new to learning a programming language.

The session started with an ice-breaking fun activity, and then the rules and challenges of the game were introduced. They were grouped into five teams and were given board each per team. Each board had 121(11 x 11) tactile alphabet tiles (see figure 1), with a chamfer on each tile (for orientation of up-side). The tiles were pre-arranged to form a jumbled word puzzle. To gamify the session, the word puzzle game was timed, and the teams competed against each other.

The session was divided into two phases- In the first phase, it was observed how blind students skimmed through large amounts

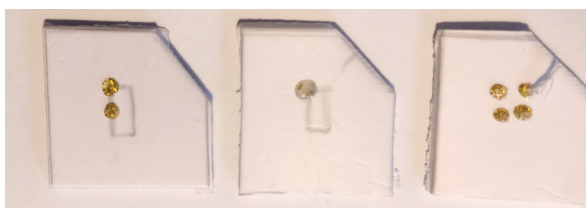


Figure 1 Foam Tactile tiles with embossed braille alphabets (Left to Right: B, A, G)

of characters to find the words (see figure 2). The rationale behind the word puzzle was to mimic the way blind users have to scan through code (with many characters) with braille displays or screen readers, on a computer. The task was to find sensible character combinations (words) among the 121 character word puzzle.

In the second phase, each team rearranges the tiles to form its own puzzle. The new puzzle is passed onto their neighbor. The teams now had to quickly skim through the puzzle and find the words, competing against each other. They were encouraged to come up with new ideas or strategies to complete the puzzle faster. A box of assorted (safe to use) materials were kept, for them to build mockup tools.

Co-creation Results

The second phase also acted as a co-creation method to derive inspiration from the prototypes participants built. Team C completed the puzzle first followed by Team A. When asked to reflect and demonstrate their prototype, participants from team C explained their idea, which was to fix one end of a string to one of the corners of the board and then with the other string end tied to a braille pointer (commonly used to punch braille paper), read braille character, and with the length of the string (slack or in tension) measure the relative distance. This helped their team to read relative

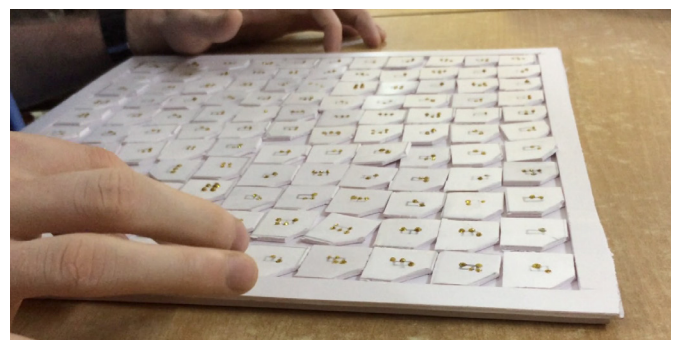


Figure 2 A participant skimming through the word puzzle

distances faster than just (cognitively) perceiving the distances using two hands/ Their mockup was also tested with other teams, and the rate of task completion was found to be more or less similar.

Another team came up with the idea of using physical markers while reading quickly through the tiles. In her words (translated from Tamil), “ It is very difficult to remember what words I read previously as I move to other words, and then I had to read from the start.” So their team came up with an idea to place foam blocks every time they read a word and then formed a story as they read each new word. So they remember the logic rather than individual words.

Insights:

1. Blind participants found it essential to use Physical Markers as cognitive landmarks while solving a problem.
2. It was important for an user to perceive the relative distances (corelations between physical representations) between the last read information and upcoming information to increase their comprehension rates. This is referred to as Allocentric Processing of Haptic space (Kappers, 2007)
3. Metaphors for information was needed to help in working memory, while reading through large amounts of information

Conclusion

The research phase provided insights into the challenges provided by the current accessibility aids like braille displays and screen readers. The reasons why programming is not yet accessible as a mainstream career opportunity were also gained. From this point, two major decision was taken,

1. The scope of the project will be about blind students and not cover the entire spectrum of visual impairment.
2. Since only a fraction of the blind population read braille in the current scenario, the further design solution exploration will be independent of braille

2. Synthesis

2.1 Design Goal

2.2 List of Requirements

Synthesis

The insights from the research phase were synthesized to form definitive design goals and a list of requirements. The design goal acted as an overarching guide for the whole design process, while all the design decisions were based on the list of requirements. A Technology Readiness Matrix (Appendix 2.2) was also developed to set the scope of the project in terms of what level of functionality needs to be achieved. A fully functional prototype in a simulated environment (Level 6) was aimed for, as this would allow user testing and take the insights forward for research purposes. It was based on factors like time limitations, work complexity, and relevant skills required to achieve the design goals. However, later, during the design process, real-time interfacing had to be developed to the level of proof of concept to facilitate the test demonstration skimming and navigation.

2.1 Design Goal

To enable high school blind students to acquire the necessary skills, so that they find successful mainstream programming careers, it is essential to provide accessible tools. In order to facilitate in this learning process, it is important to consider a multi-modal interaction device. Hence the design goal was formulated as:

To develop an affordable audio-tactile human-computer interface, to help blind users to learn textual programming languages.

Focus Areas:

Three areas were defined to be the scope of the project. It was defined based on what was found as the major challenges in programming from research insights:

F1: Skimming: Ability to get an overview of code structures and program logic, by glancing through

F2: Navigation: Ability to move the read of a desired line, word or character and also to move the cursor there

F3: Scalable and Affordable: Ability of the solution to be affordable, inclusive of people in developing countries and develop a go-to-market strategy that allows larger scalability and market accessibility to the end user.

2.2 List of requirements

The list was continually iterated based on the learnings from the different design phases. The requirements were consulted every time a design decision was made. Through the design process, the list was also updated whenever new insights were found. Requirements such as safety and reliability were added, in addition to insights from the research phase as they were considered important for the end users.

R1. Safety

R1.1 The forces of any moving part should be less enough not to cause any physical injury to the user.

R1.2 The product should not have sharp edges and holes/ grooves more than 10mm (Kima 1993), to avoid finger injury when the user interacts with the product

R2. Reliability

R2.1 Housing should be robust- internal components should not be affected by environmental factors such as temperature and dust

R2.2 Tethering with external devices should be based on physical wiring, than using Bluetooth to avoid unexpected connectivity problems

R2.3 The overall product experience should reflect reliability and trust. This means there should be no failures in functionality.

R3. Functionality

R3.1 The product should be able to convey code structures through different sensory modalities.

R3.2 The product should allow users to move to a specific character in the code

R4. Interaction

The requirements for interaction were based on a combination of Nielsen's Heuristics (1994) and Tognazzi's Principles of Interaction Design.

R4.1 The elements of the product should be positioned in such a way that it allows the user to perceive the relative distances from a fixed point of reference to seek the position of any other element

R4.2 Interaction models should mimic physical interactions that users are familiar with (Skeumorphic)

R4.3 Navigational buttons or keys in the product should be metaphorically related to reading styles. eg. reading from left to right

R4.4 The product should have physical buttons and interface outlets in the standard positions that follow the mental models of a user. Eg. power switches in the top right

R4.5 The product should allow users to know if their interaction was successful, through some means of sensory indication during the product operation, that it is working.

R4.6 The product should allow recognition and not recall of information.

R4.7 The product should have perceptual cues (markers) so that the user can pause his activity and get back without losing on the activity.

R4.8 The product should allow clear indication which are the areas of interaction.

Conclusion

The points mentioned in this chapter were revisited back and forth throughout the design process. The next chapter deals with how ideas and concepts were generated using the above goal and requirements.

3. Ideation

3.1 Functional Analysis

3.2 Creative Session

3.3 Morphological Analysis

3.4 Concept Development

3.5 Final Concept

Ideation

The ideation phase resulted in a final concept that was taken to the next stage of detailing. First, the insights from the observation session were used to form functions required in the product to be designed using the functional analysis method. Then, these functions were used as starting points for a creative session, where various ideas were generated and segregated using ideation techniques. The feasible ideas were then taken as solutions (component) to each of the functions defined earlier, and different concepts were created using the morphological chart method. Simple Wizard of Oz prototypes were made to validate and evaluate these concepts quickly. Finally, a concept was selected using the Weighted Evaluation method. Every decision made was reflected back and forth with the design goal in consideration.

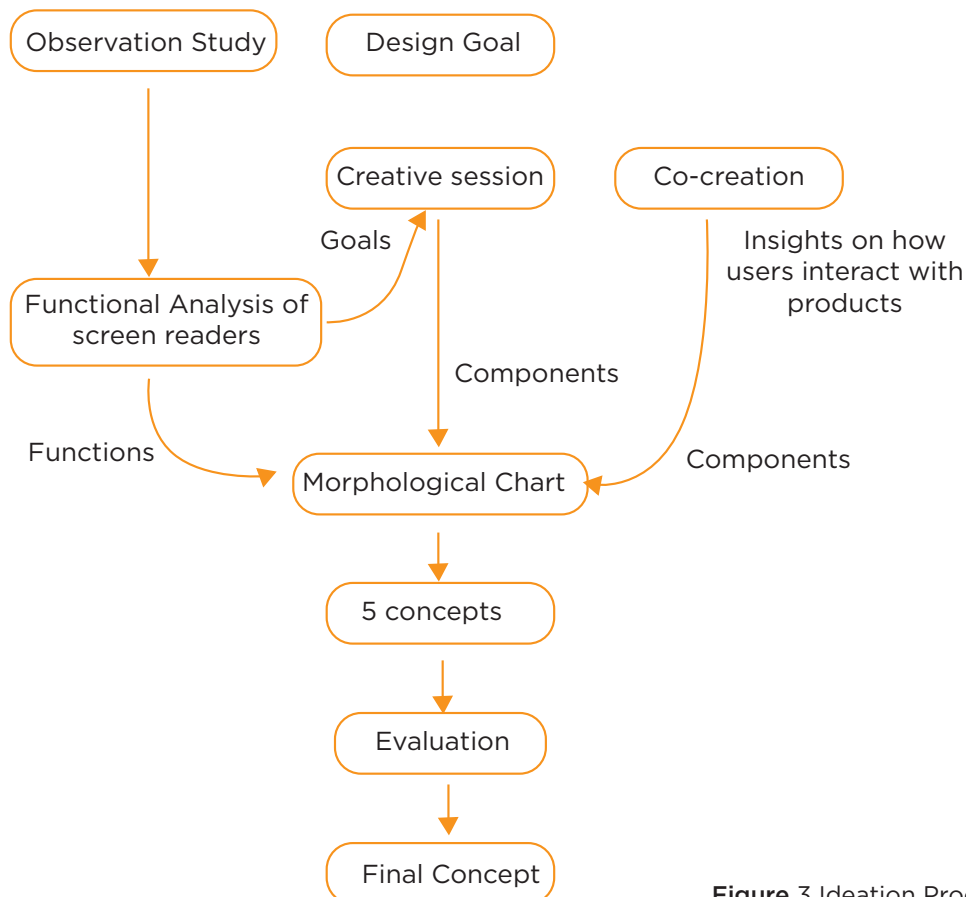


Figure 3 Ideation Process

3.1 Functional Analysis

The user observation and online survey of blind programmers (Appendix 1.3) led to insights on how a screen reader was used as a means to work on a code. The process of writing and reading code was mapped as a process tree (Roozenburg, 1995) (see figure 4). Functions required to enable this usage process were identified. These were taken as the starting points for the creative session and morphological charts.

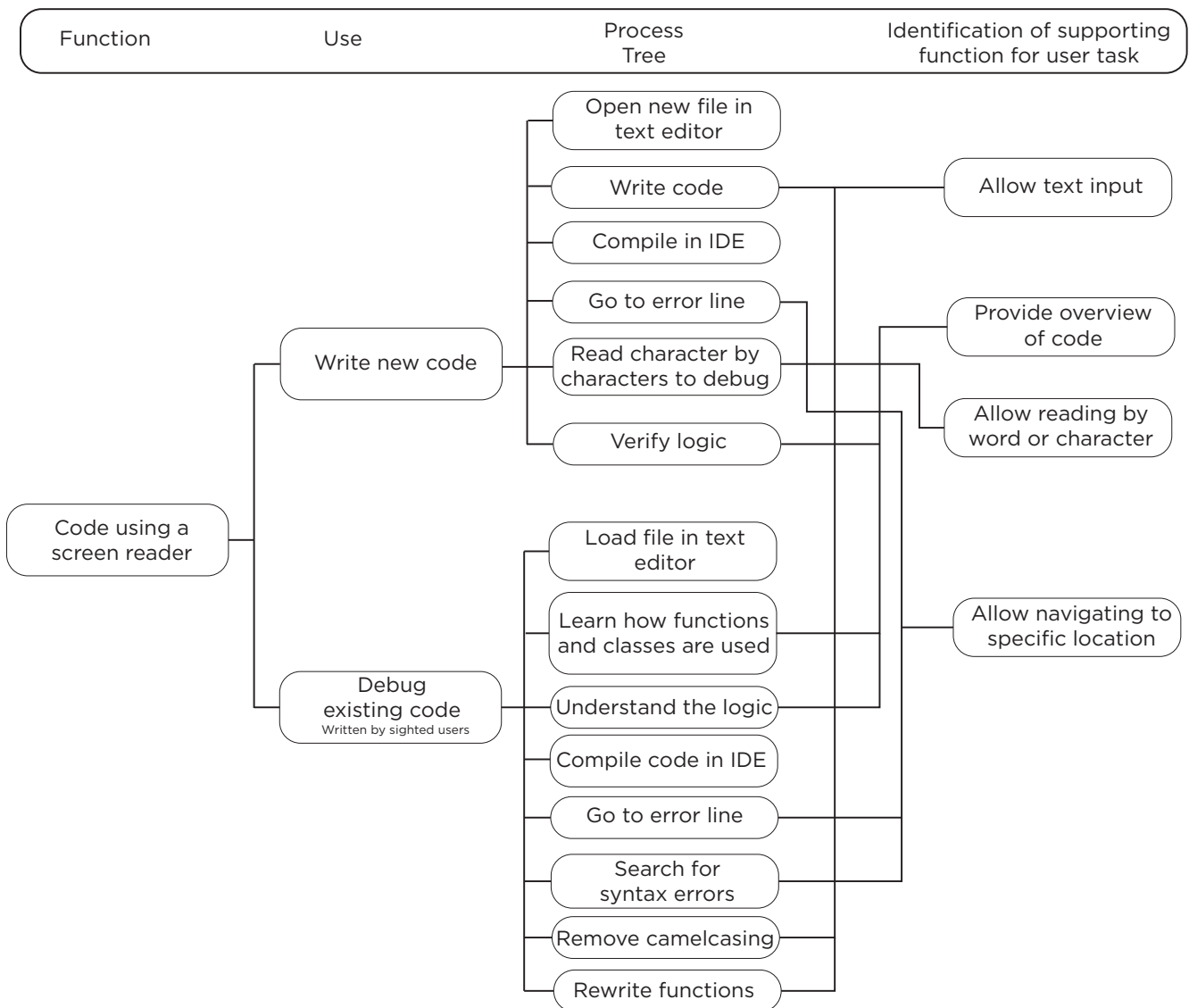


Figure 4 Functional Analysis of screen reader usage by blind professionals to code

3.2 Creative Session

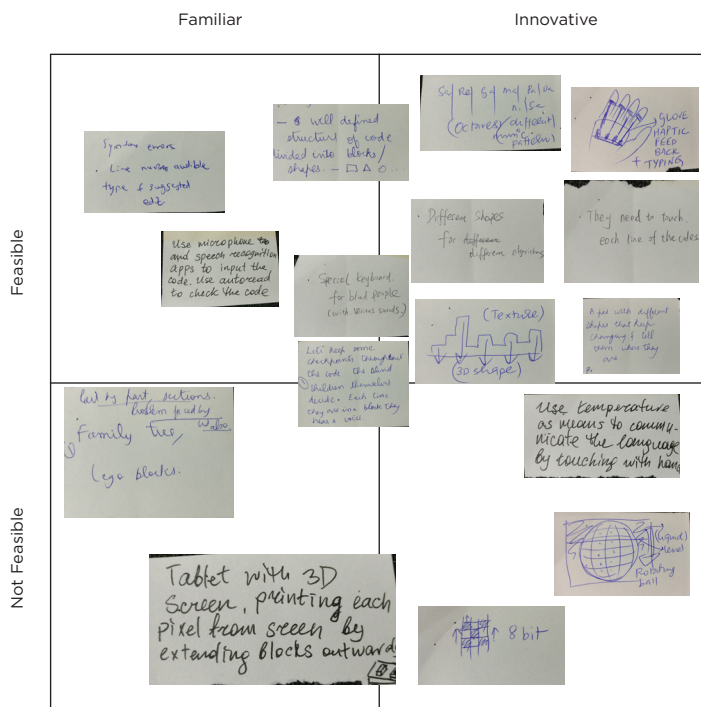


Figure 5 Ideas organized into a C-box

A creative facilitation (Tassoul, 2005) session aimed at generating a large number of divergent ideas was undertaken. Design factors identified from the functional analysis (Appendix 3.1) (Roozenburg, 1995) of existing screen reader use in textual programming languages; these functions were taken as the starting point for the creative session. Analogies, metaphors, and random stimuli were used.

Six TU Delft students from different study backgrounds volunteered towards the 90 minute session (see figure 6) (Appendix 3.4). The session generated 30 ideas out of which 17 ideas were dot voted and analyzed using the C-box method (Tassoul, 2006)(See figure 5).

Figure 6 Creative session at TU Delft



3.3 Morphological Chart

The ideas generated from the creative session and the functions defined earlier were used to form a morphological chart. This helped to combine different ideas and components to serve each of the defined functions (see figure 7). The components were arranged in the X-axis while the functions were placed in the Y-axis.

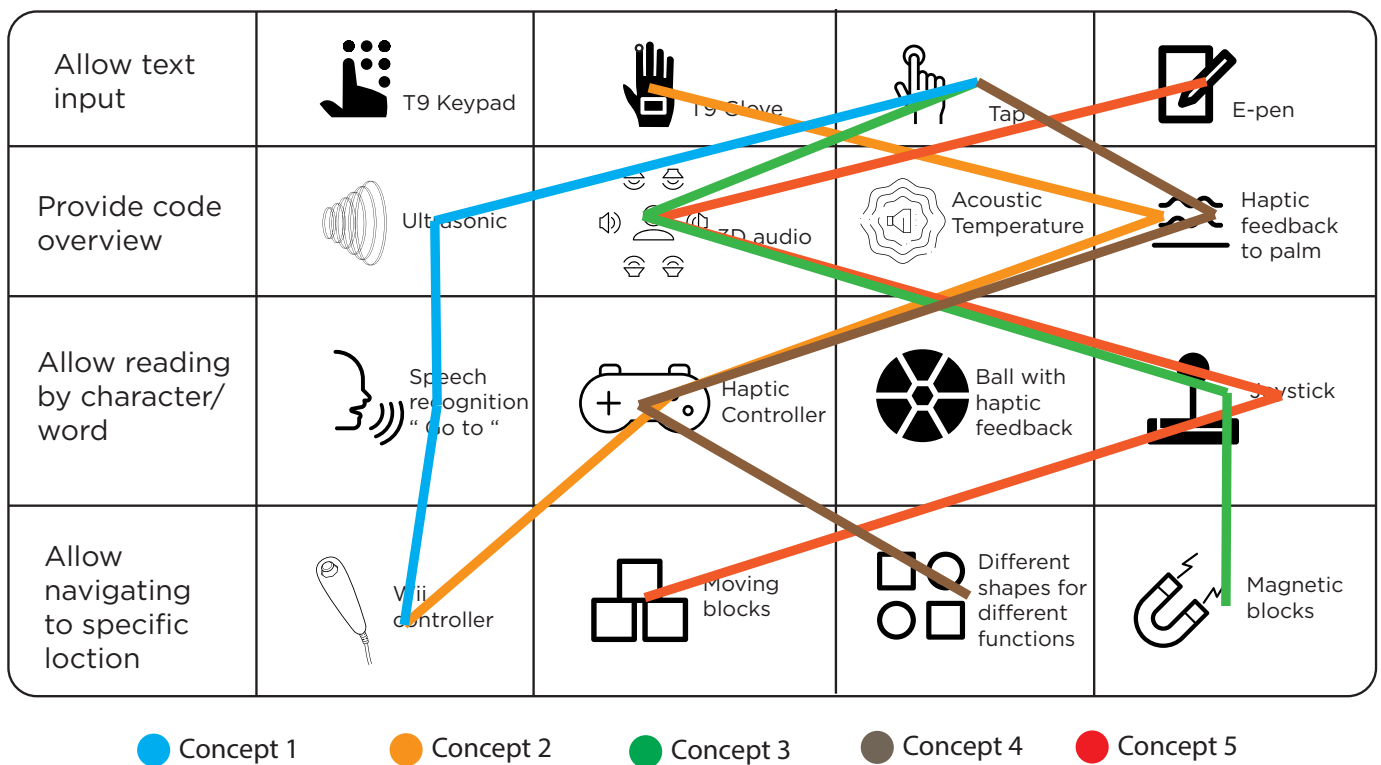


Figure 7 Clustering of components for conceptualization

3.4 Concept Development

The combined ideas from the morphological chart were further detailed, and concepts were defined. Early prototypes were made to check their feasibility and usability problems (Appendix 3.3). The focus was on developing concepts for multiple sensory representations of code structures, similar to how they are displayed for a sighted user. Five concepts were defined, prototyped, and evaluated.

Concept 1

Clip-on ultrasonic sensors to detect the relative position of the user's fingers pointing on the screen (see Figure 8), inspired by Finger Reader technology. Then the device gives audio feedback on the level of the indentation of that particular line. This would work without requiring any firmware/ Human interface Device (HID) drivers on the host computer.

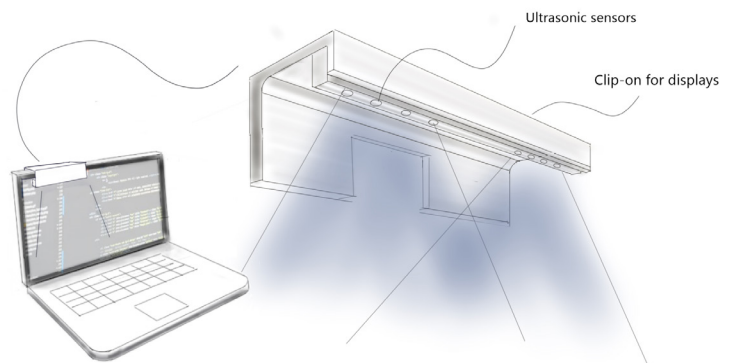


Figure 8 Ultrasonic sensors to track finger positions

Concept 2

A handheld device strapped (see figure 9) onto the wrists, with input and feedback systems. The input is based on the T9 dictionary (predictive) text. Each of the eight fingertips would have flex sensors that return an alphabet based on frequency. Haptic feedback regarding code structures is provided in the form of vibrations onto the palm, while gestures like swaying hands from left to right, would read out aloud (voice synthesizer) the text for user feedback or navigate between the words.

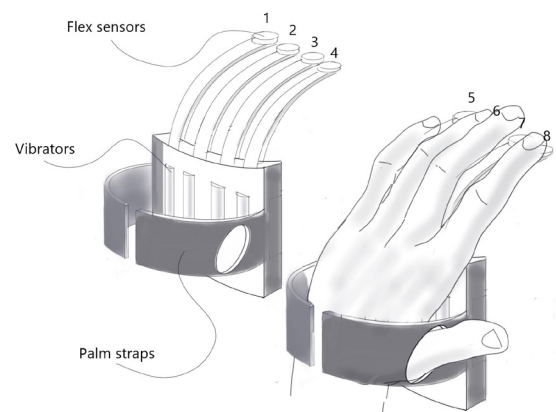


Figure 9 A pair of hand-strapped devices for text input, haptic and audio feedback

Concept 3

Inspired by magnetophoretic displays commonly found in children's magnetic drawing toys, concept three is a grid of electromagnets, and buttons on each of them, to align iron filings into bumps on the surface (see figure 10). A microcontroller reads code structures on the computer through firmware and magnetizes and demagnetizes these magnets. These iron filing bumps would provide information on code structures such as indentation. On pressing a bump, the user would listen to a synthesized voice of that code line.

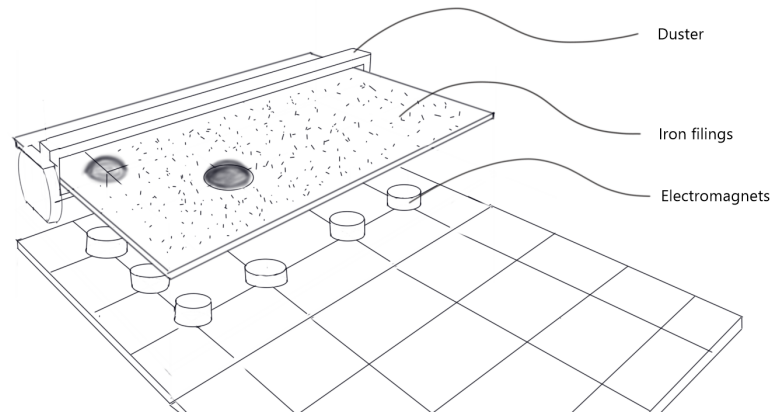


Figure 10 Concept 3: Electromagnetic grid to form tactile bumps on the surface

Concept 4

Rows of tactile tiles (see figure 11) that move sideways along a conveyor like belts. The position of the tiles would be controlled by servo motors and a microcontroller. A firmware would translate indentation levels from a text editor on the computer into their positions. A group of belt and rollers, with varying types of tactile texture materials, stuck on the belt. Each texture means a class or function. In this way, users can feel the logic of the code using real-time positions of the tactile tiles.

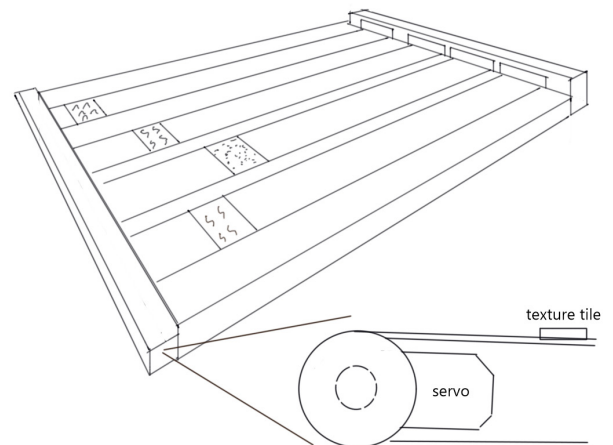


Figure 11 Different tactile surfaces on their respective conveyor belts, that represent code hierarchies

Concept 5

Building upon concept 4, this concept (see figure 12) is inspired by abacus and MIDI sound mixers. It would feature button sliders that move as per the code hierarchies. Similar to the previous concept, a firmware would interface the device with the computer. Musical strings would be used to differentiate between different indentation levels, such that when a blind user would move his fingers across these strings, it would give different notes across different strings. These strings were inspired by guitar frets and strings.

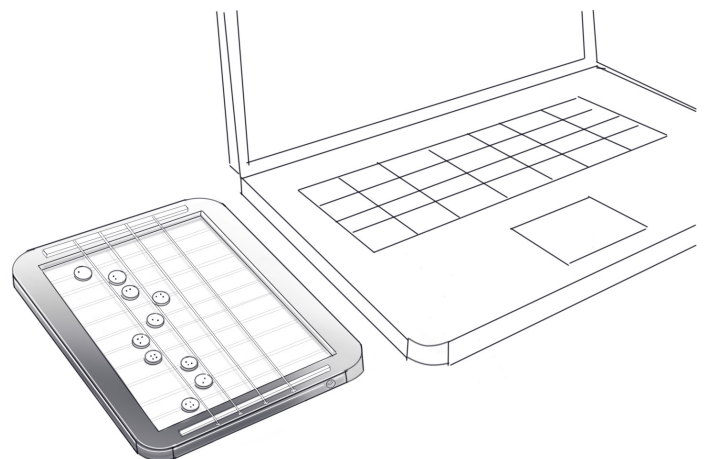


Figure 12 Rows of buttons sliders that would move according to code hierarchies

Conclusion

Concept 5 was selected as the final concept based on the weighted evaluation (Appendix 3.4). The selected concept scored higher in the ability to skim and navigate, technological feasibility, and a possibility for a failure-proof design. The participants who were blindfolded and asked to read a sample python code also expressed their comfort and sense of playfulness while interacting with the mockup, as it was familiar to them to use sliding motion to understand levels (e.g., abacus, music mixers, etc.). The next chapters explain how this concept was further developed and evaluated with users in simulated tests.

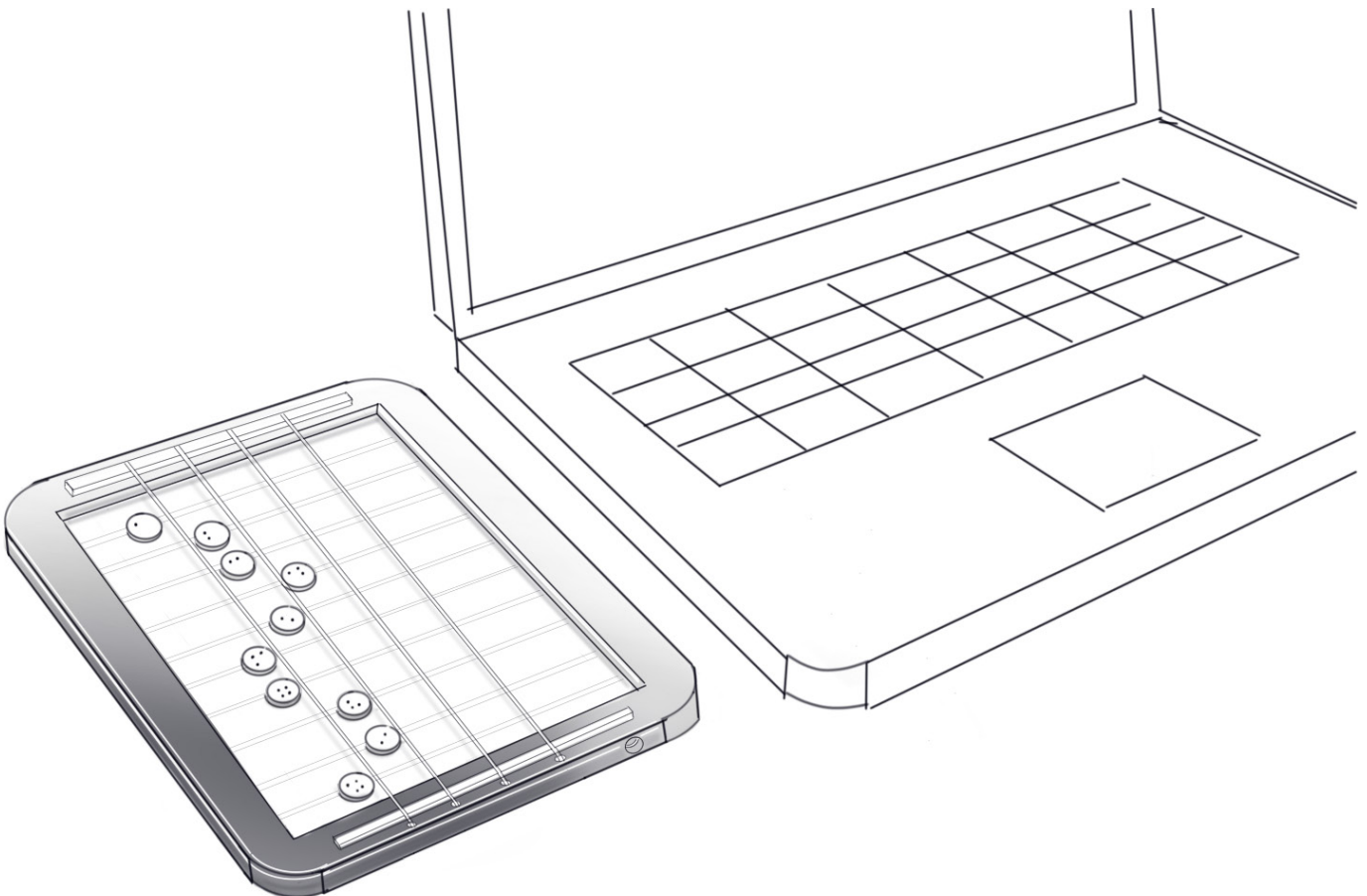


Figure 13 Final Concept

4. Detailing

- 4.1 Final design: Quick preview
 - 4.2 Aesthetics Exploration
 - 4.2 Product Interaction
 - 4.3 Embodiment
 - 4.4 Control System
 - 4.5 System Architecture
-

4. Detailing

As seen from the earlier chapter, concept five was chosen to be developed further, as it scored higher (than other concepts) in addressing the skimming and navigation challenges. The concept detailing phase resulted in a fully functional prototype to meet the technology readiness level 6, as intended in the synthesis phase. The goal of this phase was to enable the device to perform skimming, code navigation, and code folding functionalities.

Initially, an extensive effort was invested in developing a functional algorithm that can enable the device to read any given code (across all platforms and languages), for a more realistic data from the validation phase. But due to time constraints, the efforts were stopped, and the work was moved to future scope. The detailing process started with an aesthetic exploration, followed by embodiment design, and then technical aspects were developed.

The final design featured a tabletop device that has an in-built processor, to read code and represent it in the form of moving buttons in different rows. When clicked on one of the buttons, the device reads out aloud (or through headphones) the contents of the line. It can be thought of as a screen reader that gives the freedom to the user, what he/she wants to read by organizing the information feedback and splitting it between different sensory modalities (Tactile and Audio). The design was made to help users with problem-solving, to feel like an extension as one's own cognitive self (Hutchins et al., 2000) instead of the user feeling the effort of using a tool to accomplish the task. The working surface was defined and etched with tactile marking to differentiate different hierarchy levels.

A navigation joystick at the bottom of the interface helps the user to move across characters or words (x-axis) or across pages (y-axis). The joystick motion was Skeumorphically designed so that users follow familiar mental models of scrolling up and down (for different pages like in a smartphone) and reading left to right. The interface as a whole was designed to be clutter-free, and only essential information was conveyed, and also enough to perform programming operations.

4.1 Detailed Design: A quick preview



Figure 14 Final detailed concept



Figure 15 Navigation



Figure 16 Skimming and Comprehension

4.2 Aesthetic Exploration

Aesthetically pleasing products are not only desired by sighted people. It is also important for a blind user. An interview conducted by Shinohara and Wobbrock (2011) reveals how they feel, “As a blind person, yeah. Maybe I do not see it, but other people see it, and I want it to be, you know, as glamorous as the next guy”. Since the product is intended for students starting from adolescence and up, at this age, teenagers want to feel trendy among their peer group.

In addition to the emotion it evokes, a vision for aesthetics was formulated: ‘A device that portrays reliability and trust’. As one of the interview participants revealed, “I do not get to use my expensive braille device, as it feels fragile and would break anytime soon. The The Nine Moments of Product Experience (9MoPE) method (see figure 17) was

used to map the micro, macro, and meta aspects of the human-product interaction, based on which different forms were 3D modeled. The aesthetic perception was then evaluated, which will be explained in chapter 4.2, and based on the insights gained about object handling by blind users, the final aesthetic design was decided. The key aspects to be considered were:

1. A tactile reference point to help users orient the product properly
2. Clutter-free with fewer buttons as possible
3. Clear demarcation of hierarchy levels on the surface

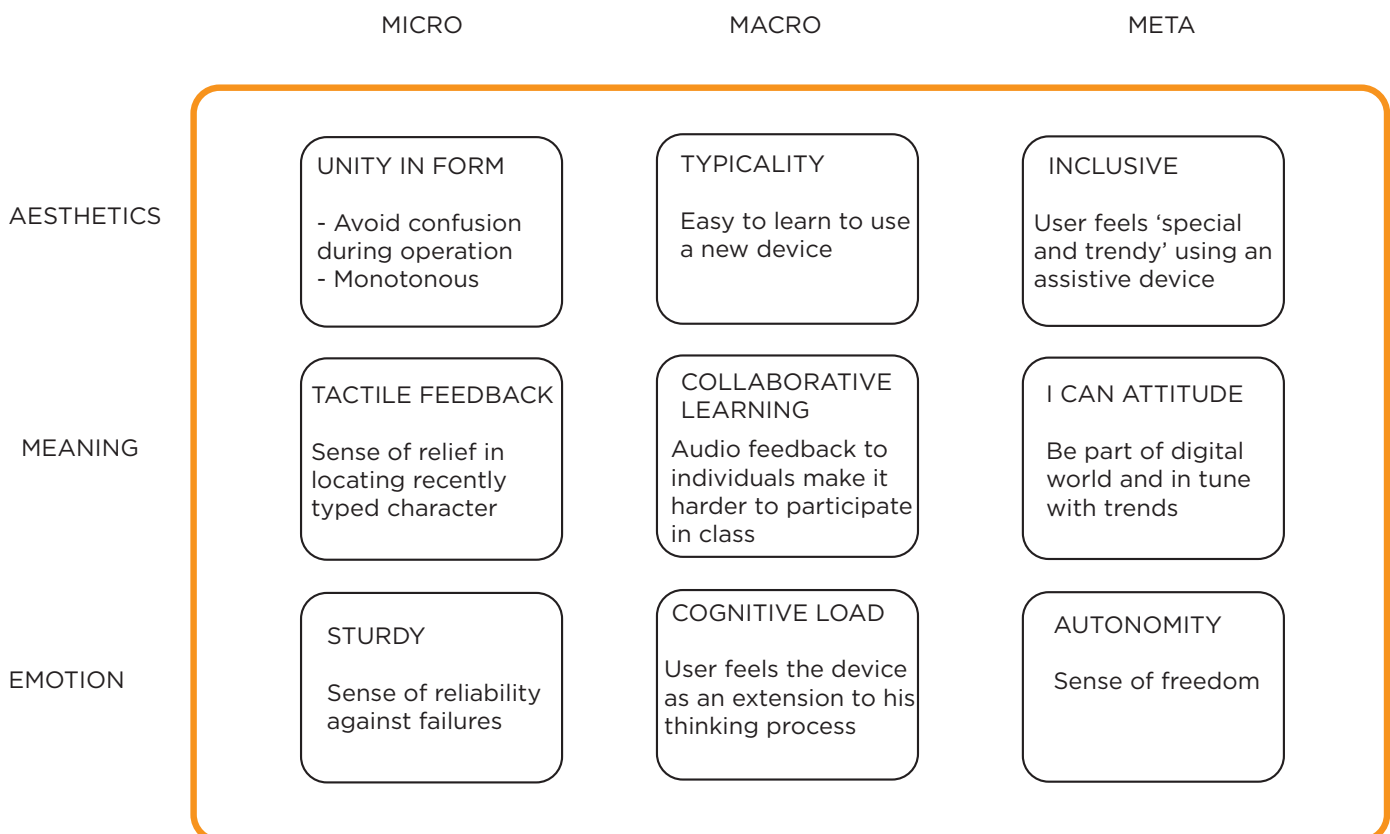


Figure 17 Product experience vision matrix

Visual mood boards (see figure 18) (McDonagh, 2001) were developed to provide visual reference of what the product should look and feel like. Inspiration was taken from accessible products as well as other gadgets that gave a safe and reliable emotion. The images compiled using Adobe Spark.

Figure 18 Moodboard for visual reference



4.3. Embodiment

The physical detailing started with embodiment design. It was started first because, to build prototypes and iterate it, a physical unit was required. It consisted of two components- Housing and chassis. The housing was designed to suit the final aesthetic form, while still being able to support the different components within and allow the intended functionality. The chassis held all mechanical drive components together. The chassis was first built in the process, as it helped to test the electronics and programs to run it. To build a quick prototype of the chassis, medium density fiber (MDF) boards were used, using laser cutting. The first version supported only one stepper motor. This served as a validation and optimization tool towards the development of the mechanical drive (more in chapter 3.4).



Figure One motor version to optimize screw design

The second version, was capable of supporting five stepper motors.

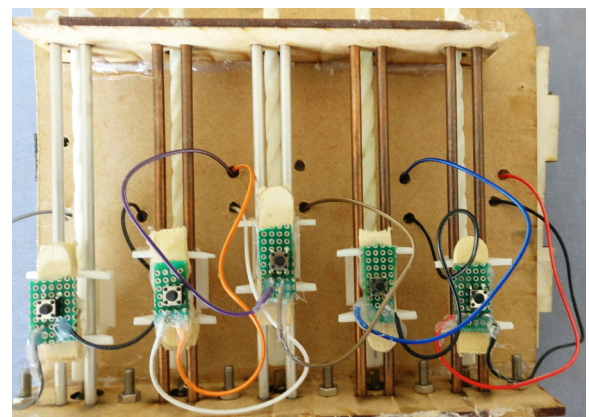


Figure Chassis with five motors

Embodiment Challenges

First, the known and speculative challenges, in embodiment, were realized. This helped to better plan the development process.

1. Need to ensure a minimum linear velocity of 25mm/s for the tactile elements, for low latency between the elements and refreshing pages or lines in the computer

2. Ensure a way to convey row and column numbers without using numerical braille

3. Need to make the chassis support all components and at the same time allow easy replacement of components during prototyping.

4. Need to find a way to effectively interface hardware and code development software.

5. Need to make the housing as compact as possible, though it was not a top priority within the project scope. However, it was considered, so as to provide an experience to validation participants as close to reality as possible

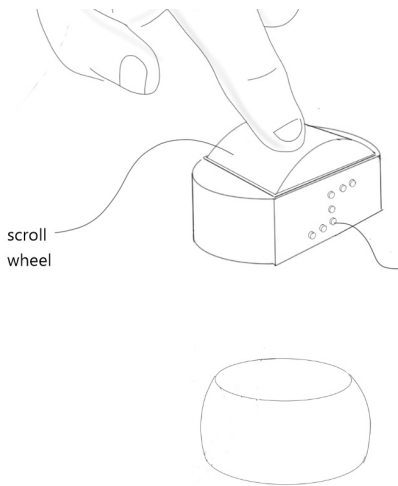


Figure 19 Scroll button concept, to navigate through words

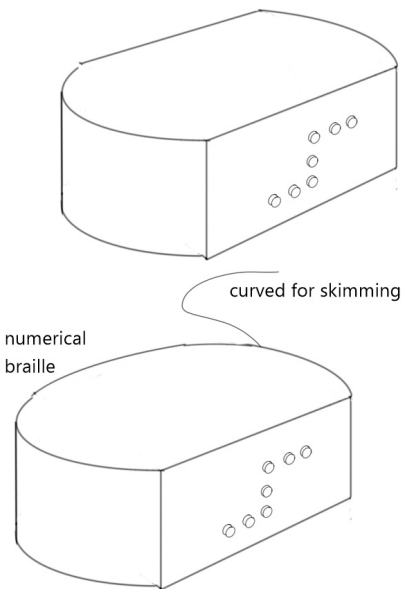


Figure 20 Curved edges to facilitate skimming

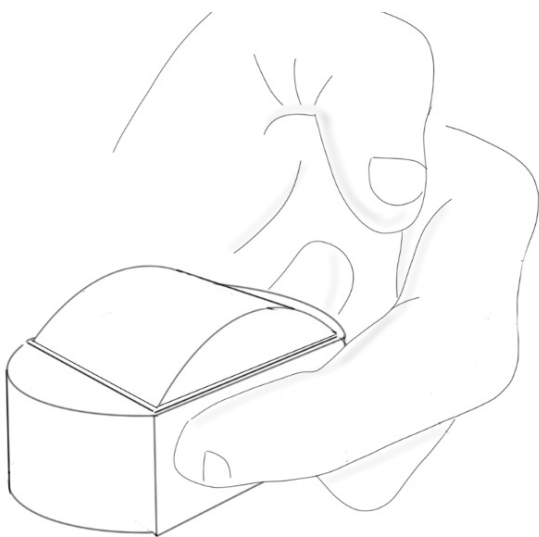


Figure 21 Using numerical braille on the side to indicate row numbers

It was observed that the MDF joints, when slightly deflected, when the steppers moved out of control and damaged these joints, the mechanical drive got stuck and overheating the motors. Hence, a 3D printed chassis was fabricated. It was sturdier than the MDF version, and the joints were glued with epoxy adhesive. But such an arrangement was not allowing changes to modify the chassis in any way for iterations.



Figure 3 3D printed chassis for mechanical drive

The housing was developed in parallel, and provisions were given in the chassis so that it can align itself within the housing.

Initially, the chassis was also built using MDF material, for Wizard of Oz type of prototyping, that included popsicle sticks which move buttons attached to it, when pushing or pulling these sticks. The interaction surface had rubber bands stretched and suspended at some height kept in different levels of tension, to make different sound tones. The idea was to mimic guitar frets, and while a user moves across these rubber bands, they can quickly recognize at which indentation he or she is in.

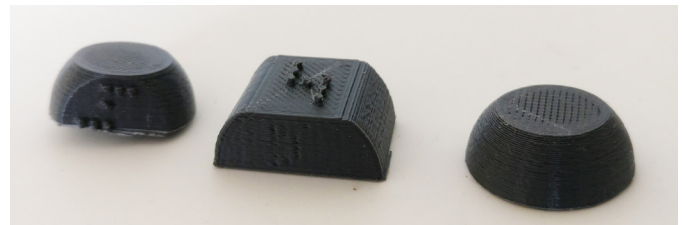


Figure 4 Button prototypes

Button concepts (see figure 19, 20) were also explored. It was difficult in the beginning to indicate to the user about

the row numbers. Column numbers were identified with the stretched rubber bands as explained above. But it was difficult to denote row numbers without using numerical braille.

Different ideas were explored to convey row numbers, including dots on the top or sides, which does not necessarily mean braille but to denote row numbers corresponding to the number of dots on the button. It was then decided just to use double tap interaction using electronics to say out aloud what row number it is.

A scroll button concept was explored to enable code folding functionality. It had a mouse encoder within a PLA casing, which would also slide along in the mechanical drive.

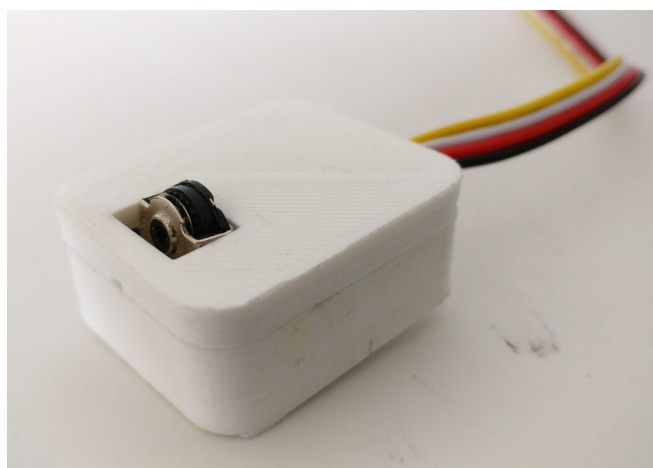


Figure PLA enclosure for mouse encoder

The scroll was assumed to be useful, but in reality, was hard to scroll and since it was a narrow width wheel with a metallic frame, it also hurt the fingers while scrolling for long. Then it was decided just to use push buttons and manage to make the code folding feature possible with the control system design.

Top Panel

The top panel was designed to be the working space with clear demarcation of edges and also had tactile markers to indicate the indentation level. These

markers mimicked the surface tiles in blind schools (Skeumorphic) as per the requirement R4.2. Slot width was also made optimum as per R1.2.

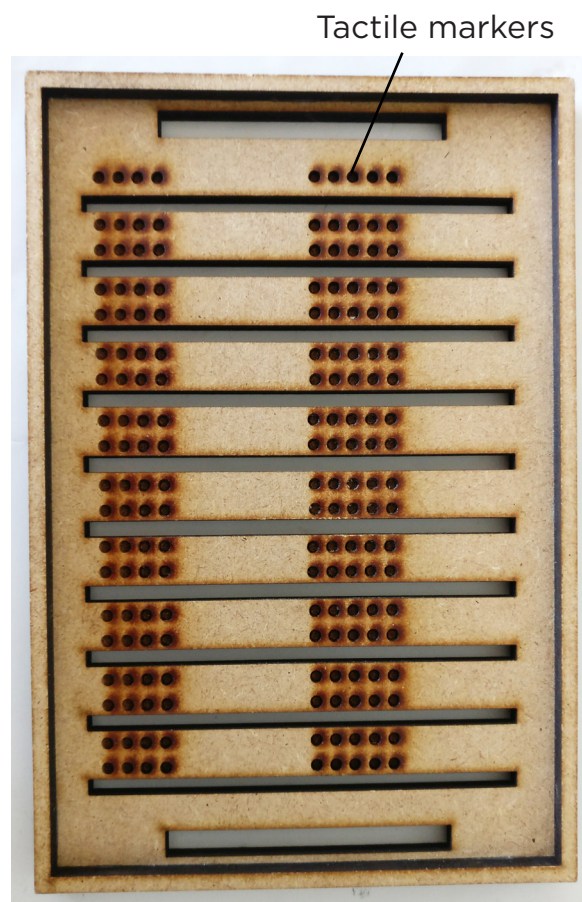


Figure Top panel

Another variant of the housing was developed as a means to fulfill focus area F3: Ensuring affordable products with more reach to end-users. Since it was initially planned to perform user tests remotely with a blind school in India, this variant was developed. Although later the remote testing plan was dropped due to time constraints, the DIY (Do-It-Yourself) variant was useful for quick iterations, which will be explained on the next page.

The goal of embodiment design was to provide a clutter free interface for the user. The interface design (see figure-interactions) decisions followed the Principles of Interaction Design (see Appendix 4.8 and R4).

Do-It-Yourself Variant

The DIY variant was built entirely of MDF material (including chassis). Laser etching was considered as a means of differentiating surface textures and communicate indentation levels.

To help the primary and secondary customers (see chapter 5.2) to assemble product on their own like IKEA model products, for higher product scalability and lower packaging costs, a Do-it-Yourself (DIY) approach was used. Following the principles of Design for

Assembly and Manufacturing (DFMA), the parts were designed to be in layers (see figure 13), which can be easily aligned using puzzle-like joints, so that users cannot misalign them (see figure 21).

The layered approach was also useful in terms of modularity, which allowed continuous iteration without having the change the entire design. This served the requirement R2.3, and when something failed it was easy to change the part design

As per Requirement R4.1, the top left corner was designed to have a 45mm fillet (see figure 23, 24). The fillet was achieved using a lattice joint in Poplar wood.

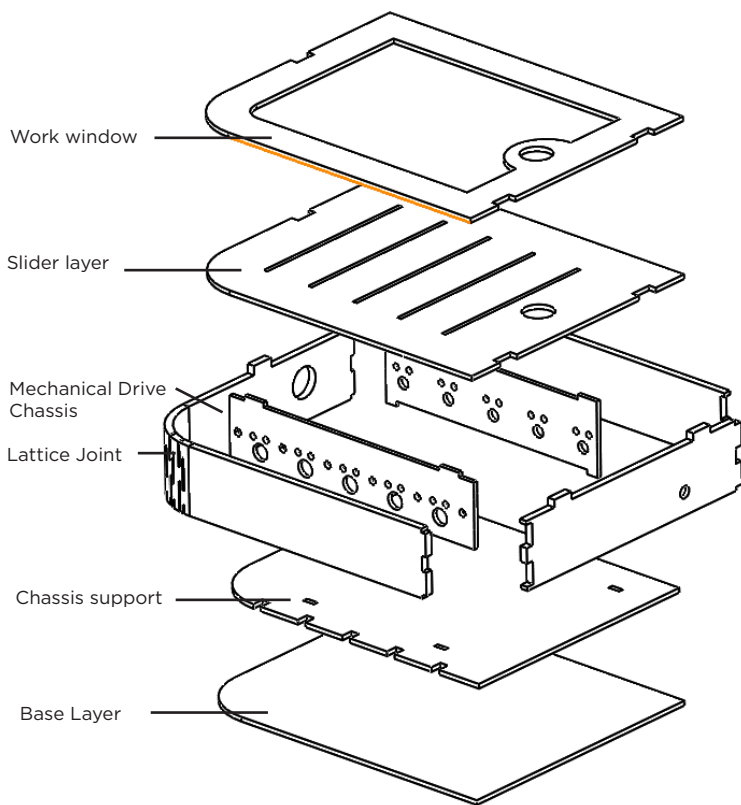


Figure 21 Exploded View of DIY variant



33 Figure 22 Assembled DIYvariant (without buttons on the sliders)

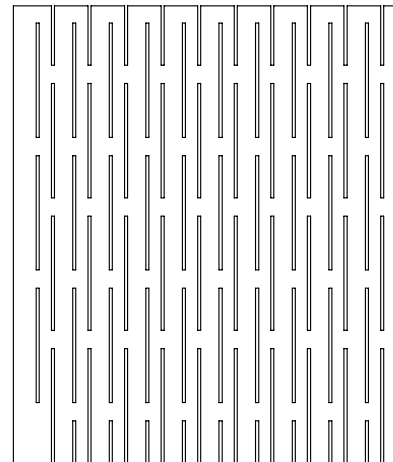


Figure 24 45mm radius lattice joint patter

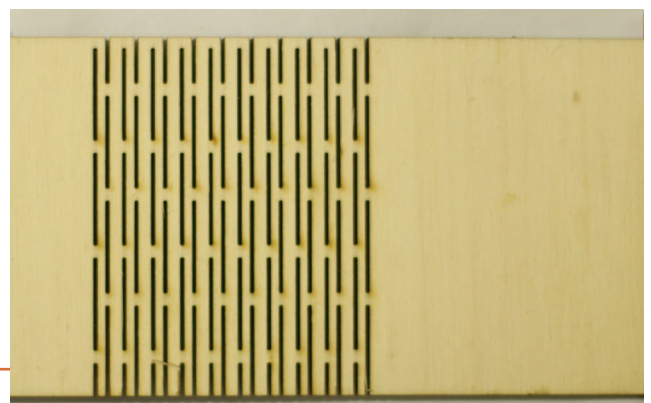
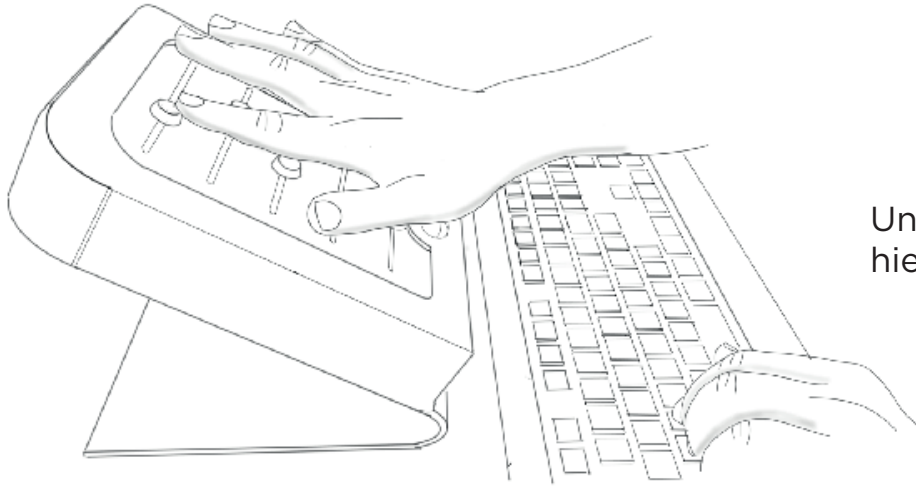
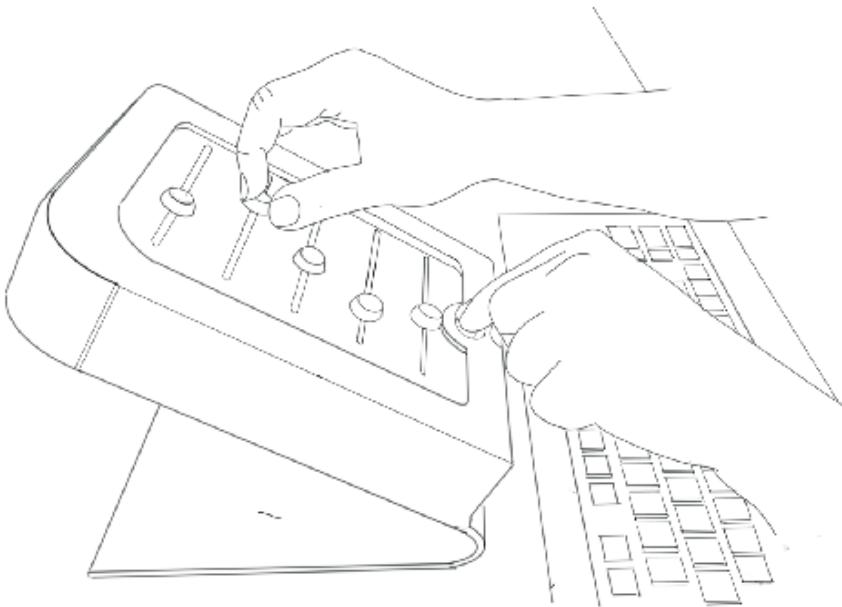


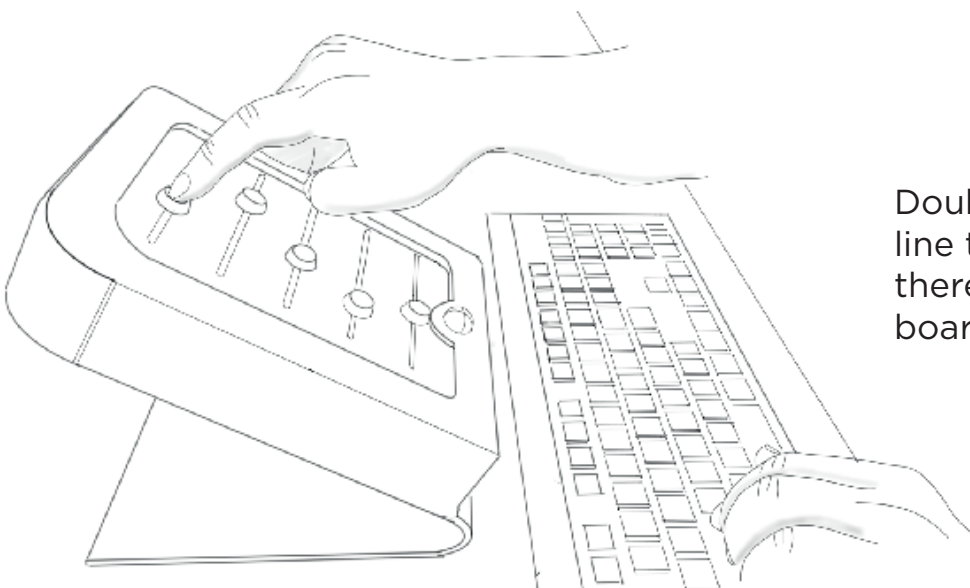
Figure 23 Lattice Joint in poplar wood



Understand code hierarchy and logic



Quickly skim through the lines



Double click a specific line to move cursor there and use keyboard input

Figure 24 Interaction features

4.4 Control System:

The control system was developed to precisely position the buttons sliders and represent code structures tactually, with the least lag possible between digital information and mechanical motion. The control system consists of two main elements- Mechanical drive and electronic controls.

Mechanical Drive

A pinboard (see figure 26) was an inspiration to create a tactile map for code structures. However, it was declined as actuating each pin would be not a cost-effective idea.

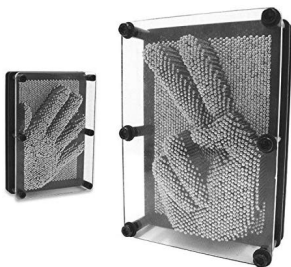


Figure 26 Pinboard

The leadscrew based slider mechanism was chosen, as the main requirement from the drive mechanism was that the tactile representation should be as instantly as possible whenever the lines or pages on the computer refreshes. This meant the displacement of the moving part should

- 1) Have large displacement per unit time,
- 2) Have fewer slips,
- 3) More positional control.

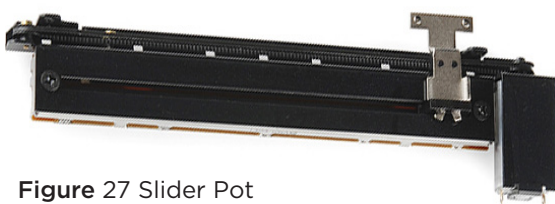


Figure 27 Slider Pot

In the early stages of this development, off the shelf drives such as motorized pot sliders (see figure) were considered. These were used in MIDI sound mixers, and because of this standard application, these spares were expensive for this product case. So linear actuators based on lead screws were fabricated. The number of positions of tabs was considered to

be 4, with each indentation level spaced at 25mm apart, so that there is enough differentiation between each level.

Lead Screw Design

Different materials were used to experiment pitch and displacement time, between each indentation level. M4 steel screws with milled aluminum coupling were used while the traveling nut was embedded in a laser-cut MDF housing.

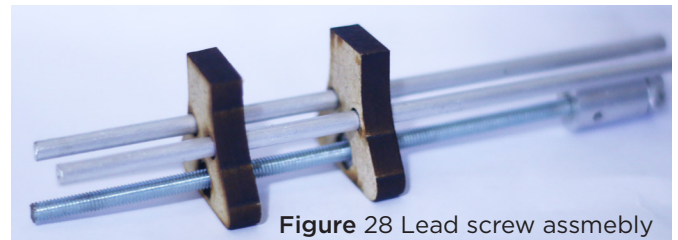


Figure 28 Lead screw assembly

But since the standard M4 nut had a single start pitch of 0.7mm, it took 35 seconds to traverse a distance of 25mm (across a single indentation level). The stepper motor used here was 28BYJ-28, which had 15 RPM. The speeds at different pitches and the number of starts were calculated (table). The slow-motion meant significant latency between change in digital information, and it's reflection in the position of the slider buttons. To increase the rate of displacement of travelling nut per revolution, a lead screw with a large pitch was needed.

Table 1 List of lead screws with time taken to cover 25mm

| Part | Screw diameter (mm) | Thread pitch (mm) | Stepper RPM | Time Taken (s) |
|----------|---------------------|-------------------|-------------|----------------|
| Standard | 4 | 0.7 | 15 | 31 |
| Standard | 8 | 1.25 | 15 | 18 |
| Standard | 12 | 1.75 | 15 | 14 |
| Custom | 4 | 25 (1 start) | 15 | 8 |
| Custom | 4 | 25 (2 start) | 15 | 4 |

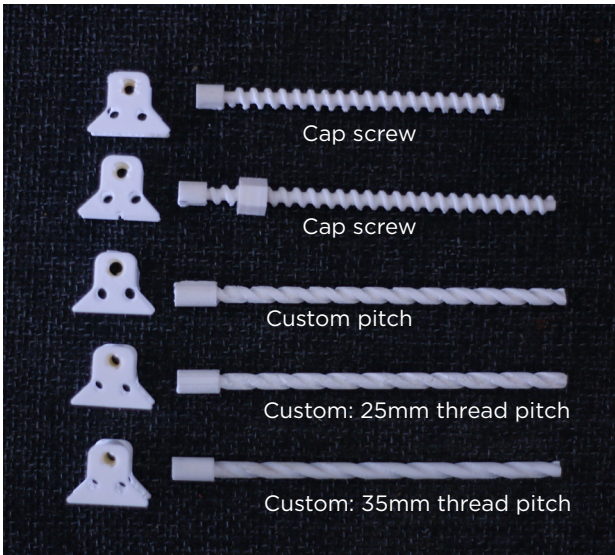


Figure 29 Evolution of lead screw and their corresponding travelling nut design

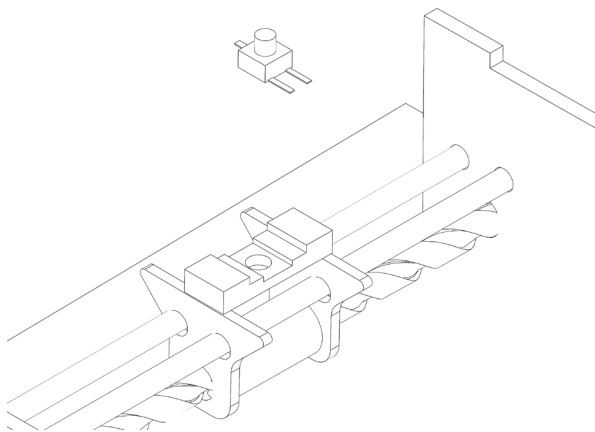


Figure 30 Push button sub-assembly

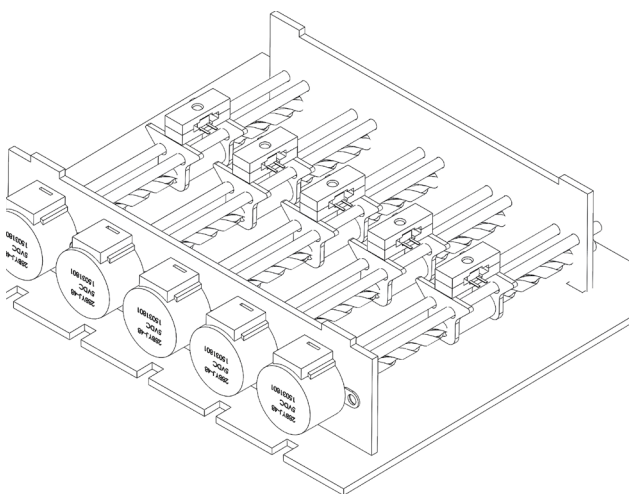


Figure 31 All the lead screws assembled in the internal chassis

| | | | | |
|--------|---|--------------|----|---|
| Custom | 4 | 35 (1 start) | 15 | 5 |
| Custom | 4 | 35 (2 start) | 15 | 2 |

crews up to M12 had a pitch standard of up to 1,75 mm, meaning it would still take 14 seconds. Also, screws with larger diameter meant more weight, and this would require higher powered and expensive stepper motors.

Since metal lead screws tend to become weaker when creating a pitch of more than 0.8mm on an M4 or M6 screw, 3D printing was considered to fabricate the lead screws. For the ease of material availability, PLA was chosen as the material, while fabricating the screws in metal required tooling, which was again expensive. However, it is recommended to use ABS or POM (for its self-lubricating properties). Since the 3D printing at the Industrial Design Faculty (TU Delft) did not allow POM/ABS printing, PLA was considered for prototyping.

A standard one start and two start bottle thread screws (see figure 29) were fabricated. Since it still had lower displacement rates, the custom design of the lead screw was made. After several iterations (see figure), two-start lead screws and corresponding traveling nuts were fabricated with a 35 mm pitch, using PLA plastic. It was found to be the most optimum for the indentation levels. The motor couplers were designed to be in-built within the plastic screw part. Two starts made it possible for a faster displacement of the traveling nut, for a given pitch. It was post-processed with PTFE lubricant and housed in a laser-cut chassis. They were assembled into the internal chassis and coupled to steppers (see figure 31).

Push-button sub-assemblies were placed over the traveling nut, and each of these sub-assemblies was given a provision to couple the buttons (external) to it.

4.5 System Architecture

The most important part of the control system and even the whole detailing chapter is the electronics and its interfacing with the computer software. Extensive efforts were taken to ensure the least latency as possible between the hardware (sliders) and the software (like text editors or IDEs). Real-time interfacing was developed during the course of the project, but could not be completed entirely within the graduation time. However, it is being developed further by the Open source community, since the project has been open-sourced (see chapter 5.3). Some of the code developed can be found in appendix 4.6. Despite the fact that this part was beyond the scope of the project, it was still attempted at because 1). There was a need to make a

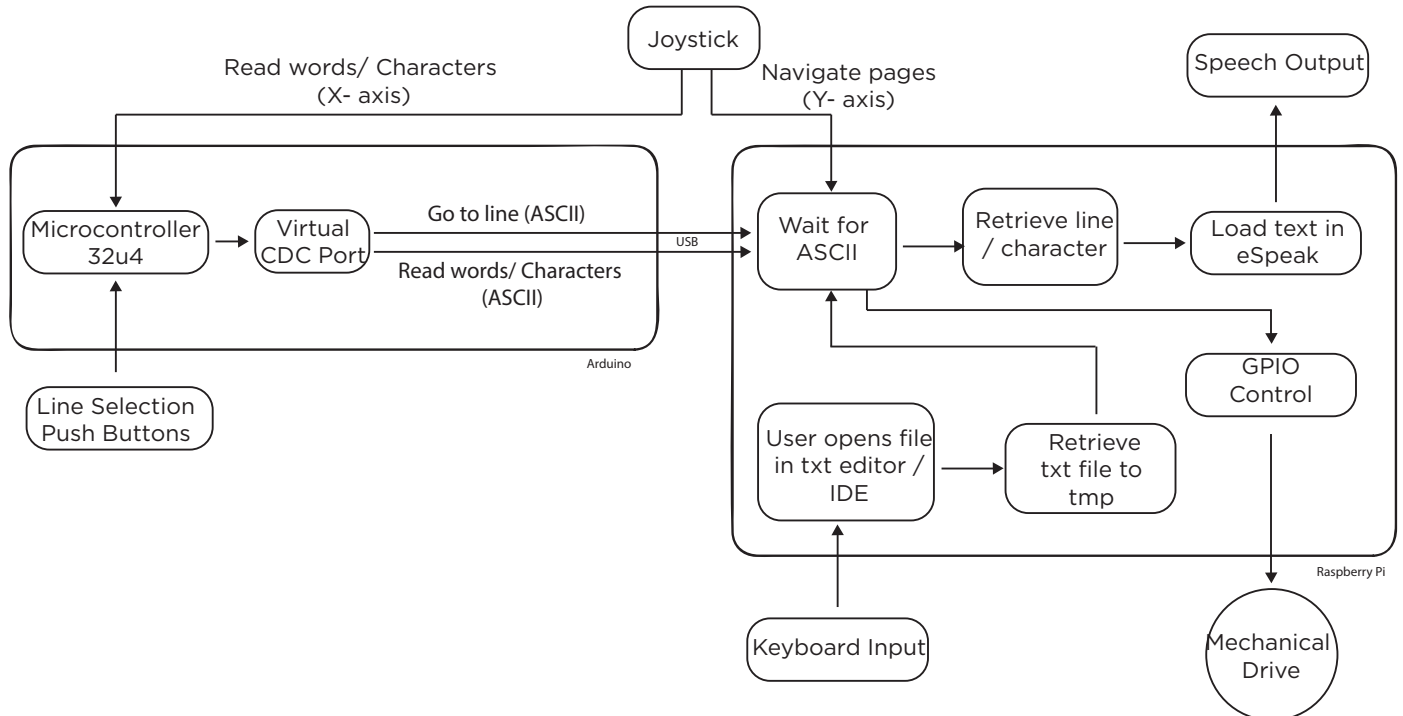


Figure 32 System Architecture showing the fore-end and backend processes

real-time functional prototype so as to validate the interactions and make it feel as close to the real product for getting realistic usability test results, 2) It was considered that by making it to this level of functionality, the feasibility of the approach would be validated.

Main challenges

1. Many text editors and IDEs are not accessible with voice synthesizers and screen readers
2. Screen readers are also dependant on the operating system (OS).
3. So the question was how to create a system that is as independent of platforms (OS and software tools like editors and IDEs).
4. There was a need for a simpler approach to integrate software or the operating systems with the hardware (stepper motors) without having to write specialized driver software. Another disadvantage of drivers could bring more incompatibility bugs.

Electronics

The electronics of the control system (see figure 20) used a Raspberry pi 3B+ and an Arduino Leonardo. The ULN2003a type of stepper driver was considered for its ease of spare part availability and the maximum torque. It has a holding torque of 34.3mN.m, which is low enough to cause any physical injury to the blind user (as per requirement R1.1) even in the case of product misuse and high enough to drive the lead screw without any slip.

The system consisted of both Arduino (Leonardo) and Raspberry Pi, both of which have their individual computing power. The reason why both were selected was that only a processor with

32u4 architecture (in Leonardo) can act as a programmable USB- HID (Human Interface Device). Hence, the Leonardo is programmed to send ASCII (American Standard Code for Information Exchange) with respect to the user interaction (eg. using joystick to move between pages or navigate to specific character or word) Another advantage in this approach is that the active cursor in the text editor can also be moved to a desired position, from where the user can enter or edit text using a regular keyboard.

Raspberry nor any other microcontrollers have this functionality. On the other hand, it was decided to use a Raspberry pi as the main computation device, instead of tethering to an external device because of the following considerations:

1. Including Raspberry pi as a main computation device allowed the product to be a standalone and not affected by tethering failures (Requirement R2.2)
2. Using python scripts was easier in Raspberry pi to control GPIO (general Input and Output) pins to directly interface with the stepper motors.

PCB Design

To minimize hardware errors during prototyping and user testing, and to help focus more on software integration, a PCB was designed (see figure 58. 59) and fabricated (see figure 22). The PCB also helped to resolve critical issues in terms of system architecture and the components required in the final embodiment of the concept, when the designs were open-sourced, and the project gained traction.

Hardware- software integration

As it can be seen from figure, the points of user interaction like buttons and joystick were interfaced with Leonardo, while

the stepper motors, audio output was interfaced with the raspberry pi. As per the expert interview from Visio (appendix 1.3) students use Keyboards as the primary input device. Hence it was decided to use a keyboard as the input for raspberry pi as well. As mentioned above, Leonardo acted as an HID, and corresponding ASCII values were sent to the raspberry pi.

The product launches a boot script as the device starts in the raspberry pi. When the user loads a text file in an editor or IDE, the script reads the file location and retrieves it to a TMP (temporary) folder. The script accesses this file whenever the user presses the function button. Upon access, the file is read line by line, as per the user input (through ASCII codes sent from the Leonardo), if the file has to be read by words or character (Joystick X-axis) or by lines and page (Joystick Y-axis). The buttons on each slider also indicate Go to Line function, and the Leonardo sends a corresponding ASCII value to the Raspberry Pi. In this way, the user can tactually feel the code and move the cursor to a specific location just by tapping twice.

To make the integration as independent

as possible from platforms, the eSpeak engine was selected as the speech server. It allowed the script to load the words or characters to be loaded into the eSpeak server, which then talks it aloud through synthesized speech. This way, real-time integration of the tactile feedback and the audio output was achieved.

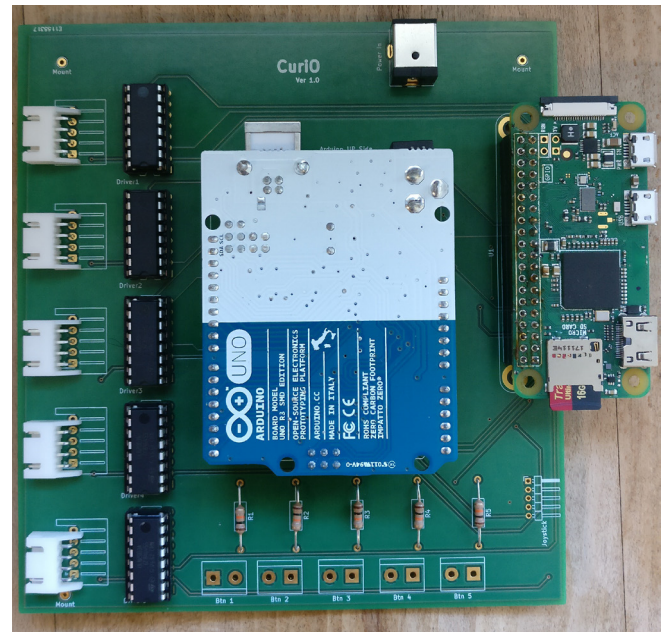


Figure 33 Assembled electronics

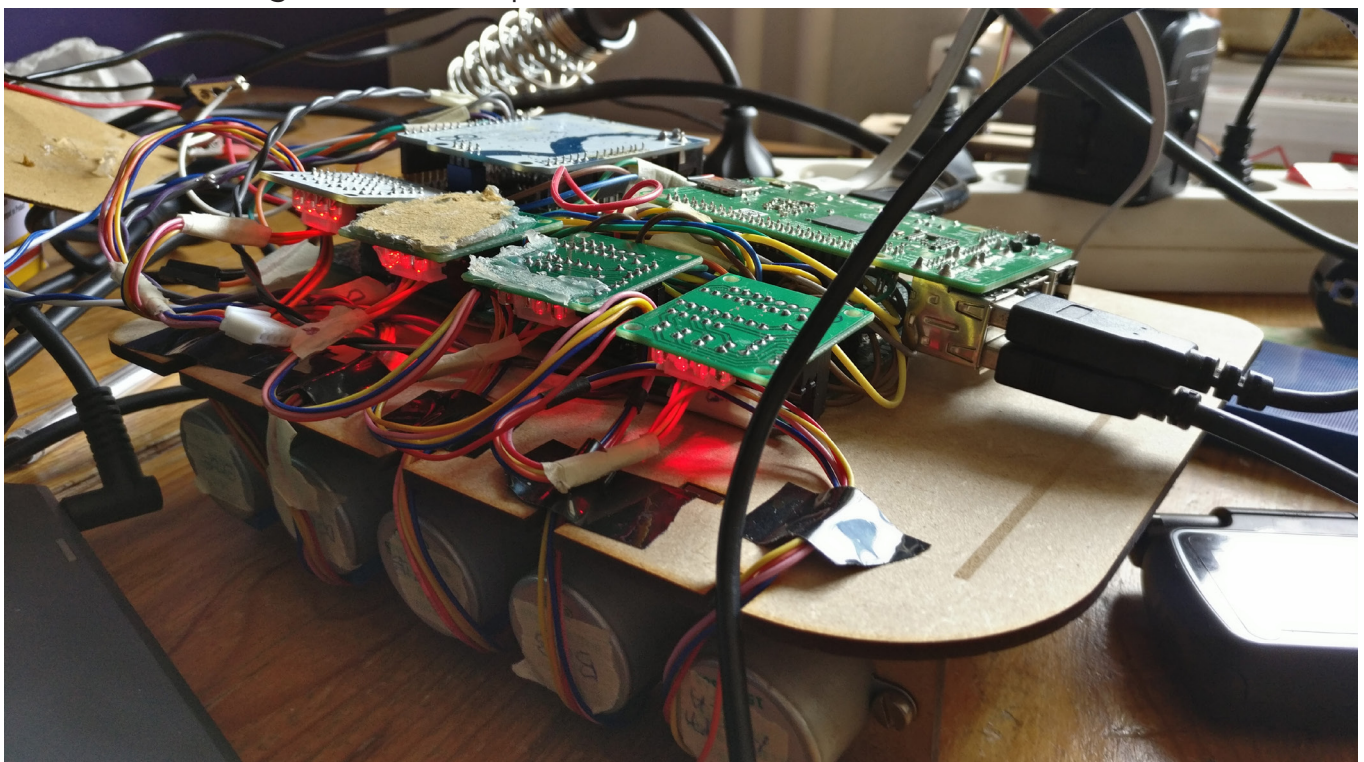


Figure 34 Functional Prototype with electronics

Conclusion

The detailing phase allowed the concept to be developed up to a level of a functional prototype, and allow user tests in a simulated environment. Although all features were not functional by this phase, the intended Technology Readiness Level was achieved. The lessons learned in this chapter was compiled into the recommendation section (chapter 7.1).

5. Validation

5.1 Usability Test

5.2 Aesthetics Perception

Introduction

Validation was essential and useful to find if the design decisions made during the process had been reflected in the final design. The validation was undertaken to measure System Usability and the aesthetic perception and to see if the design has fulfilled the intended requirements and design goal.

5.1 Usability Test

System usability System (SUS) (Brooke, 1986) was used to determine how interaction friendly the detailed concept is and how much it serves the design goal. Six TU Delft students from different study backgrounds and one professional blind programmer volunteered in the test. The blind programmer was a professional at Booking.com, Mr. Parham Doutsdar participated. The students were blindfolded with a sleeping mask.

A form prototype was placed in front of the participant (see figure 23, 24), and they were asked to perceive the logic of two sample python program (Fibonacci Series and Singly Lists), whose indentation levels were preset for the participant before the start of the test, with slider popsicle sticks. Synthesized screen reader voices were pre-recorded and played at every instance the participant interacted with the prototype. A Minimum Viable Testing approach was undertaken (Appendix 6.2).

Feedback:

“It is playful and I like the feel of exploring. It responds to my interactions and I get excited to feel how my code is written”- Shreyas Prakash, Masters student- IO

“It is easy to reach and simple to use “- Vinayak Krishnan, Masters student- 3ME

“Because its a new product than the ones I am used to, maybe I need training to learn all the features”- Parham Doutsdar

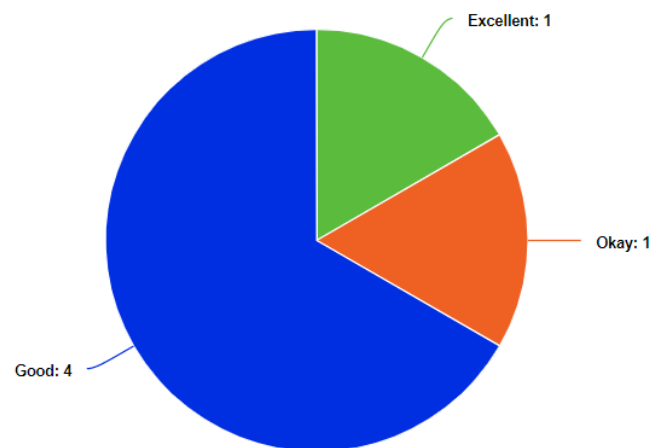


Figure System usability Results



Figure 36 A blindfolded participant interacting with the prototype

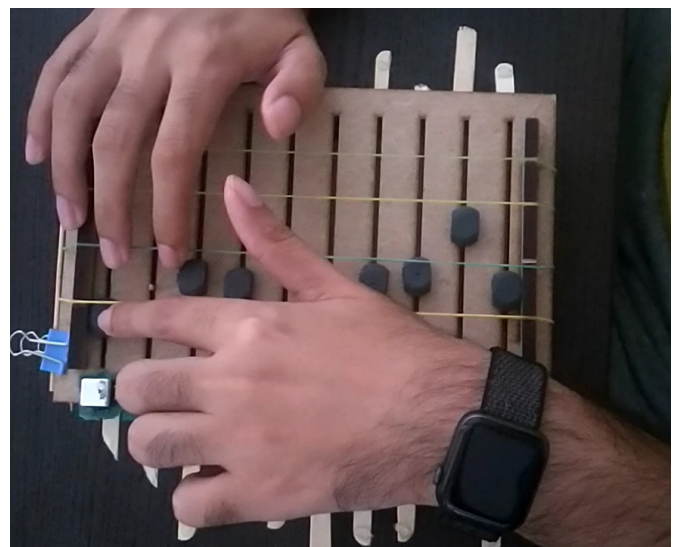


Figure 35 The blind programmer interacting with the prototype

5.2 Aesthetic Perception

A buyer's perception of aesthetics and end user's tactile perception was evaluated to determine if the product experience vision of reliability and trust is reflected in its form. The test participants were students from TU Delft, from different disciplines in the age group of 21- 29 years old and one blind user. It was ensured to have an equal proportion of male and female participants.

They were shown a rendered image (see figure 12) of the product on a computer screen and presented a questionnaire with 5 points Likert scale. The haptic aesthetic (see figure 25) helped gain insights on whether a blind end-user would prefer to buy the product after experiencing its aesthetics and the emotions it conveys.

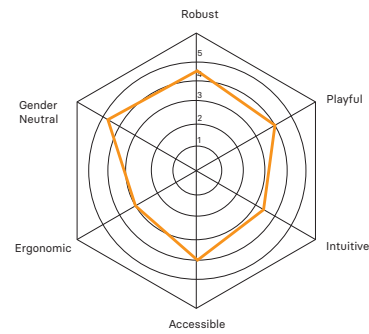


Figure 37 Radar chart showing the average score for each aesthetic criteria

Feedback:

"I like the interaction of it with a lot of buttons and I get invited to play with it " - Parham (Blind programmer, Almere)

"I feel it does not break, even when i drop it. I can imagine this is important for the blind"- Vinayak Krishnan (3ME, TU Delft)



Figure 38 3D printed form prototype presented to a blind programmer, for tactile perception of the product.

Conclusion

Using the methods used in this chapter, it was realized that ergonomics and intuitiveness could be improved in the future. As one of the participants said, “It is playful to use at this angle, but I think it might force me to reach out forward, causing strain in my back.” A possible solution could be to remove the 30-degree inclination and lay the product flat on the table, like in the MDF variant (figure 36). Due to time constraints and the scope of the project, these iterations could not be fulfilled. But it was worth noting that the general system usability was scored high, in terms of ability to skim and navigate through code and understand the logic of it.

6. Business Model

6.1 Market Analysis

6.2 Customer segmentation

6.3 Open source Hardware Model

6.4 Revenue Model

6.5 Cost Structure

6.6 Promotional Strategy

6.7 Financial Plan

Introduction

Also, as an approach to find design solutions to improve the accessibility of programming education, the project also focussed on the ways to make the approach accessible the end-users. This chapter explores the potential of the design solution as a sustainable business proposition. On the contrary to conventional models, the business model was based on the Open Source Hardware approach. The guiding question for most of this chapter is,

How can an Open Source Hardware model be sustainable?

The first question that is asked about the Open Source hardware model is, “ Where does the money come from if the designs are open-sourced?” This chapter answers this critical question. It was realized during a number of conversations with the Delft Open Hardware Community that it was essential to develop a sustainable business model that can make a venture sustainable as any other business, but at the same time promote the spirit of Open Source- Freedom.

6.1. Market Analysis

Market Segmentation

Though the product was initially developed for blind users, on market segmentation studies, it was found that with minor adjustments, the product can also serve those users who have uncorrectable low vision and those who have Dyspraxia. Dyspraxia is a condition in which a person has an affected excellent motor control and unable to use a mouse or a regular keyboard.

Market Size

For the first two years, we are aiming to target blind users from middle and high economic classes from Netherlands and India, for which the Total Addressable Market(TAM) is 64 Mn Eur, and our Serviceable Obtainable Market(SOM) at the rate of 1% market share is 640,000 Eur.

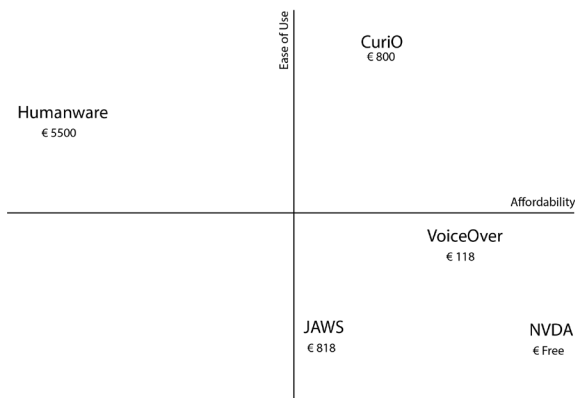


Figure 39 Market Analysis

6.2. Customer Segmentation

Economic Buyer

Families and friends of blind people (retail e-commerce)
Municipalities wanting to fulfill employment opportunities for blind people, adhering to the Dutch Participation (Participatiewet) Act
Employers who can accommodate a blind programmer and benefit from

schemes like Wage Cost-Benefit, LKV, LIV and Employee Insurance contribution.

High schools in the Netherlands who have a dedicated programming track as part of their curriculum. Such schools at the moment are limited. For instance, the Metis Montessori Lyceum is the only high school offering coding lessons as part of their main curriculum

End User

High school children (14 to 17 years)
Professional programmers (workplace),
adult users (18 to 30 years)

Secondary Customer

Makers, Tinkerers, researchers, educators

Early Adopters

Established blind schools were engaged during product development, further testing, and validation. The network was established based on personal connections and from previous startup experience in the education domain. The plan is to engage them for Influencer based marketing while seeing if they will be interested in becoming Early Adopters. Some of them could be Bartemeus Zeist (NL), Royal Visio (NL), Technology Centre for Blind (India), and National Institute for Visually Impaired (India).

6.3. Open Source Hardware model

The secret sauce is in open-sourcing the hardware designs and yet having a sustainable business model. The decision was taken to make it Open Source, in adherence to the Design Goal (ii), for two more main reasons:

- Successful implementation of this project on external dependencies, such as educational research and content development, software integration and hardware improvements, before it could hit the markets

- The niche market and its reach cannot follow conventional promotion strategies. There is a heavy need to follow influencer based and word of mouth outreach.

From a business point of view, Open sourcing the design also enables community-based product development while cutting down costs on R&D. The case of Arduino inspired the business model. Their reference designs are open-sourced and continue to be replicated globally. However, their trademarked brand name 'Arduino' gives them the mark of trust in manufacturing. Also, makers and tinkers cannot compete in terms of cost of production and sales, when it comes to mass manufacturing advantages (such as mold and die costs and PCB printing) and sales through a first-in-market approach to customer base.

The design files were published in the Github repo, and since then, several issues have been fixed. Contribution guidelines, code of conduct, and license documents were also published. Gitkraken was used to visualize and manage contributions. The traction was achieved through pitching the project at forums like the Delft Open Hardware and within connections of MIT



Figure 40 Guest talk at Delft Open Hardware, TU Delft Library, brought new collaborators

Innovation Bootcamps.

6.4. Revenue Model

The inspiration is from the sales of Arduino. It is open-sourced, yet have a successful and sustainable revenue model. Their brand name is Trademarked, and in spite of competition from the low-cost Chinese clones, the Italian company is able to sustain due to the quality they provide. Likewise, the major revenue stream for CuriO is through sales of the finished products. The pricing is based on a contribution-based discount model. The designs are released under MIT Licence as open-source hardware, wherein the primary consumers of the design are computer engineers/ scientists, makers, accessibility testers and researchers. The various issues and feedback from the end-users, as well as further development work, are posted as issues on the webshop page. Anyone (primarily the design consumers) who contribute to these issues, get a discount based on the level of the solution provided. This model ensures an incentive for R&D while cutting down costs for the same (hiring developers, researchers, and engineers).




| Customer Segment | Desired Value | Production Method | Production Costs | Gross Profit /Gain |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------|-------------------------|------------------|-----------------------------------------------------------|
| End users Educators Blind schools |  Finished Product | Bulk manufacturing | 250 Euros | 425 Euros (Margin: 175 Euros) |
| Researcher Designer Developer |  Contribution based discounts | Bulk manufacturing | 200 Euros | 350 Euros (Margin: 150 Euros) + Product Development |
| Maker/ Tinkerer |  Technical Designs | On-demand Manufacturing | 450 Euros | 0 Euros + Product Development |

Table 2 Revenue Model

On the other hand, if one does not want to opt for discounts or does not possess the necessary skills to contribute, they have the option to buy the finished goods. This way, revenue is generated positively

through open sourcing while promoting the spirit of it. According to my qualitative study, the researchers and developers fall into the tertiary consumers, while the blind end-user (primary) and economic buyer (secondary) still preferred to buy the finished goods, without opting for discounts.

A potential risk of this model could be in the form of a big company like Microsoft or Google, with higher resources manufacturing the finished good after taking in the improved designs. This risk could be mitigated by partnering with them and not have them as competitors. According to the MIT Licence, the designs and those derived from the original design should still attribute to the original innovator and license shall remain open source. In this case, since giants prefer acquisition rather than developing a similar product in-house, the developments by such big companies could also help back my team in improving product sales.

Popup Workshops

The secondary revenue stream shall be conducting training workshops, in special schools, mainstream schools, coding clubs, pop up spaces, and after school learning spaces. This also acts as one of the main promotion strategies for publicizing the work while generating revenue. Partnering with Laboratia, an organization that trains women in Latin America in coding skills and empower them with job opportunities, can help Curio to reach media attention and tech giants like Google and Facebook.

6.5. Cost Structure

The finished product would be priced at 425 EUR. More on the cost of goods sold can be found in Appendix 7.3.

6.6.Promotional Strategy

Firstly, the plan is to do digital advertising through a combination of Pay Per Click (PPC) to gain actual clicks to our online stores (partner) and Pay Per Impression (PPM) to gain brand visibility. Also, to use display and affiliate marketing, SEO, and SEM to reach as many targeted digital users as possible.

Influencer Based Marketing: Product reviews by blind programmers and educators on their own Youtube channels are becoming increasingly popular for others in the target group and blind communities willing to buy a product. We plan to reach out to a few such influencers to cover our product in their next video, which can get us the brand and product visibility to thousands of youtube channel subscribers.

6.7. Financial Plan

The plan is to build three more prototypes for testing and validation with blind schools and programmers, after graduation. The insights and reviews from these validation sessions will be used to raise the initial funding of €200,000 through the Kickstarter campaign to bring the prototype to a low volume production phase for the first 100 early adopters. Kickstarter levies a fee of 5% of total funds raised. To scale up production and penetrate the market, venture capital investment will be sought.

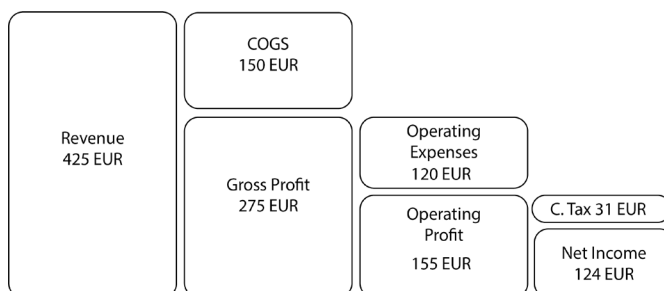


Figure 41 cost structure

Conclusion

The chapter allowed to explore the different business potential of the design solution developed. Since the project was open sourced, the impetus for further development was developed. Towards the end of the project, two more contributors have shown interest. When the visually impaired community, as part of the end user, starts to engage in further development within the Open Hardware model, that is when the business approach will be regarded as successful.

7. Conclusion

- 7.1. Recommendations
 - 7.2. Next Steps
 - 7.3. Reflections
 - 7.4 Conclusion
-

7.1 Recommendations

A list of recommendations were developed based on the learnings throughout the project. It could be useful for the researchers who are willing to take up this approach into programming education. The list has been divided into software, user testing, embodiment and control system.

Software

1. Make product training/ tutorial like a tangible gaming experience, for lesser learning curve
2. Create audio based startup messages when logging on to the system, for easier navigability
3. Create a landing page for market analysis and willingness to buy the product

User Testing

1. More SUS tests have to be conducted with the target age group (14 to 17 year old) blind students
2. Perform playful evaluations such as PLEX framework and safety- risk evaluation for possible use and misuse of the product

Embodiment

1. Support lattice joints with an internal chassis framework, so that it survives a fall or any other mechanical damage to the housing
2. Remove the 15 degree tilt, as user tests revealed that it is not ergonomical to always lean forward and interact with the product with hands suspended in the air

Control System

1. Use of micro stepper motors will reduce the height of the product by 38mm.

2. Use auto-indent scripts to include indentations to curly bracket languages so that the product becomes functional in other languages other than python.
3. Make a pypi installer to easily deploy scripts in a Raspberry Pi
4. Replace Raspberry Pi 3B+ with a Raspberry Pi Zero W to cut down costs and complexity (but include a mics hat)
5. Print lead screws and travelling nut using Nylon 12 filament for its self lubricating and mechanical strength
6. Create more resolution (slider rows) for better interaction experience.

7. 2. Next Steps

The personal intention of the project is not only to fulfill the academic requirements of the master's program but take it forward towards community impact. The interest shown by the enthusiasts at the Delft Open Hardware movement has given the momentum for the project to evolve forward. The next step would be to adapt the approach to explore programming education materials further with Programming Education Research Lab (PERL), as it is one of their main research areas. It could be followed by developing the next version (see appendix 6.4) before it can be tested with the actual target audience- high school blind children and perform more experiments with them. There is also a plan to take the project forward as a startup, having learned the methods and principles of entrepreneurship at TU Delft.

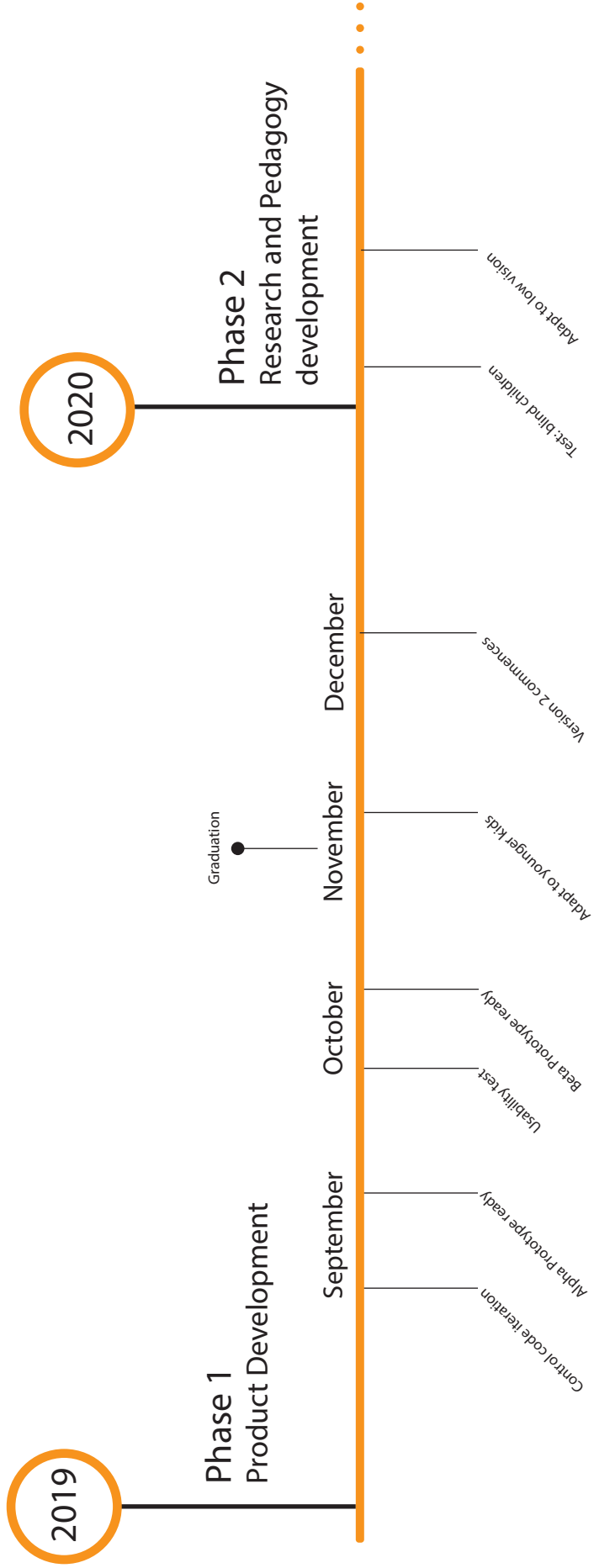


Figure 41 Roadmap for future version

7.3 Reflections

It has been a fulfilling journey to have worked on this project. When I visited blind schools in India for the co-creation session, the innovation officers were really motivated to help me in every possible way, and I could see their deep aspirations and desires to get better opportunities in life. I could empathize how progress in this domain could actually impact their lives, and this is what drove me to push my limits, and whenever I broke down, the emotion is what motivated me to move forward, despite the fact that it was challenging more than my capabilities. There have been several learnings academic and person along the way, which I would like to share in this section.

The complexity of the project, with some of its aspects being very new to me, such as programming in itself, made it hard at times to make decisions. But it was this complexity that motivated me to learn and use research methods that were totally new to me and different methods to analyze the information collected.

I could now say, with contentment, that I have learned what it takes to be a product designer. Having a background (Bachelor's) in Mechanical engineering, I often slipped into the engineering way of solving a problem. I would like to thank my mentors, who advised me at the right moments.

Although my mentors suggested me not to become very ambitious, at times I spent most of my time developing the approaches in hardware-software integration, I could not come up with a simpler and dirty way of validating the concept without really making it work. Though it took away a lot of time, I am glad when I reflect on how this actually proved the feasibility and spark enthusiasm among the Delft Open Hardware community.

7.4 Conclusion

Looking back, what started out of inquisitiveness and a personal desire to contribute to the visually impaired community, has led to the design of an approach, if not towards a finalized design product. There is so much more to learn and fail and learn from that. There are many unexplored questions within the scope of this project, and with that, a true conclusion would be to say that it is just the beginning.

References

Limburg, H., & Keunen, J. E. (2009). Blindness and low vision in the Netherlands from 2000 to 2020—modeling as a tool for focused intervention. *Ophthalmic epidemiology*, 16(6), 362-369.

Vision impairment and blindness. (n.d.). Retrieved November 6, 2019, from http://www.who.int/mediacentre/factsheets/fs282/en/.

E. Glinert (1986). Towards “Second Generation” Interactive, Graphical Programming Environments, Proceedings of the IEEE Workshop on Visual Languages.

O. Zuckerman, T. Grotzer and K. Leahy (2006) Flow blocks as a conceptual bridge between understanding the structure and behaviour of a complex causal system. In Proceedings of the 7th international conference on Learning sciences (ICLS '06). International Society of the Learning Sciences, 880-886

A. Stefik, C Hundhausen and D. Smith (2011). On the Design of an Educational Infrastructure for the Blind and Visually Impaired in Computer Science. In Proceedings of the 42nd ACM Technical symposium on Computer science education, 571-576

The 7 Most In-Demand Programming Languages of 2019 ... (n.d.). Retrieved November 5, 2019, from https://www.codingdojo.com/blog/the-7-most-in-demand-programming-languages-of-2019 (https://www.codingdojo.com/blog/the-7-most-in-demand-programming-languages-of-2019)

Y. Goertz, B. van Lierop, I. Houkes and F. Nijhuis. (2010). Factors related to the employment of visually impaired persons: A systematic literature review. *Journal of Visual Impairment and*

Blindness, 104(7), 404

Villar, N., Morrison, C., Cletheroe, D., Regan, T., Thieme, A., & Saul, G. (2019, April). Physical Programming for Blind and Low Vision Children at Scale. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (p. INTO03). ACM.

Apple Partners with RNIB to Bring its Everyone Can Code Curricula to Blind and Low Vision Students in the UK. (n.d.). Retrieved November 5, 2019, from https://www.applevis.com/blog/apple-partners-rnib-bring-its-everyone-can-code-curricula-blind-and-low-vision-students-uk (https://www.applevis.com/blog/apple-partners-rnib-bring-its-everyone-can-code-curricula-blind-and-low-vision-students-uk).

Moors, L., Luxton-Reilly, A., & Denny, P. (2018, April). Transitioning from Block-Based to Text-Based Programming Languages. In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (pp. 57-64).

Stefik, A., & Ladner, R. (2017, March). The quorum programming language. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 641-641). ACM.

Schanzer, E., Fisler, K., & Krishnamurthi, S. (2013). Bootstrap: Going beyond programming in after-school computer science. In *SPLASH Education Symposium*.

Guillaume Pothier, Éric Tanter, and José Piquer. 2007. Scalable omniscient debugging. *ACM - SIGPLAN Notices* 42, 10 (2007), 535-552.

National Federation of the Blind. 2009. *The Braille Literacy Crisis in America: Facing the truth, Reversing the Trend, Empowering the Blind*. Jernigan Institute.

Rex, E. J. (1989). Issues related to literacy of legally blind learners. *Journal of Visual Impairment & Blindness*.

Schroeder, F. (1989). Literacy: The key to opportunity. *Journal of Visual Impairment & Blindness*.

Stephens, O. (1989). Braille—implications for living. *Journal of Visual Impairment & Blindness*.

Ferrell, K. A., Mason, L., Young, J., & Cooney, J. (2006). Forty years of literacy research in blindness and visual impairment. Retrieved from University of Northern Colorado, National Center on Severe and Sensory Disabilities website: [http://www.unco.edu/ncssd/research/literacy_meta_analyses.shtml].

Potluri, V., Vaithilingam, P., Iyengar, S., Vidya, Y., Swaminathan, M., & Srinivasa, G. (2018, April). CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (p. 618). ACM.

Baker, C. M., Milne, L. R., & Ladner, R. E. (2015, April). Structjumper: A tool to help blind programmers navigate and understand the structure of code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 3043-3052). ACM.

Ann C Smith, Justin S Cook, Joan M Francioni, Asif Hossain, Mohd Anwar, and M Fayezur Rahman. 2004. Nonvisual tool for navigating hierarchical structures. In *ACM SIGACCESS Accessibility and Computing*. ACM, 133-139.

Andreas Stefik, Andrew Haywood, Shahzada Mansoor, Brock Dunda, and Daniel Garcia. 2009. Sodbeans. In *Program Comprehension, 2009. ICPC'09*.

IEEE 17th International Conference on. IEEE, 293-294

Andreas Stefik, Roger Alexander, Robert Patterson, and Jonathan Brown. 2007. WAD: A feasibility study using the wicked audio debugger. In *Program Comprehension, 2007. ICPC'07. 15th IEEE International Conference on. IEEE*, 69-80.

Stefik, A., Fitz, K., & Alexander, R. (2006, June). Layered program auralization: Using music to increase runtime program comprehension and debugging effectiveness. In *14th IEEE International Conference on Program Comprehension (ICPC'06)* (pp. 89-93). IEEE.

Diane H. Sonnenwald, B. Gopinath, Gary O. Haberman, William M. Keese III, and John S. Myers. Infosound: an audio aid to program comprehension. In *System Sciences, 1990., Proceedings of the Twenty-Third Annual Hawaii International Conference on*, volume 2, pages 541-546, Jan 1990.

Lay-Ekuakille, A., & Mukhopadhyay, S. C. (2010). *Wearable and autonomous biomedical devices and systems for smart environment*. Springer.

Sanders, E. B. N., & Stappers, P. J. (2008). Co-creation and the new landscapes of design. *Co-design*, 4(1), 5-18.

Paas, F., Renkl, A., & Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. *Educational psychologist*, 38(1), 1-4.

Tuovinen, J. E., & Sweller, J. (1999). A comparison of cognitive load associated with discovery learning and worked examples. *Journal of educational psychology*, 91(2), 334.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning.

Cognitive science, 12(2), 257-285.

Baddeley, A. D., & Hitch, G. J. (1994). Developments in the concept of working memory. *Neuropsychology*, 8(4), 485.

Designing for Screen Reader Compatibility. (n.d.). Retrieved November 6, 2019, from <https://webaim.org/techniques/screenreader/#linearization>.

Moreno, R., & Mayer, R. E. (2000). A learner-centered approach to multimedia explanations: Deriving instructional design principles from cognitive theory. *Interactive multimedia electronic journal of computer-enhanced learning*, 2(2), 12-20.

Rayner, K., Foorman, B. R., Perfetti, C. A., Pesetsky, D., & Seidenberg, M. S. (2001). How psychological science informs the teaching of reading. *Psychological science in the public interest*, 2(2), 31-74.

Siegfried, R. M. (2006). Visual programming and the blind. *ACM SIGCSE Bulletin*, 38(1), 275. doi: 10.1145/1124706.112142

Jackson, R. M. (2012). Audio-supported reading for students who are blind or visually impaired. Wakefield, MA: National Center on Accessible Instructional Materials.

Choi, K. Y., Sumini, V., & Ishii, H. (2019, June). reSpire: Self-awareness and Interpersonal Connectedness through Shape-changing Fabric Display. In *Proceedings of the 2019 on Creativity and Cognition* (pp. 449-454). ACM.

Tung, M. J., Ko, W. S., Huang, Y. T., & Yang, M. D. (2013). U.S. Patent Application No. 13/710,465.

Shilkrot, R., Huber, J., Meng Ee, W., Maes,

P., & Nanayakkara, S. C. (2015, April). FingerReader: a wearable device to explore printed text on the go. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 2363-2372). ACM.

Carter, T., Seah, S. A., Long, B., Drinkwater, B., & Subramanian, S. (2013, October). UltraHaptics: multi-point mid-air haptic feedback for touch surfaces. In *Proceedings of the 26th annual ACM symposium on User interface software and technology* (pp. 505-514). ACM.

Koh, J. T. K. V., Karunanayaka, K., & Nakatsu, R. (2013, September). Linetic: Technical, usability and aesthetic implications of a ferrofluid-based organic user interface. In *IFIP Conference on Human-Computer Interaction* (pp. 180-195). Springer, Berlin, Heidelberg.

Sodhi, R., Glisson, M., & Poupyrev, I. (2013, October). AIRREAL: tactile interactive experiences in free air. In *Proceedings of the adjunct publication of the 26th annual ACM symposium on User interface software and technology* (pp. 25-26). ACM.

Nakagaki, K., Fitzgerald, D., Ma, Z. J., Vink, L., Levine, D., & Ishii, H. (2019, March). inFORCE: Bi-directional-Force Shape Display for Haptic Interaction. In *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction* (pp. 615-623). ACM.

Schneider, O., Shigeyama, J., Kovacs, R., Roumen, T. J., Marwecki, S., Boeckhoff, N., ... & Baudisch, P. (2018, October). DualPanto: A Haptic Device that Enables Blind Users to Continuously Interact with Virtual Worlds. In *The 31st Annual ACM Symposium on User Interface Software and Technology* (pp. 877-887). ACM.

Tassoul, M. (2005). Creative facilitation: a Delft approach. VSSD.

Tassoul, M. (2006) Creative Facilitation: a Delft Approach, Delft: VSSD.

Shinohara, K., & Wobbrock, J. O. (2011, May). In the shadow of misperception: assistive technology use and social interactions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 705-714). ACM.

Nielsen, J. (1994b). Enhancing the explanatory power of usability heuristics. Proc. ACM CHI'94 Conf. (Boston, MA, April 24-28), 152-158.

(George Lakoff and Mark Johnson. 2003. Metaphors We Live By. University of Chicago Press

Guiard, Y., & Beaudouin-Lafon, M. (2004). Fitts' law 50 years later: Applications and contributions from human-computer interaction. International Journal of Human-Computer Studies, 61(6), 747-750.

Garner, S., & McDonagh Philp, D. (2001). Problem interpretation and resolution via visual stimuli: the use of 'mood boards' in design education. Journal of Art & Design Education, 20(1), 57-64.

Brooke, J. (1996). SUS-A quick and dirty usability scale. Usability evaluation in industry, 189(194), 4-7.

Hart, S. G. (1986). NASA Task load Index (TLX). Volume 1.0; Paper and pencil package.

Rozenburg, N.F.M. and Eekels, J. (1995) Product Design: Fundamentals and Methods, Utrecht: Lemma

Kappers, A. M. (2007). Haptic space processing--Allocentric and egocentric

reference frames. Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale, 61(3), 208.

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. ACM Transactions on Computer-Human Interaction (TOCHI), 7(2), 174-196.

Appendix

Appendix 1

1.1 User Interviews

Participants

Initially, it was planned to do observation studies with blind high school children and setup interviews with the computer science teachers. But due to time constraints requests were made to interact with blind children at their respective schools. However due to confidentiality aspects of this vulnerable group of children, the requests were turned down. So, adult blind programmers from outside of Netherlands were contacted to gain more insights on their challenges. Hence, adult blind programmers were interviewed. Though their needs, aspirations and behaviour towards programming would be different from that of high school children, it was assumed that they would still represent the blind community. Also, their coping strategies to circumvent the current challenges in textual programming could inspire creative ideations.

The participants were reached out through blogs, linkedIn and through personal connections. Four blind programmers from Poland, Australia and India participated in the interviews. Two blind non-programmers were also contacted, to gain insights on how new learners (without coding skills) interact with the computer.

Method

The semi-structured (qualitative) interviews were conducted through skype and their conversation were recorded (audio).

Questionnaire- Programmers

1. Can you tell me a little about yourself, educational background and what kind of visual impairment you have?
2. When it comes to screen readers, what is your favourite? What are their limitation in your experience?
3. What accessibility aid like screen readers, braille displays etc do you use and why?
4. Is Braille going obsolete? Do you use braille note takers, how has been your experience with it so far?
5. Do you code? What languages and what are the major issues you have had in coding?
6. How do you navigate and skim through code? Any personal strategies?

Condensed Transcript

Participant 1

1. I was born with 2 to 3 % visual ability in the right eye and complete blindness in the left eye. I could see shape and colours vaguely in the right but nothing in the left.
2. I use voice over. But it doesn't read paranthesis that allows me to format what I write.
3. I use braille displays together with screen readers. Braille displays are good because they allow me to feel the code. But I cannot read things like hierarchies. I have to mind map them in my head.
4. Braille is very useful. But people find it difficult to access it. For example, braille books are heavy to carry and

not all places print braille versions. In terms of computer use, notetakers (displays) are very expensive and the braille in it is not the same as printed braille. Anything to touch and feel are really useful when inexpensive. Also, braille helps to utilise muscle memory rather than having to focus only on what the screen reader says.

5. Yes, I code. I learnt to code when I was in grade 8. I now use Xcode in my Mac and C++. I use only command line to code.

6. To skim through code, I read line by line, word by word to get a gist of it and this takes a lot of time.

Participant 2

1. I am totally blind from birth and I did my Ph.D in Computer Science from the Polish Academy of Sciences

2. I use NVDA for screen reading. The difficult part is read all the content line by line and understand abstractly what is happening.

3, 4. I use both screen readers and braille displays. Braille displays are useful to bring the cursor to a specific location. But it is not possible for me to do parallel tasks, like using braille and listening to screen reader at the same time and because I do not read braille so well. Screen reader is fast reading, while braille display is for specificity.

5, 6. I like F# because it uses one third of the number of lines that are normally used, to write a program. HTML is good for start as it has quick feedback by refreshing. Screen readers often work without any incompatibility issues with Visual Basic. Python is bad for blind people, because it comes with white spaces. NVDA makes different tone of beeps for different indents. This will drive you crazy especially when you are coding for like 10 hours. Languages that use curly

brackets are also not good. As a blind person, I cannot differentiate which code block is closing which one is opening, nor I can understand which block does a certain code is located.

6. Did you just say skimming, that is not possible for someone who is blind. I will have to read by line. To navigate, I see how certain functions are coded and use find function or class or get parameters to navigate around. In terms of syntax feedback, since I use visual studio it gives me sound alerts which are short and clear. For debugging, I use IRcons, that make a peculiar sound when I am on a line with warnings or errors.

Participant 3

1. I lost my vision due to an explosion. I did my bachelors in Computer Science and a masters in mathematics and Artificial Intelligence.

2. I mostly use NVDA. The major drawback I see in screen readers and coding with it, is not being able to have an overview. When there are hundred's of classes, I have to mind map them. So I always have text editors open. Though I can only sequentially process sensory information (tactile and listening to screen reader), I can parallely process in my head.

3. I use only screen readers, as I do not read braille. I lost sight later in life and so it is hard to learn braille now. Also, when I tried to learn to code using braille, I found it to slow down my work.

4. I do not know much about if braille is going obsolete or not. But I do not read braille.

5. I like python, as it gives freedom. I also use PHP and Swift.

6. I use a software that produces 3D spatial sound that I developed myself. This helps me feel where I am in the

code. To navigate, I use find and replace feature in some tools, but this applies only after I read the whole code line by line and I know what function or class has been called, to find it later.

Participant 4

1. I am blind since birth, and have been working.

2. I use NVDA. But it does not work properly with text editors. Also with some IDEs with which it is compatible, it doesn't give live feedback, but only on compilation. I really do not mind listening for a long time, as long as it can provide correct feedback.

3, 4. I prefer audio feedback over braille. For me learning braille takes more time. Also it takes a lot of time to read technical or complex details with braille. Speed of comprehension is important. Also braille displays are very expensive and inaccessible in India, that I have only seen them in some recognised blind institutions.

5, 6. Yes I code. HTML is my favourite as it allows me to jump between header tags. IDEs tell me where the error is.

1.3 Expert Interviews

Goal of the interviews was to understand aspects of education to blind high school students, in programming education using screen readers.

Participants

High school teachers specializing in inclusion of visually impaired children in regular schools, were interviewed from Visio and Bartmeus organizations. Interviews were in-depth semi-structured and was carried out in Dutch and was helped by Benjamin Bosdijk, a bachelors student from PERL. Purposive-Homogenous sampling was used.

Transcript- Visio (English)

Q: so to start off with, what is your function within this school and what do you primarily do?

Originally I am a mathematics teacher, but quite rapidly that evolved into ICT business and primarily on 'how visually impaired children (low to high visual ability? can operate a computer'. You have different aspects for instance what kind of training is necessary. What kind of aid is necessary. What do the teachers need to know to continue working with the children. These are the aspects that I'm working on at the moment. It used to be a big range of visually impaired children but now it's really focused on braille students.

Q: is the use of braille in this scenario increasing or does it decrease?

It seemed to be decreasing because of the speech synthesis that came to the foreground. But if you look at the current situation.. we ran into some difficulties, and a study emerged in which it was found that braille was actually quite important. And now we are actually directing our vision towards incorporating more braille. And for this reason braille is an important component, within our school other visually impaired Schools. more and more is paid to braille.

Q: what type of visual impairment do the children have at this school?

in between seeing everything and seeing nothing is a lot and you have a wide range of diseases, people with a brain tumor, people who have it since birth, some people might have a mental illness, people who become blind at a later age. there are so many types of diseases. and there's also a wide variety in what they can see. some students I have seen putting a thread through a needle, and you wonder to yourself how is it possible?

Q: Are there in general any things that have been altered in their surroundings to adapt to them?

are you speaking specifically about technology? Mobility: different colors for doors. white= toiler, yellow=meeting room, red=class room. Every door has a different feeling in front of the floor. Development of last 7 years in technology have not been major. with the coming of iphone ipads it was really big. The accessibility of the visually impaired was certainly enlarged through these items.

Q: Is it difficult for kids to switch to new assistive technologies ?

A: Actually no. A boy who started to work with the ipad (completely blind) has no experience before. he thought it was important to look for games, he wanted to search the internet. What is important? He needs to put the device on, the braille machine should be connected to the ipad. he needs to know where all the buttons on the screen are. In 45 minutes you can get a far way, he knows where all the apps are. The learning time is fast. It is far easier than 'how to send an email with an extra attachment'. In the approachability/ accessibility of new products there is more consideration for visually impaired compared to 25 years ago. in those days the products were really made for 'us' but in these digital devices it is a standard to have tools for the visually impaired.

Q: how do the physical or other restrictions influence the use of computers and learning?

There are a couple of multiple disordered children, which have some difficulties in the use of computers. They do use it but far less than the children who have more ability. Good apps can assist them, compared to 10 years, there was a lot less to offer these children. It can help them communicate with their family, such as whatsapp..

Curriculum

What sort of computer education is given?

Biggest problem, not being able to read. Blind people need to use braille. Bad sighted: vergrootglas. We try to get them behind the computer as fast as possible. Where all the settings can be changed: color, size. Teach them how to use the computer. To do their homework. Cursus teaches, how to do the tests, how to open the documents and how to open the word. Internet is not easy. And depends per website. Internet easier on hand held/ iphone/ipad than computer. On website you have multiple tabs, and not really on ipad.

Educational: You need to learn english, dutch, other courses. Use of educational software is also tough.

Do all students with different kinds of visual impairment follow the same kind of computer lessons?

Everybody has the same hardware, laptop. Some children might need assistive technology, depending on the program. Such as scan software.

What types of software are used in coding education?

Programming is started to be thought at the kindergarden with programmable robot kits. And then later python to control a drone.

Which programming language is first taught?

Microbits kits are used in general at the beginning, but can be difficult because it is so difficult to visualize. Can be difficult with the colors. Though you can zoom in. Some investment is needed for python to teach children, but it is more accessible. Navigation of the screen is difficult. Program for navigation is still missing. (use of notes then copy into a python runner, is difficult) We can make our way to do it, but it hinders enthusiasm.

Q: What is the most important difficulties in textual programming languages?

Navigation from place to place, with the use of anchors or headers, but this is missing in coding language. Reading is ok, but a lot of energy is used to make the code itself.

Computer use:

Now we want to elaborate further on the use of assistive technologies in computer education. We are mainly interested the effect of screen readers.

Q: Which screenreaders are used?

NVDA is for the hardcores. Supernova is not used anymore
Reminding some code-names is hard.
JAWS software is good, but its licensed.

Q: How does the screenreader influence the education?

Teachers incorporate headers to put in their documents so children can easily navigate. Bold italica colors and underscore their can read. Teacher of jaws knows these things regular teachers can have some difficulty to understand jaws. You need to know what you can expect from children.

There is a learning line, for each child, to show at what age they will learn something, and who teaches this.

The internet opens up a lot of information for children of the current day and age.

Q: When purchasing assistive technology, where is this taken into account?

What considerations play a role? E.g. standards, costs.

There is not alot of choice. For braille reading there are only a 8 machines suitable for our target group, the prices are close to each other. Quality is similar. When one is broken, and i give a child a different brand they easily adapt to it. Braille reading guides and braille printers. Everything they read they can digitally

understand. And for a presentation they like to have a paper. (printed with braille printer).

Some people have speech synthesiser. And zooming in software, can be in jaws. Supernova nad windows enlargement, quite stable. Also speech and they can be put int brail reading.

Q: Educational software, does it support this system?

No that is one of the biggest problems, the publishers make math/economy books and those are used. These books sometimes now rely on external websites for exercises, using a login code. Then the website is badly made. They use visual langauge, They cannot see the image and then they cannot understand that they have to click, not java but CSS. It is not the case that the speech software cannot read the website but it is not recognised as text. For instance, i cannot read the images, they need a TAG. Pictures need readable captions. (NEXT images are not labelled as NEXT but as picture 9192 - not understandable) Some pages are with pictures with colors too close in contrast. Some questions want drag and drop. Accessibility can be improved.

Typing lessons are specially made.

Some software packages are completely unusable, some schools still use them but then these children have to move to other schools, or will go to the visually impaired schools. The publishers are not doing a good job. Treaty of Marrakesh: publishers will try better. They have been around the table often. Some publishers, change the version of that time, but wont remember it in the future.

Technology is there but not implemented. Extra external keyboards are necessary for inputting the code. The cursor could not be moved with the braille reading guide. With a programmer, they can change this.

Is the same difficulty there with a normal computer?

The most easy would be text-based software. If something is not perfect, children cannot continue.

Q: What do you think of the future of braille?

It is important, speech tries to come into the situation, and it is really helpful but for mathematics and stuff it can be difficult. When reading in braille it becomes easier to understand. With increase complexity, the person switching to braille understands more than the one relying on speech (study done). Problem, row of text with a place to put an answer. You can't put the cursor there.

Demonstration of Braille Notetaker

Guy: you should try to use these python programs, and understand what the child can read. It can be that everything is readable. If you can't put a pointer, to dictate, in between these pointers you should copy the text. There are only two possibilities, change the source file or ask Mr. Jobs to write a script for this program. But rather have it fixed at the source, so that every new version of the software is supported. These small things inhibit the progress.

It is not the case that they lose the point where they are working in the code, but the problem is the tuning of the program with their aiding tool.

Exact: accounting program, very much difficulty. Sat down with Jaws and them to finish it. They have their own icons, not window-compliant. Most windows-compliant programs are usable, but already if only the lay-out changes it is a problem.

All the buttons are specific actions. Tab is very important, Alt cntrl windows

and shift are usable and have specific functions.

Most people use a keyboard, it is faster than braille? (he says 'this way')
'There is a mouse cursor'

Within 1 line of 120 characters and 3 braille lines of 40 characters, how do i move? You need to use "this button" for the next 40 characters.

They read a bit slower than us, but the speech control they can understand very fast what is said. It is generally learned how to hear speech very fast. If they don't understand the speech (dictation), they don't reduce the speed but read instead.

There is a button that edits the configuration?

From 7-8 they use a keyboard.

Most of the young people at this school are born blind. These children are taught braille. If you can't read with your eyes, you need to use braille. Some student who are not blind, still use braille. With braille they can react fast.

Sometimes it's hard to convince the children to change to braille, because they keep wanting to read on the screen. Sometimes it takes years. First with zooming in, then with speech dictator. And then finally they want to use braille. Some braille readers are faster than than even "i" am with the mouse.

Because of the speech dictator, most parents give their children an expensive Iphone.

Jaws only works with windows.

Some people have the speech commands off, and they have the screen off. But then for a regular person, understanding how to turn on the screen again is very difficult.

If you understand these codes of swiping etc on the smart phone, it makes a world of difference. You can type with the numbers on iPhone using the Braille system.

How can they memorize 15 orders to go back?

You can always go back to the home screen, and apps you can save. Many tricks.

Being able to ask where you are in the code is important. Bridge between software and screen. It should be able to convey all the info we can see, such as color. As a programmer you need to understand how Jaws does that.

Alfa, hired blind people to work on the product. They made a product that didn't actually catch on and are now bankrupt. Alfa started blind himself. Alfa made ???

Jaws has a wide range of actions. Commands can be made to read the text that has certain colors, bold, italic. There are many types for Braille within any language. Computer Braille and literary Braille... Jaws

Mathematical language is just now being translated to the spoken method in speech dictation

Your report is important to us. And any prototype too.

1.3 Online Survey of blind programmers

An online survey was made to learn about the preferences, pain points and needs of professional blind programmers. This also helped in functional analysis of existing tools. Although the needs of a high school blind student would be different from that of a professional, mostly adult participants were selected assuming they would represent the needs of a blind person irrespective of age. The survey was structured such that questions appear based on the previous answer, in a logical manner. The questions were designed The sections were divided primarily based on whether the participant used a screen reader or a braille display to code.

Participants

11 blind programmers voluntarily contributed to the survey. They were reached out through Program-I, an online community of visually impaired programmers. Out of the 11 respondents, 9 were programming professionals and 2 were high school students. Not all questions were mandatory.

Survey Results

What accessibility products do you use for programming?

11 responses

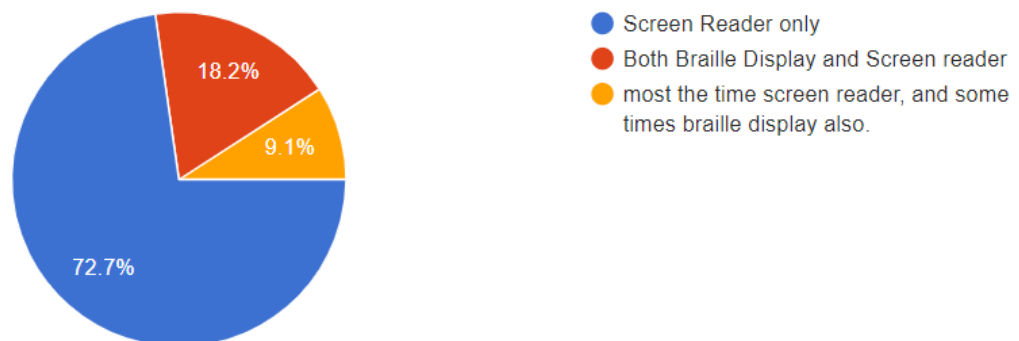


Figure 30 Chart showing preference for kind of accessibility products

How do you skim through long lines of code

Letting the screen reader read by line or using the word by word feature to read through tricky code syntax.

I don't.

reformat with fewer characters per line, if possible reformat line to remove or reorder material

Usually line by line.

Reading of course. :) And spelling if necessary.

ctrl+right/left arrow

Figure 31 List of strategies to skim through code (using screen readers only)

How do you skim through long lines of code

2 responses

Not think fancy, just read the line or use a braille display. Yes, probably haven't met the monsters you're talking about.

Pan the Braille display or navigate with my keyboard (Home, End, Ctrl+right/Left arrows)

Figure 32 List of strategies to skim through code (using braille displays)

How do you get feedback about, if you have written the right syntax or spelling?

8 responses

I know how to type and my ide warns me of mistakes

The spoken line of text exposes typos very clearly. When in doubt, I arrow over the word character by character.

Re-read, and then run code

Emacspeak plays an "Error" sound on the line with syntax errors, if it is detected by whatever command line tools it uses.

check syntax with interpreter or compiler

I turn on punctuation, and sometimes reading of text parameters so that it makes reading easier.

Most often, go to next/previous problem in code. Code a piece first, then check, to fix errors incrementally.

error list, plus the autocomplete stops working

Figure 33 List of strategies used for getting feedback on syntax

Why did you prefer not to use other aids like braille readers?

8 responses

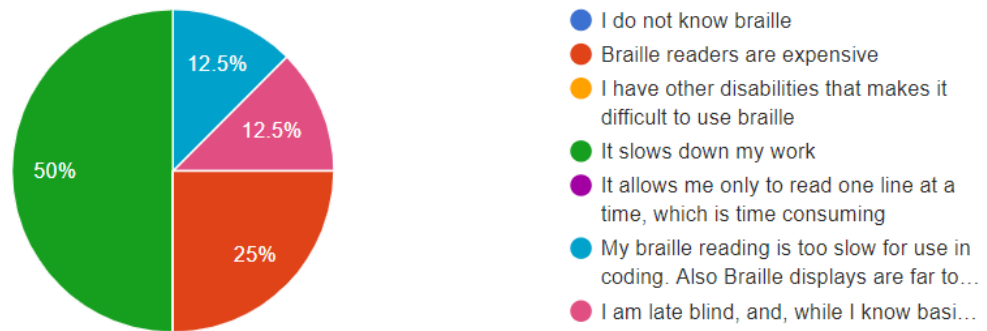


Figure 34 Chart showing the reasons why braille displays are not used

What is your most favorite IDE?

11 responses



Figure 35 Chart showing most preferred IDE

Will you be interested to get a discount in procuring a new accessibility product if you have the option to contribute to the project?

11 responses



Figure 36 Chart showing the preference to receive a discount on contribution

What programming languages do you use often?



11 responses

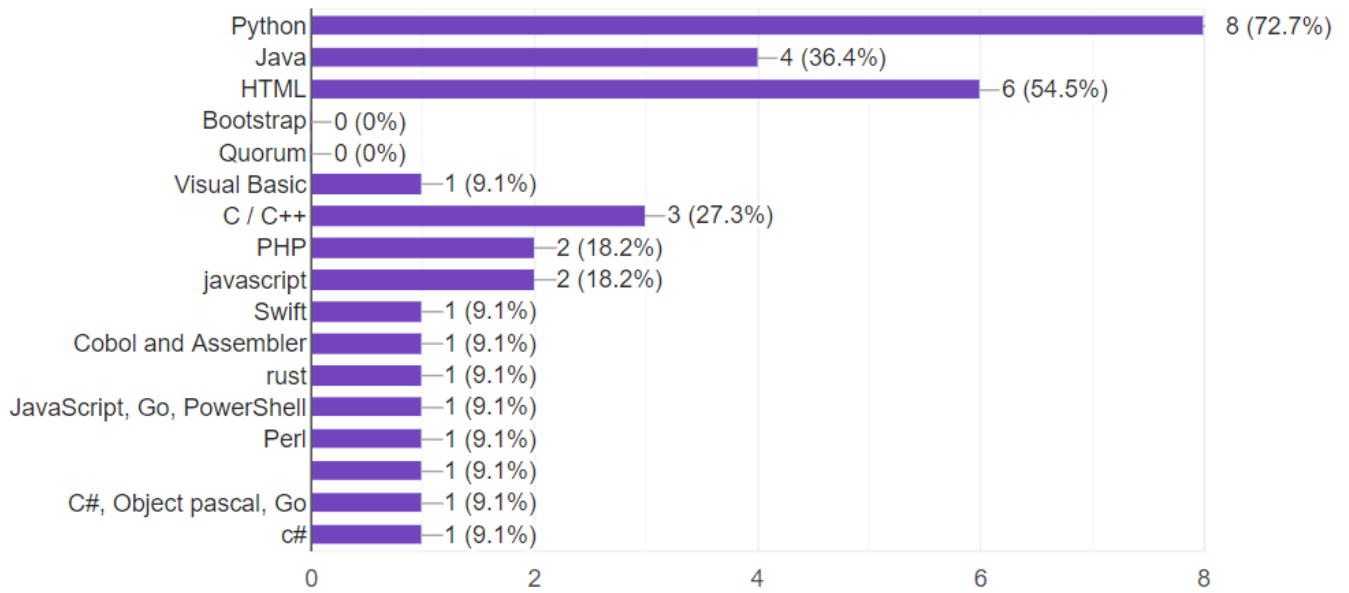


Figure 37 Chart showing the preferred programming languages

Appendix 2

2.1 Technology Readiness Level

Technology Readiness Level (TRL) (ESTEC, 2009) is a metric to measure the maturity of a new technology-based product, see fig 38. The scope of the project, and aspects that require improvement before it can reach the market was defined using the TRL method (see figure 1). Level 6 was aimed for, as this would allow user testing and take the insights forward for research purposes. It was based on factors like time limitations, work complexity, and relevant skills required to achieve the design goals. As shown in figure 1, the planned scope of the project was in developing skimming and navigation capabilities of the product to be able to demonstrate on the field. However, during the design process, real-time interfacing had to be developed to the level of proof of concept to facilitate the field demonstration skimming and navigation. The TRL was continuously visited and reflected upon, while taking design decisions, especially in the concept detailing phase.

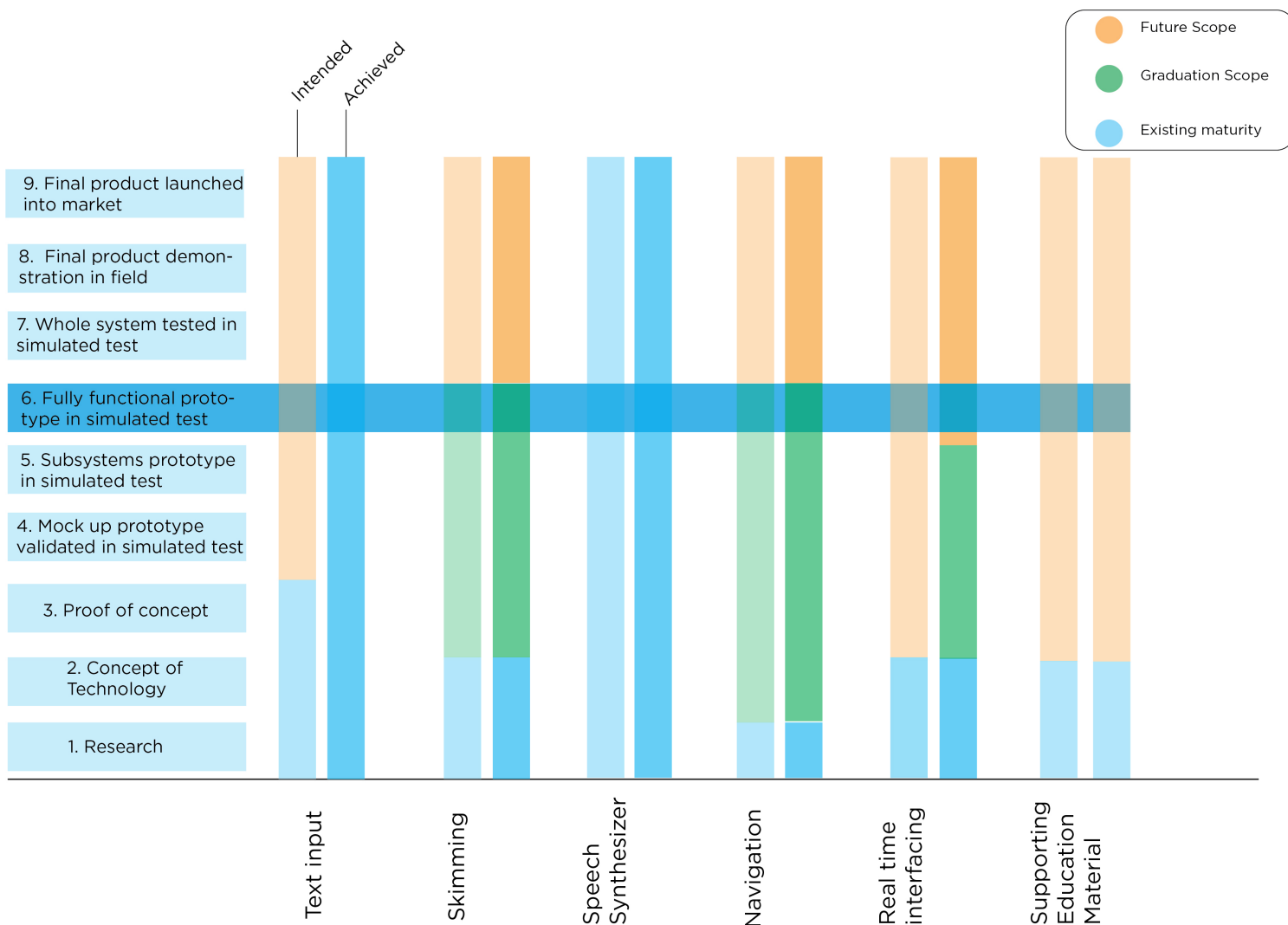


Figure 38 Technology Readiness Level

2.3 Functional Analysis

Introduction

An abstract model of the product and its intended functionality was developed. The supporting functions identified using functional analysis (see figure 39) (Roozenburg, 1995) was used as starting point for creative ideation and as parameters for morphological analysis.

A process tree about using a screen reader to code was first developed. It was based on the observations (appendix 1) of how blind programmers use screen readers to do so currently.

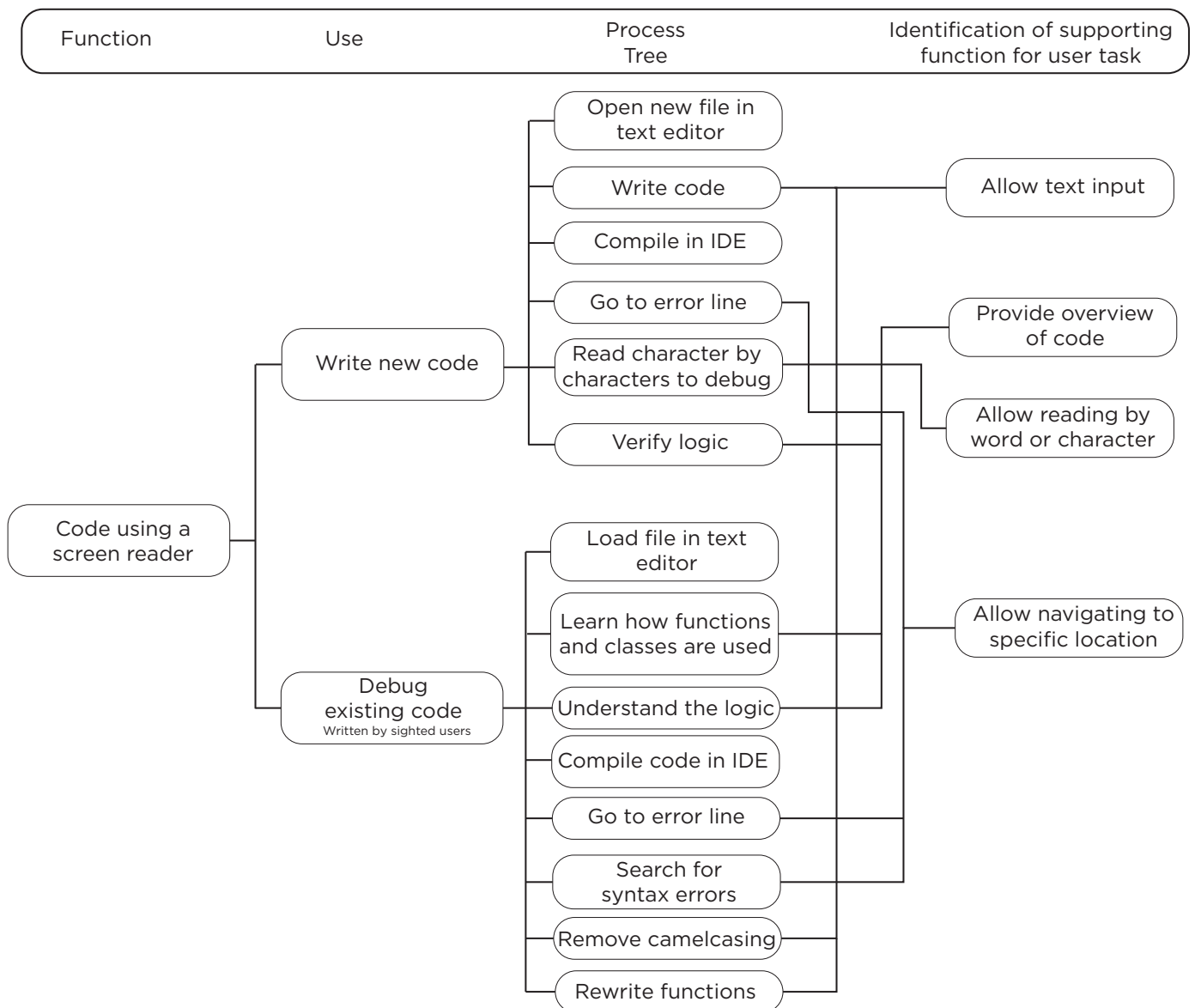


Figure 39 Functional Analysis of screen reader usage by blind professionals to code

Appendix 3

3.1 Group Brainstorming Session

A creative facilitation (Tassoul, M., 2006) session aimed at generating large number of divergent ideas. Design factors identified from the functional analysis (Appendix 3.3) of existing screen reader use in textual programming languages, these functions were taken as the starting point for the creative session. Analogies, metaphors and random stimuli were used.

Participants

Six Students (aged 23 to 29 years) the TU Delft from multi-disciplinary study programs participated voluntarily. The diversity in background was to ensure diversity in perspectives. The basic criteria was novice to intermediary experience in textual programming.

Methodology

The session started with a 10 minute introduction to the problem at hand, session schedule and the design goals. The functions identified from the functional analysis were taken as the design goals of the session:

- To provide overview of code (skimming)
- Allow navigating to specific locations inside the code
- Allow text input

The brainstorming was divided into three parts:

I. Shoe Box: 20 mins

1. A shoe box with a slit at the top was placed on the table. Each participant was asked to sketch 20 ideas onto sticky notes in 20 minutes and drop it through the slit. They were encouraged to think of metaphors of everyday objects and create analogies.

2. As ideas slow down, random sticky notes were taken out of the box to spark new directions or combine with existing and put back to box

II. Dot Voting:

1. The sticky notes were taken out of the box and put on a board.

2. The ideas were clustered theme wise. Each participant was given three sticky dots that they can use to vote the ideas that fit into the design goals.

Braindrawing:

1. The ideas with atleast 2 dots were selected and were drawn on to a bigger paper.

2. Each of these papers were passed around one by one and each participant was given five minutes to add details on top of what is there already.

3. The faciliator (me) kept asking provocative questions using the SCAMPER method (Roozenburg, 1995)

SCAMPER:

Can it be? (provocative questioning)

- Substituted by something else?
- Combined
- Adapted
- Modified
- Put to other use
- Eliminated
- Rearranged or Reversed

4. The iterated ideas were written on a seperate paper

5. The iterated ideas were then evaluated using the C-box method

Evaluation:

The various ideas were evaluated using the C-box method (Tassoul, 2006) (See figure 40)

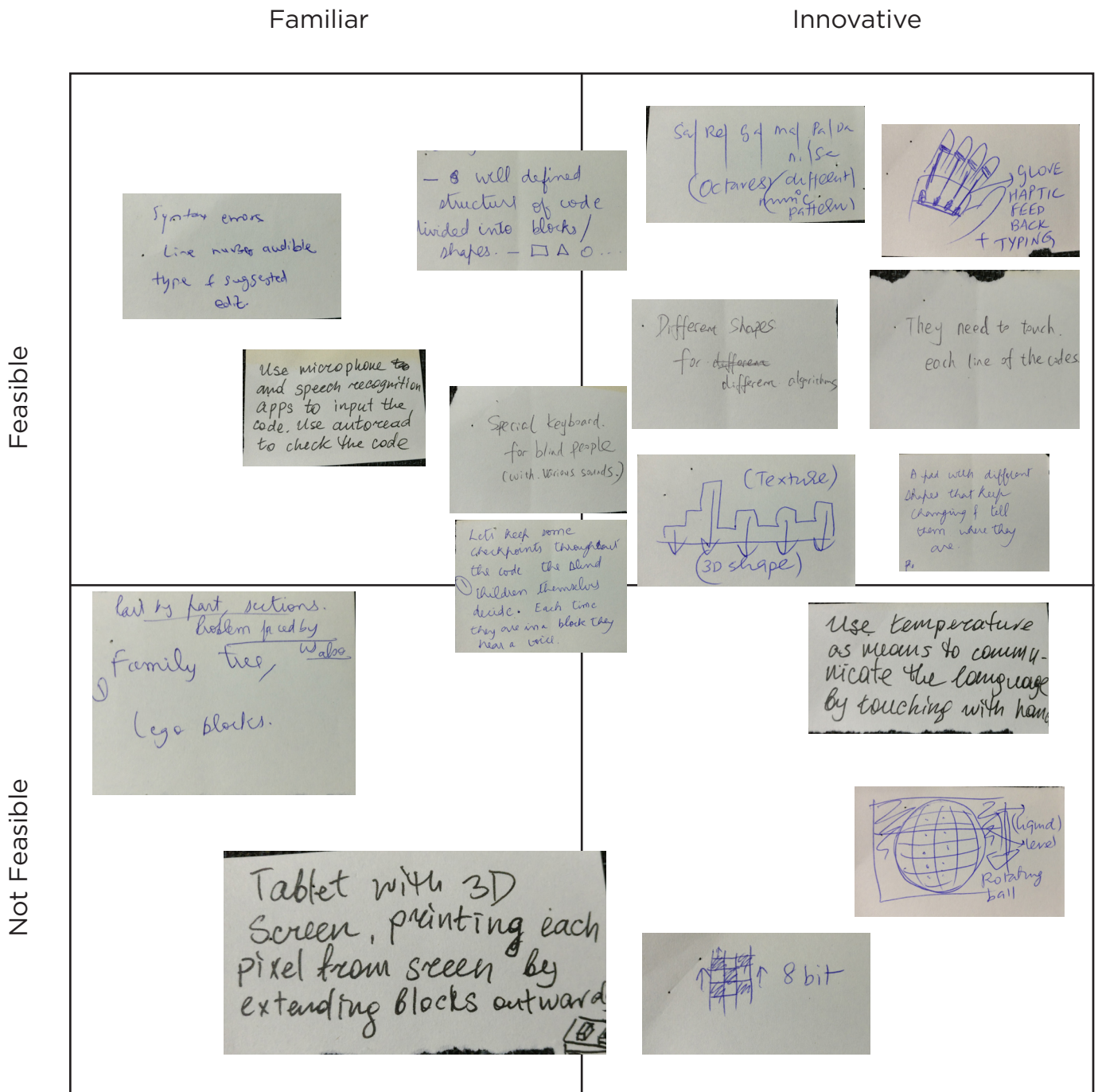


Figure 40 Brainstormed Ideas organized into a C-box

Results:

The most feasible Ideas were added as component domains for the morphological analysis

3.2 Morphological Analysis






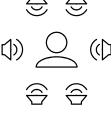



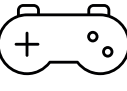



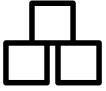
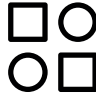

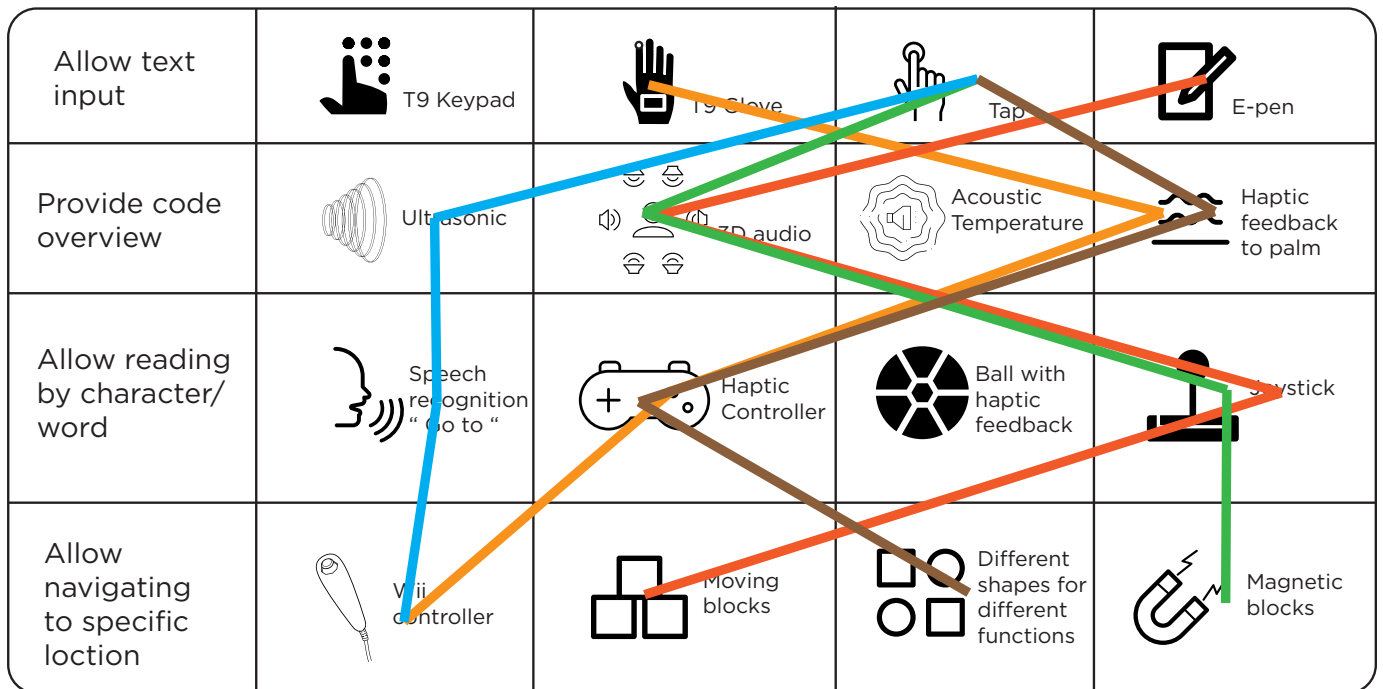
| | | | | |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| Allow text input |  T9 Keypad |  T9 Glove |  Tap |  E-pen |
| Provide code overview |  Ultrasonic |  3D audio |  Haptic feedback to palm |  Haptic feedback to palm |
| Allow reading by character/word |  Speech recognition "Go to" |  Haptic Controller |  Ball with haptic feedback |  Joystick |
| Allow navigating to specific location |  Wii controller |  Moving blocks |  Different shapes for different functions |  Magnetic blocks |

Figure 41 Morphological analysis



● Concept 1
 ● Concept 2
 ● Concept 3
 ● Concept 4
 ● Concept 5

Figure 41 Clustering of components for conceptualization

3.3 Conceptualization and early prototyping

Feasible principle ideas from the creative ideation session were taken towards concept development, after morphological analysis. The ideas were detailed further to form concepts. The focus was on developing concepts for tactile display of

code structures, similar to how they are displayed for a sighted user. Five concepts were defined, prototyped and evaluated. The common task was to identify code hierarchies in a given test program using the mockups.

Concept 1

Clip-on ultrasonic sensors to detect the relative position of the user's fingers pointing on the screen (see Figure 42), inspired by Finger Reader technology. The device gives audio feedback on the

level of the indentation of that particular line. This would work without requiring any firmware/ Human interface Device (HID) drivers on the host computer. A foam mockup was made to assess the ergonomics of the concept (see figure 43).

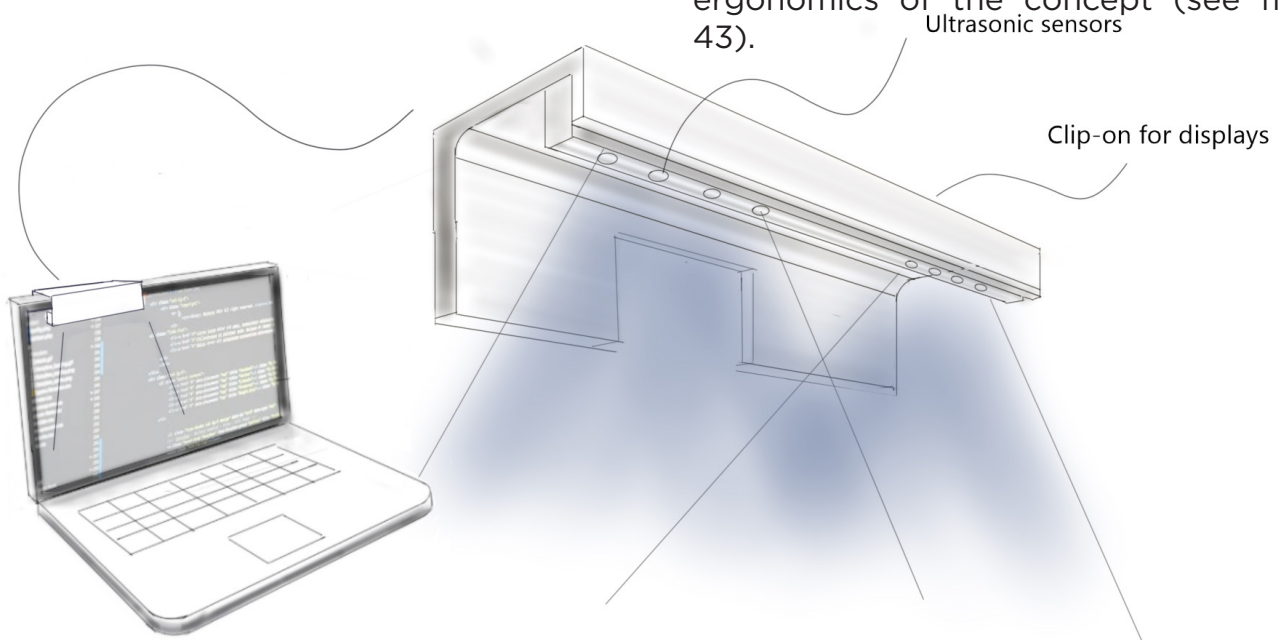


Figure 42 Concept 1: Using ultrasonic sensors to track finger positions

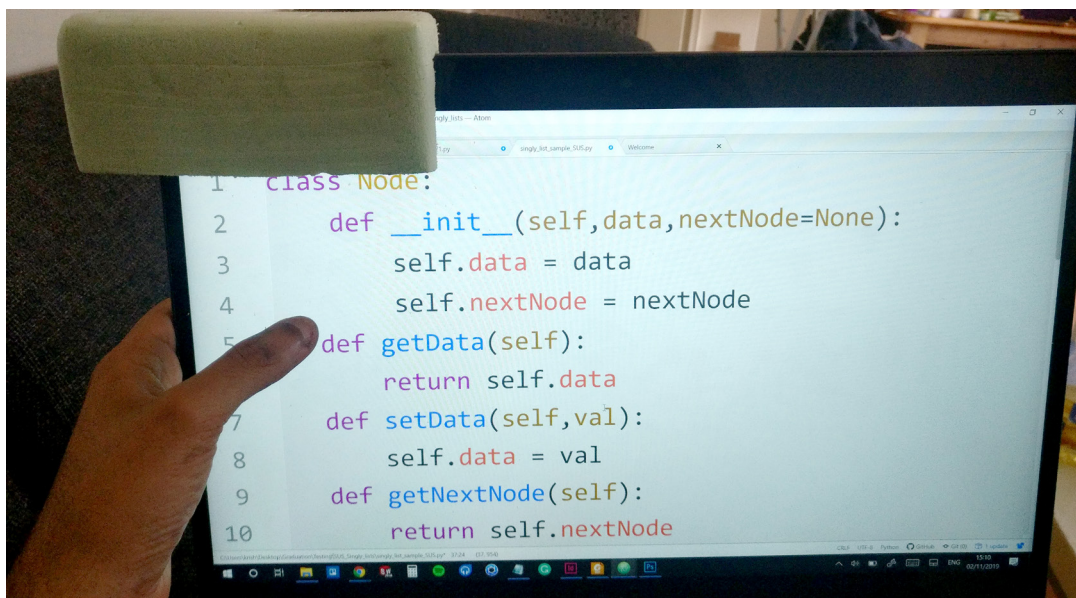


Figure 43 Foam mockup used in test interaction

Concept 2

A hand held device strapped (see figure 44) onto the wrists, with input and feedback systems. The input is based on the T9 dictionary (predictive) text. Each of the 8 fingertips would have flex sensors which returns an alphabet based on frequency. Haptic feedback regarding code structures is provided in the form of vibrations onto the palm, while gestures like swaying hands from left to right, would read out aloud (voice synthesizer) the text for user feedback or navigate between the words.

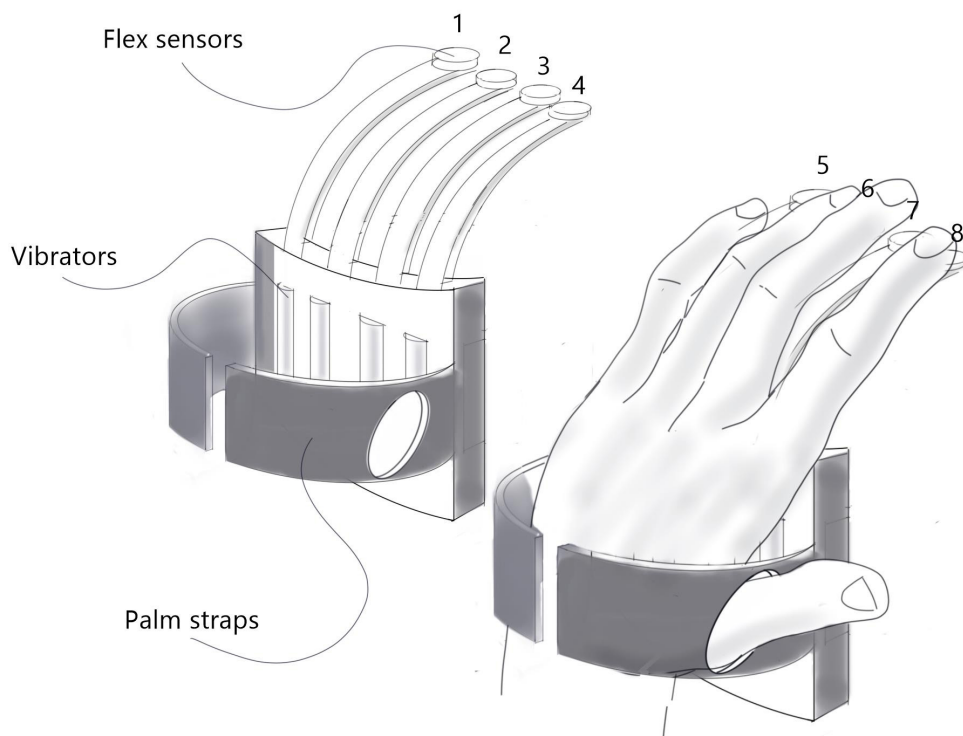


Figure 44 A pair of hand-strapped devices for text input, haptic and audio feedback

Concept 3

Inspired by magnetophoretic displays commonly found in children’s magnetic drawing toys (see figure 45), concept 3 is a grid of electromagnets, and buttons on each of them, to align iron filings into bumps on the surface (see figure 46). A microcontroller reads code structures on the computer through a firmware and magnetizes and demagnetizes these magnets. These iron filing bumps would provide information on code structural information such as indentation. On pressing a bump, the user would listen to a synthesized voice of that code line. A functional prototype was developed (see figure 47). The iron filing bumps formed (see figure 48), but the electromagnets used did not have enough holding force and the filings dispersed when touched



Figure 45 Erasable magnetic drawing boards for children

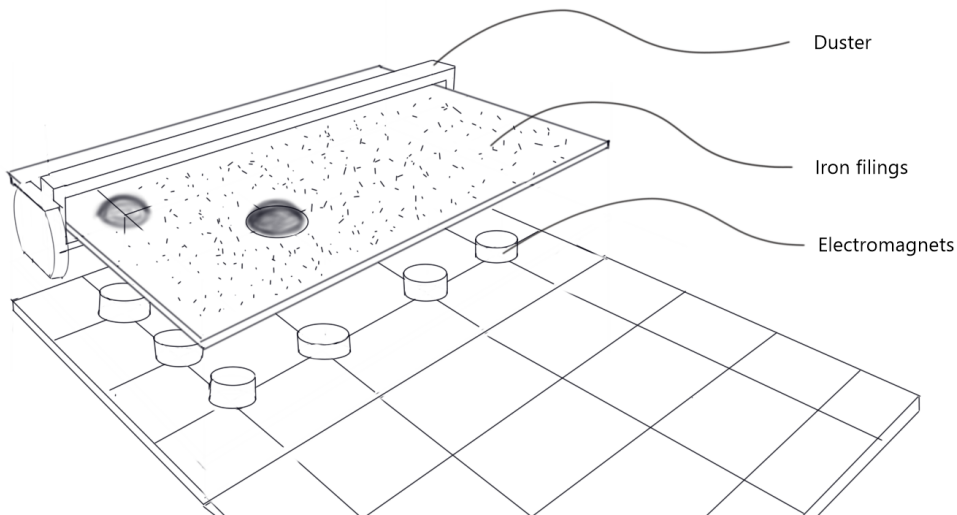


Figure 46 Concept 3: Electromagnetic grid to form tactile bumps on the surface

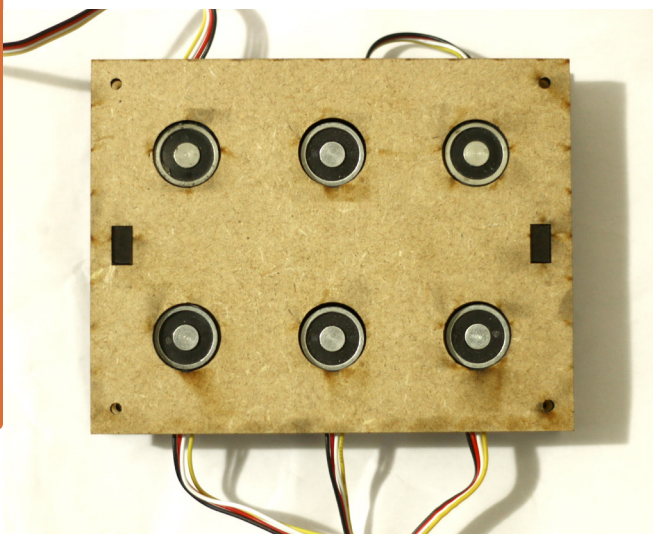


Figure 47 A functional mockup built using six grove electromagnetic modules controlled by an arduino



Figure 48 Testing formation of tactile bumps using iron filings

Concept 4

Rows of tactile tiles (see figure 49) that move sideways along a conveyor like belts. The position of the tiles would be controlled by servo motors and a micro controller. A firmware would translate indentation levels from a text editor on the computer into their positions. A group of belt and rollers, with varying types of tactile texture materials stuck on the belt. Each texture means a class or function.

In this way, users can feel the logic of the code using real time positions of the tactile tiles.

A foam mockup (see figure 50) was built to evaluate the concept. Sanding paper pieces were stuck to the foam board at positions mimicking the indentation levels of each line in the test program.

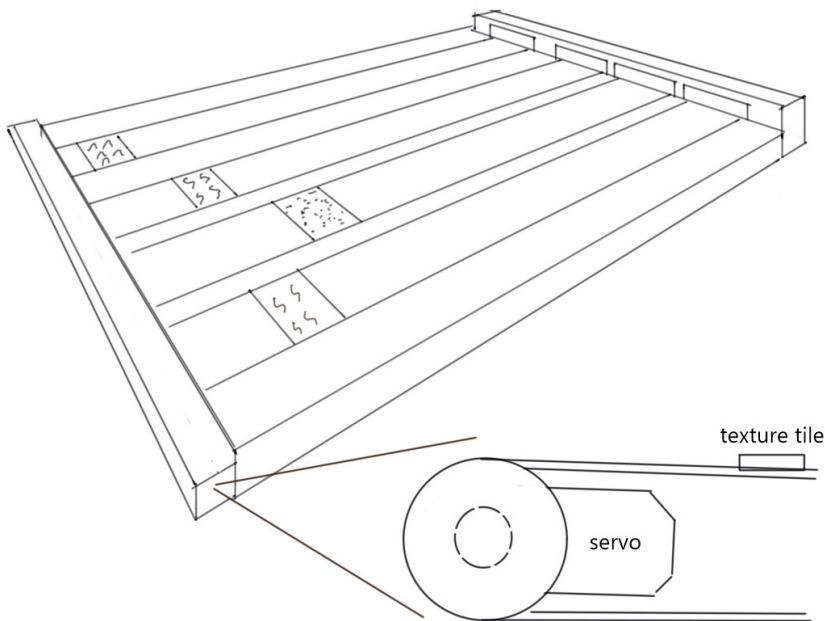


Figure 49 Different tactile surfaces on their respective conveyer belts, that represent code hierarchies



Figure 50 A foam mockup with sanding paper tiles, of different grit sizes stuck.

Concept 5

Building upon concept 4, this concept (see figure 1) is inspired by abacus and MIDI sound mixers. It would feature button sliders that move as per the code hierarchies. Similar to the previous concept, a firmware would interface the device with the computer. Musical strings would be used to differentiate between different indentation levels, such that when a blind user would move his fingers across these strings, it would give different notes across different strings. These strings were inspired from guitar strings and frets.

A foam mock up was made (see figure 1), fitted with sliders underneath using

popsicle sticks. For evaluation tests (see figure 1), the positions of each slider buttons were preset by pulling and pushing these popsicle sticks. Then the participant could listen to pre-recorded synthesized voices as he/she pushed buttons corresponding to each of the buttons. Non-visual Desktop Access (NVDA) software was used to synthesize the voices and were recorded into a SD card. An arduino MP3 shield was used with arduino to playback the respective voices as and when the buttons were pressed.

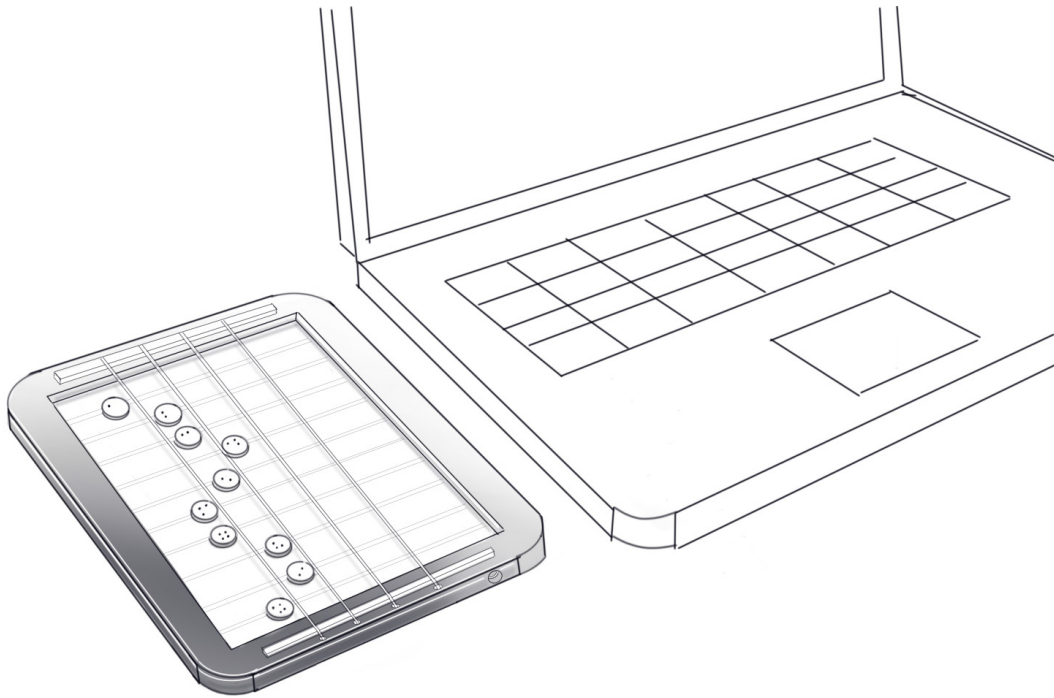


Figure 51 Rows of buttons sliders that would move according to code hierarchies

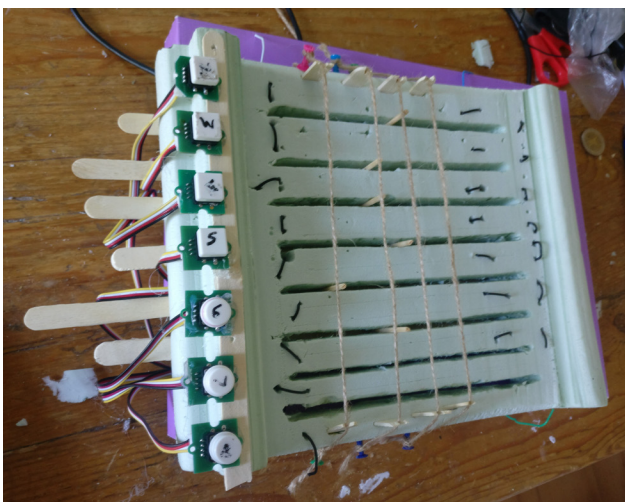


Figure 52 Foam mockup with buttons for playback of pre-recorded sounds

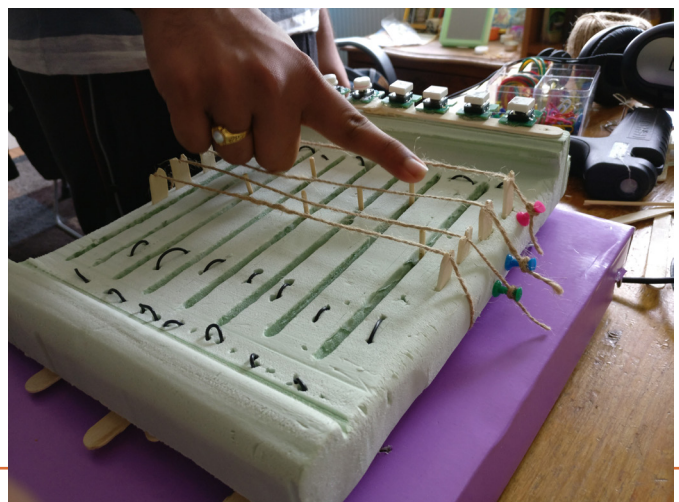


Figure 53 A participant interacting with mockup

3.4 Concept Evaluation

Method

The concepts were evaluated using the Weighted Objectives method (Rozenburg, 1995). The participants were blindfolded (with a sleeping mask) TU Delft students. First, they were blindfolded and were given the prototype to touch and get a feel of it. The final application nor the concept was explained to them, to reduce bias. Though they cannot replace the actual blind user entirely. For reasons of practicality and lack of access to actual end-users, students were blindfolded for the tests.

| Comparison Criteria | Weight | Concept 1 | | Concept 2 | | Concept 3 | | Concept 4 | | Concept 5 | |
|-------------------------------------------|--------|-----------|--------------|-----------|--------------|-----------|--------------|-----------|--------------|-----------|--------------|
| | | Score | Weight Score | Score | Weight Score | Score | Weight Score | Score | Weight Score | Score | Weight Score |
| Ability to skim code | 5 | 5 | 25 | 2 | 10 | 4 | 20 | 3 | 15 | 5 | 25 |
| Minimal Risk in operation | 5 | 4 | 20 | 2 | 10 | 3 | 15 | 1 | 5 | 3 | 15 |
| Technologically Feasible | 3 | 4 | 12 | 4 | 12 | 4 | 12 | 4 | 12 | 5 | 15 |
| Ability to navigate to specific locations | 4 | 1 | 4 | 3 | 12 | 4 | 16 | 3 | 12 | 5 | 20 |
| Failure proof (Robust Design) | 4 | 3 | 12 | 2 | 8 | 2 | 8 | 2 | 8 | 4 | 16 |
| Cost Effective | 1 | 4 | 4 | 1 | 1 | 3 | 3 | 4 | 4 | 2 | 2 |
| | 22 | 77 | | 53 | | 74 | | 56 | | 93 | |

Table 2 Weighted Objectives evaluation

Results

Concept 5 was chosen based on the evaluation table above.

Appendix 4

4.1 Aesthetic Form Exploration

The button concepts were derived from an analysis of existing designs (see figure 54) used in products with an approach to accessibility.

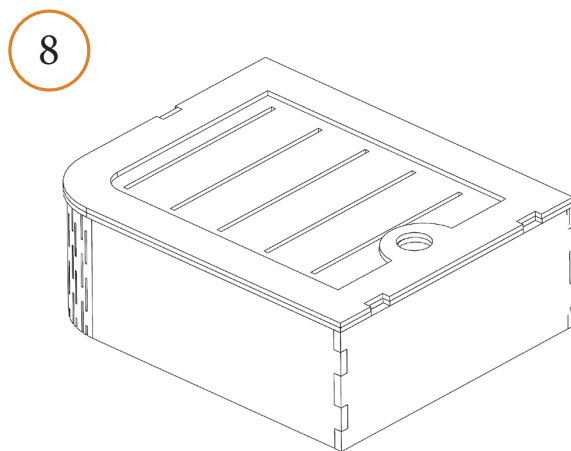
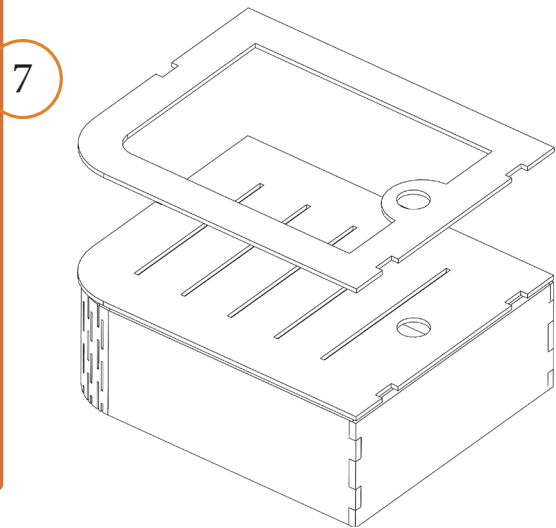
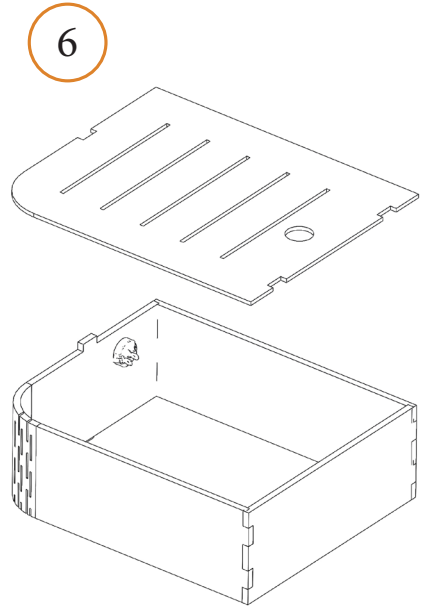
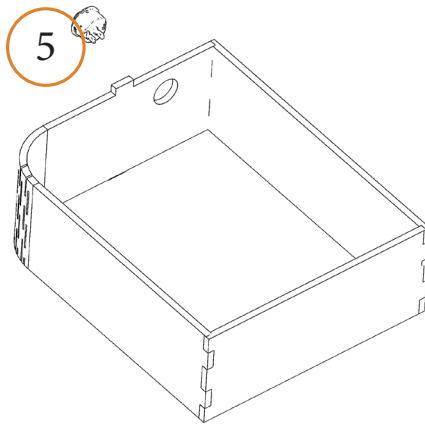
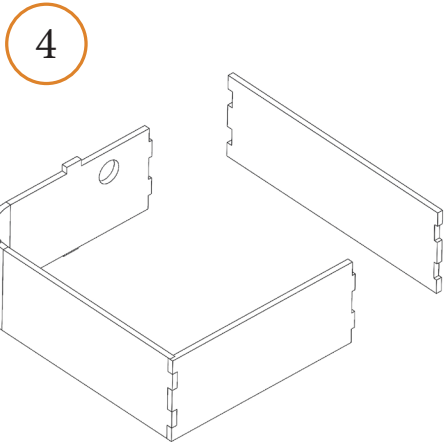
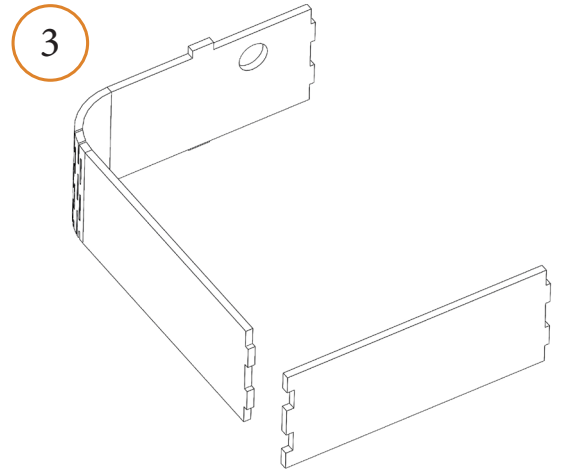
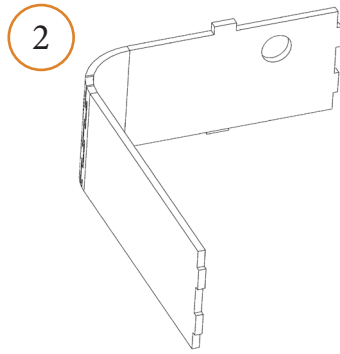
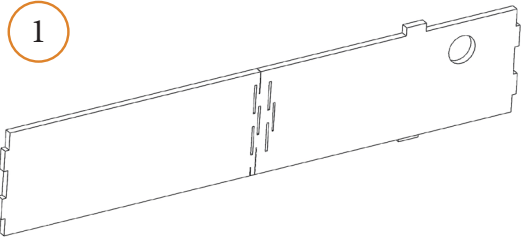


Figure 54 Aesthetic analysis of buttons from existing products having accessibility features

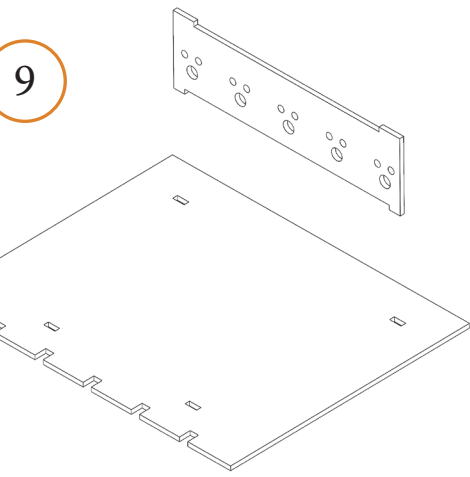
4.2 DIY Assembly

To facilitate assembly by secondary customers, a DIY version of the housing was developed. Lasercutting was chosen to be optimum manufacturing process, given its widespread technology availability. Self mating parts were designed, that could be cut out of a single sheet (600mm x 800mm) of poplar wood or Medium Density Fibre (MDF) board. The part design followed the following Design for Assembly principles (DFA), to reduce errors during self assembly by low skilled people. The assembly instructions can be found in Appendix 5.6

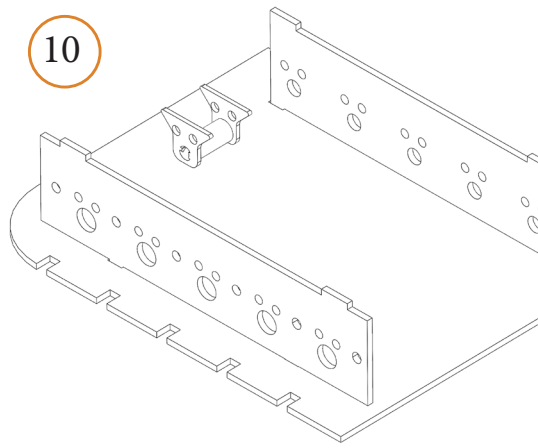
4.3 DIY Assembly Guide



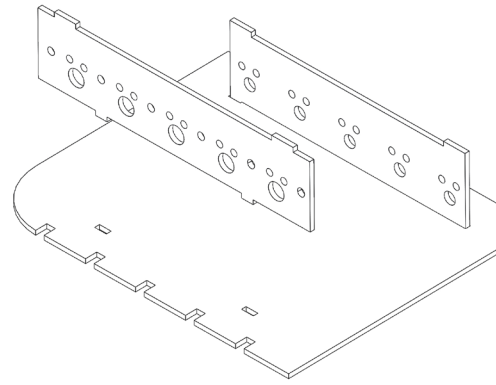
9



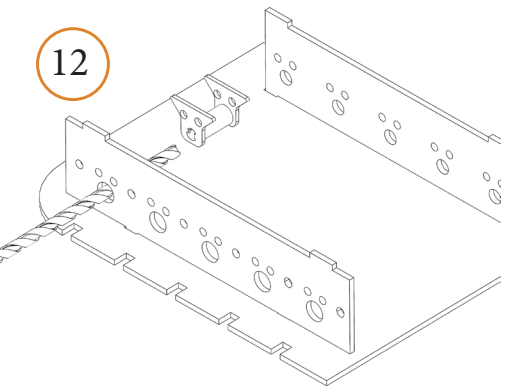
10



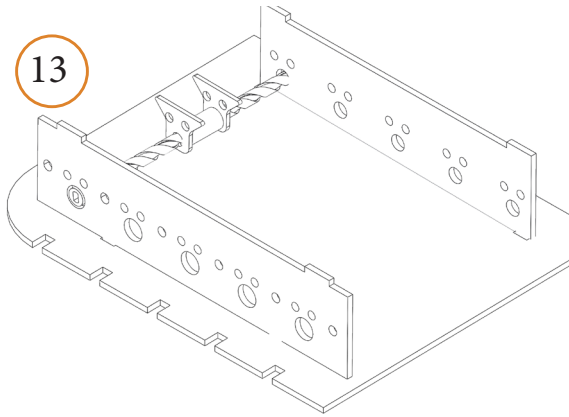
11



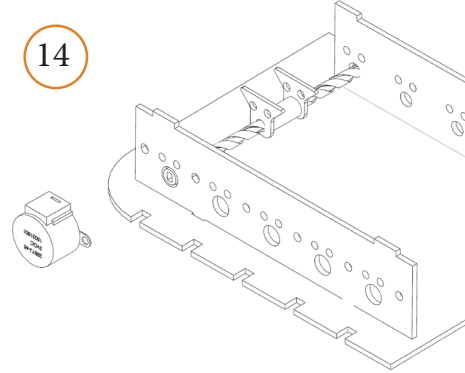
12



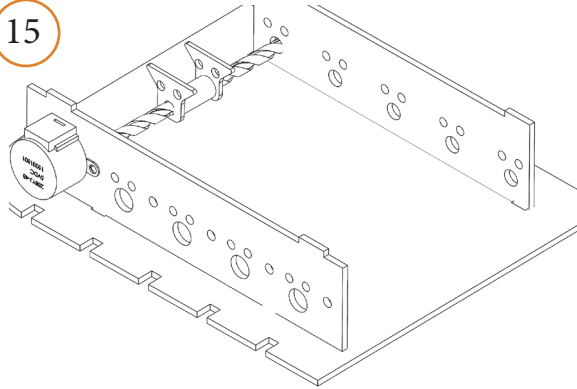
13



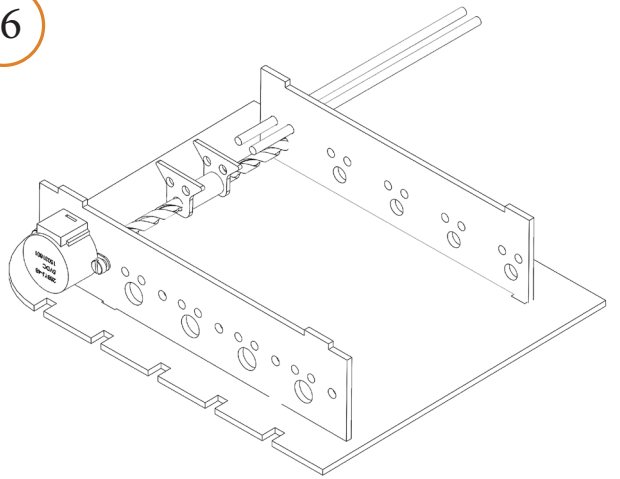
14



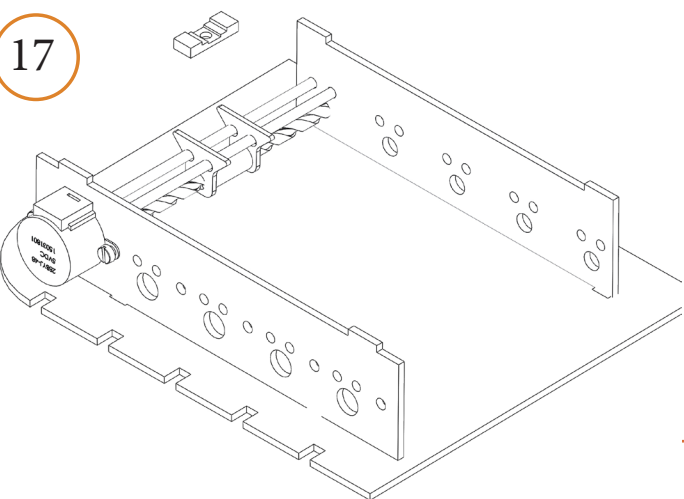
15



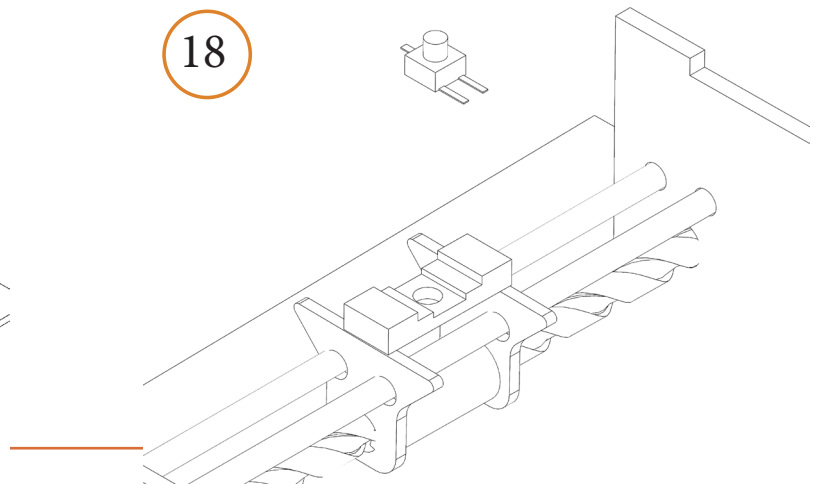
16

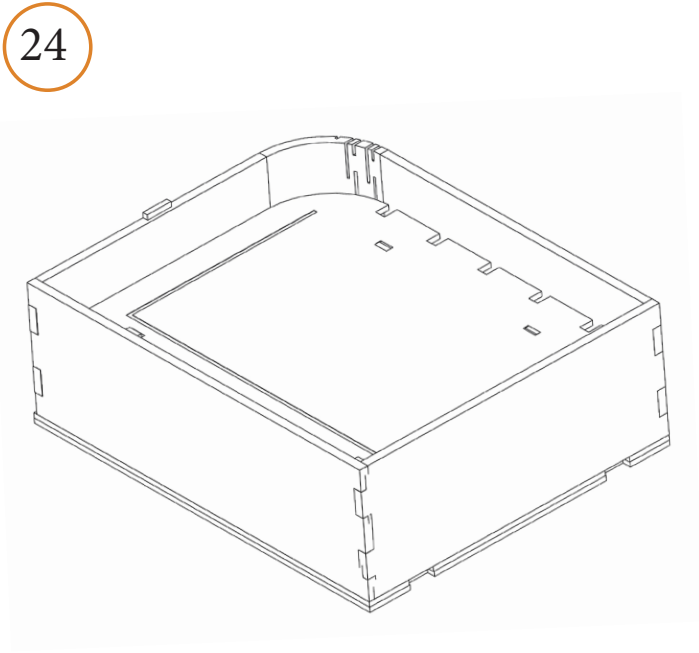
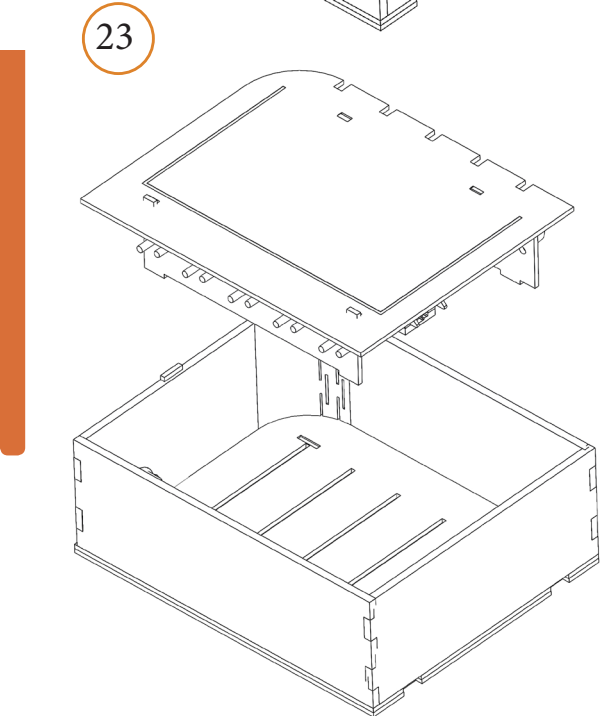
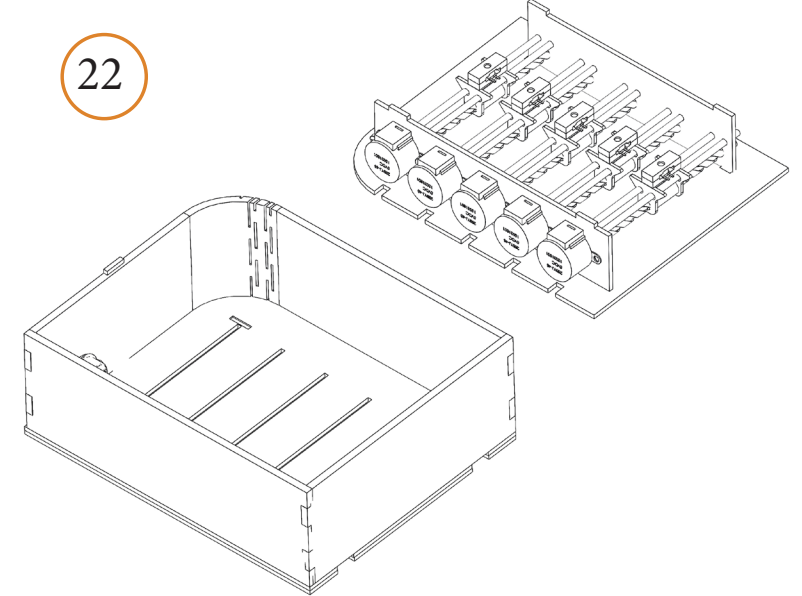
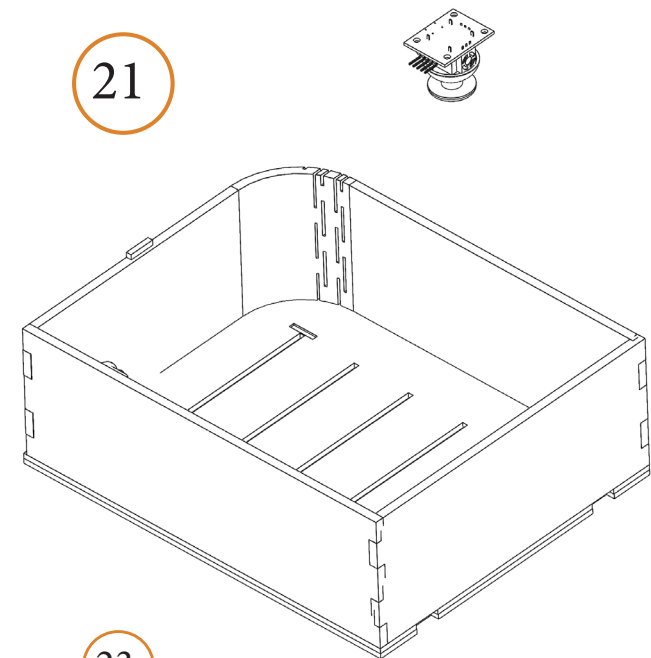
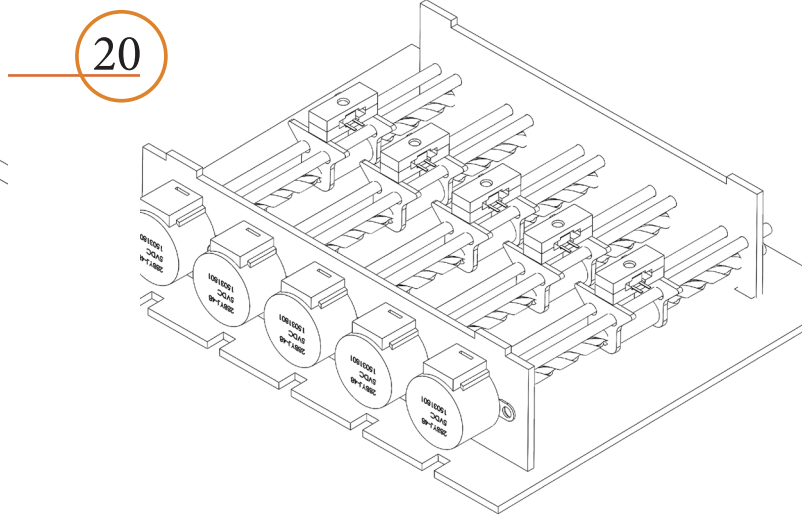
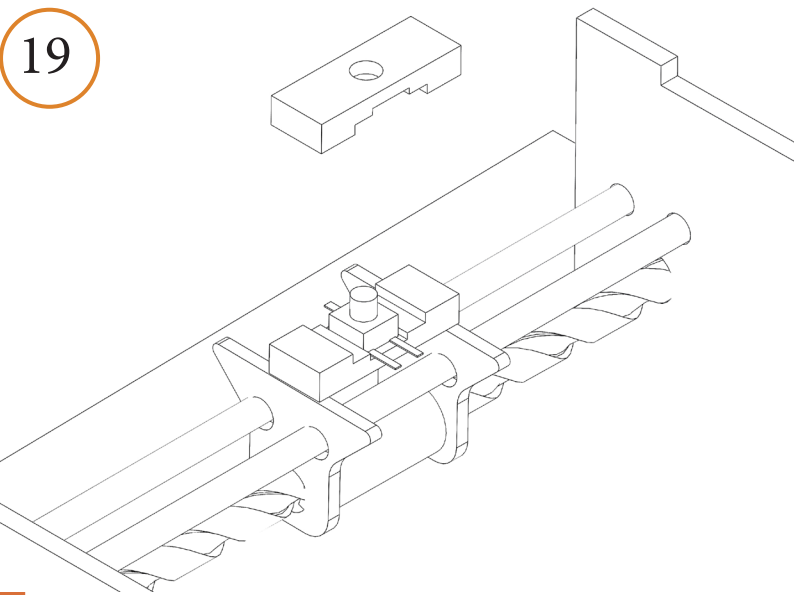


17

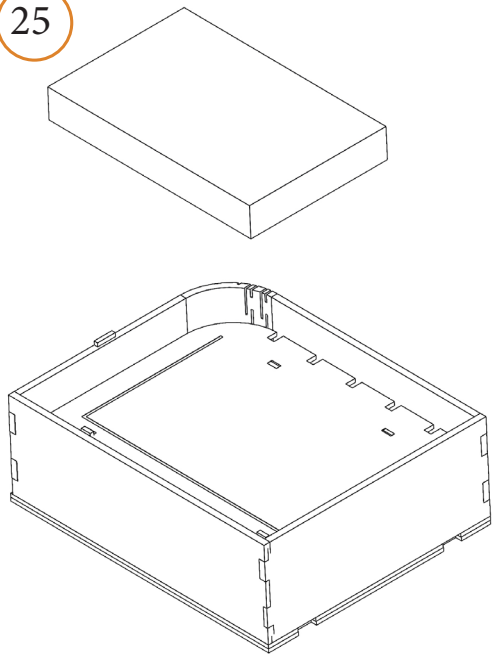


18

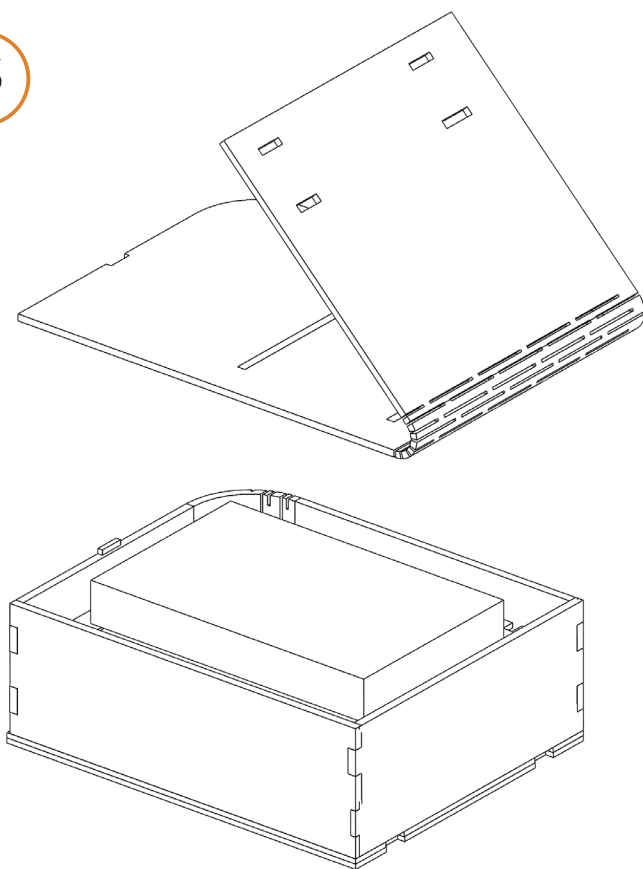




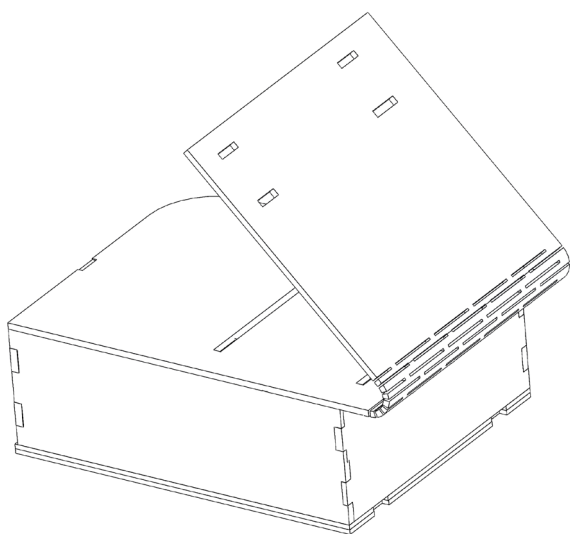
25



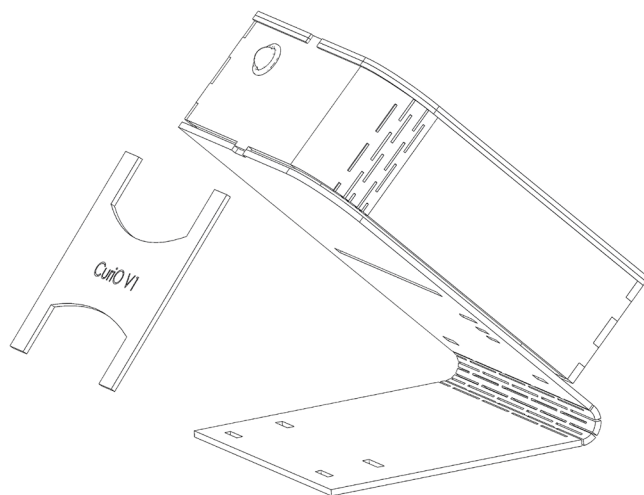
26



27



28



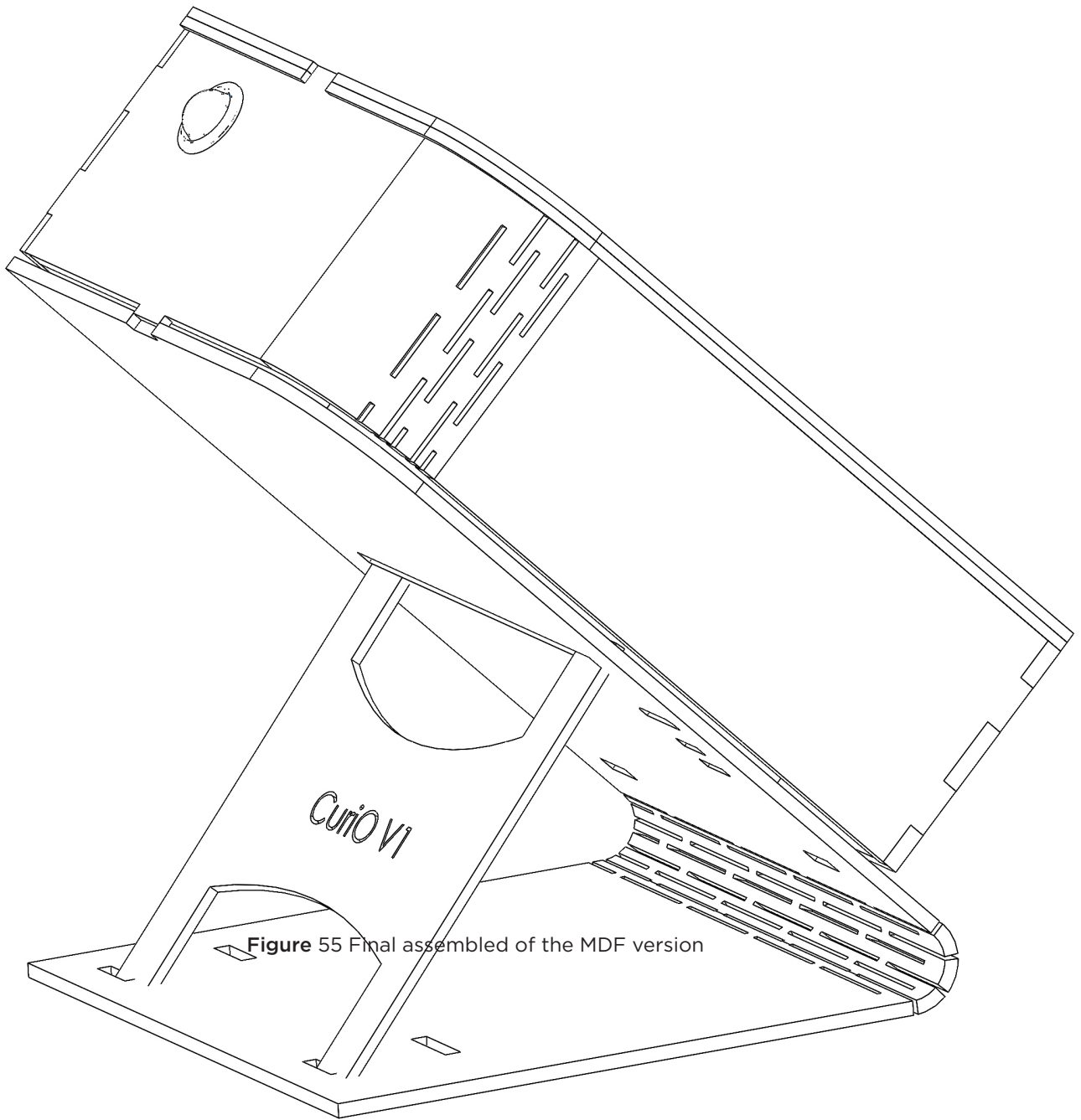




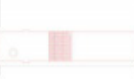









Figure 55 Final assembled of the MDF version

4.4 Bill of Materials - MDF

| Category | Part Name | Specifications | Quantity | Image | Cost (based on order links) | Total Cost | Order link | Reference Code |
|-------------|-------------------------|---------------------------------------------------------------------|----------|-------------------------------------------------------------------------------------|-----------------------------|------------|-----------------------------------------------------------|----------------|
| Laser cut | Startup costs | | - | | 3,63 | 3,63 | www.laserbeest.nl | |
| | Work Window layer | 3mm MDF | 1 |  | 19,92 | 19,92 | | |
| | Slot layer | 3mm MDF | 1 |  | 0 | 0 | | |
| | Mechanism support Left | 3mm MDF | 1 |  | | | | |
| | Mechanism support Right | 3mm MDF | 1 |  | | | | |
| | Long side | 5mm Poplar wood | 1 |  | 0 | 0 | | |
| | Bottom Side | 5mm Poplar wood | 1 |  | 0 | 0 | | |
| | Top side | 5mm Poplar wood | 1 |  | 0 | 0 | | |
| | Mechanism layer | 3mm MDF | 1 |  | 0 | 0 | | |
| | Bottom support layer | 3mm MDF | 1 |  | 0 | 0 | | |
| | Wood Screws | 2mm diameter, 12mm length | 10 |  | 0,1 | 1 | amazon.de/800-3 | 0 |
| 3D printing | Lead screws | Length 125mm Pitch 20mm Nylon filament SLS Printing method | 5 |  | 8,35 | 41,75 | https://www.3dhu.com | 1 |
| | Travelling nut | Pitch 20mm Nylon filament SLS Printing method | 5 |  | 5,59 | 27,95 | https://www.3dhu.com | 2 |

| | | | | | | | | |
|-------------|---------------------------|----------------|---|-------------------------------------------------------------------------------------|-------|---------------|-------------------------------------------------------|----|
| Electronics | Arcade Button | 16mm | 1 |  | 1,25 | 1,25 | https://www.kiwi- | 1 |
| | Audio Jack male to female | 35mm | 1 |  | 4,07 | 4,07 | https://nl.rs-onlin | 2 |
| | Arduino Leonardo | - | 1 |  | 21,95 | 21,95 | https://www.kiwi- | 3 |
| | Thumb Joystick | Analog 2 axis | 1 |  | 6,95 | 6,95 | https://www.kiwi- | 4 |
| | Stepper motors | 28 BYJ-48 | 5 |  | 5,95 | 29,75 | | 5 |
| | Motor Driver IC | ULN2003a | 5 |  | 0,54 | 2,7 | https://nl.rs-onlin | 6 |
| | IC Socket | 16 way DIL | 5 |  | | | https://nl.rs-onlin | 7 |
| | Power Bank | 5000 mAh, 2.1A | 1 |  | 24,5 | 24,5 | https://nl.rs-onlin | 8 |
| | Rocker Switch | MRS-101-9 | 1 |  | 0,75 | 0,75 | https://www.kiwi- | 9 |
| | Push Buttons | Round, 12mm | 6 |  | 0,46 | 4,63 | https://www.kiwi- | 10 |
| | SD card | 8GB | 1 |  | 7,95 | 7,95 | https://www.kiwi- | 11 |
| | Rpi power supply | 5V DC, 2.5A | 1 |  | 8,41 | 8,41 | https://nl.rs-onlin | 12 |
| | | | | Total | | 207.16 | | |

Table 3 Bill of materials

4.5 System Architecture

The electronics control system uses a Raspberry pi 3B+ and Arduino Leonardo, with ULN2003a stepper drivers. 28BYJ-28 stepper motors were considered for its ease of spare part availability and the maximum torque. It has a holding torque of 34.3mN.m, which is low enough to cause any physical injury to the blind user, even in the case of product misuse. A Leonardo version of Arduino was selected as only chips with 32u4 architecture can send ASCII keystrokes through their native USB port.

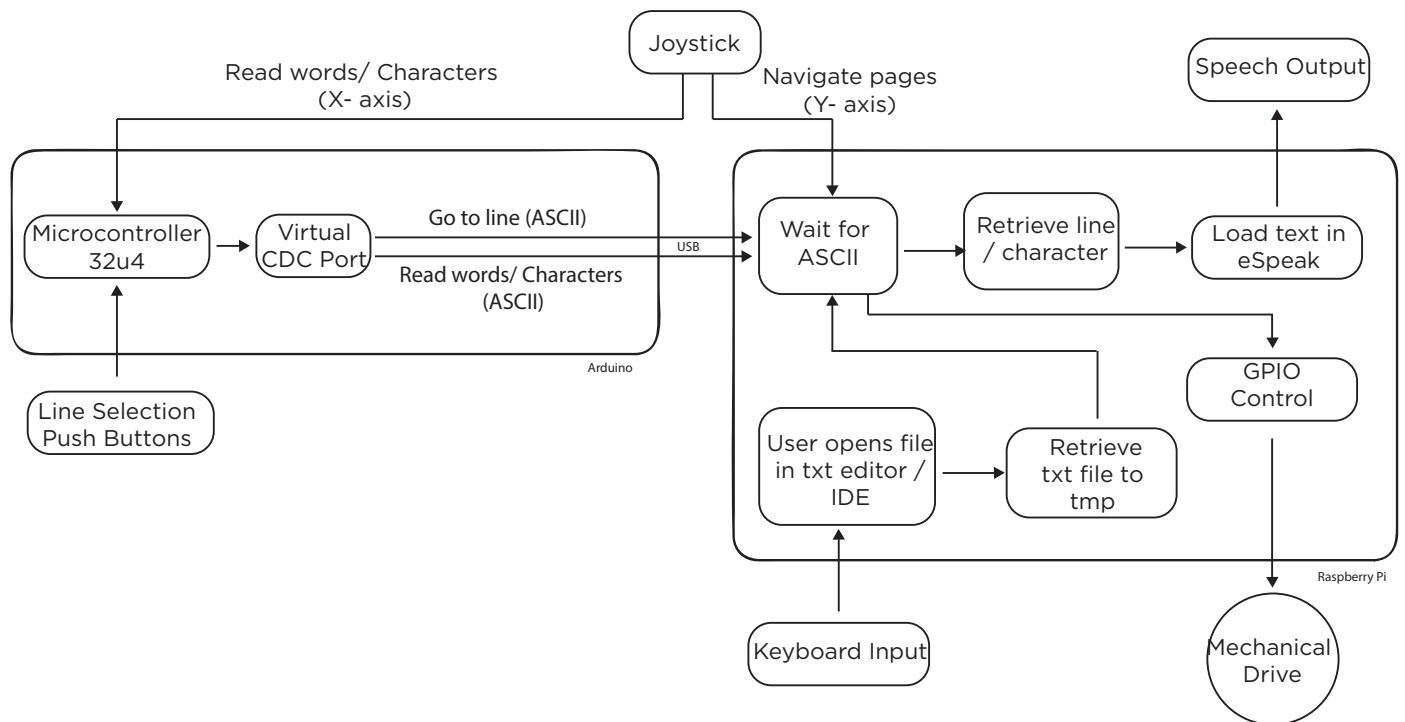


Figure 56 System Architecture showing the fore-end and backend processes

4.6 Python code for stepper control

```
import RPi.GPIO as GPIO
import time
```

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
```

```
motor_control_pins = [
    ([11, 7, 15, 13], "Alice"),
    ([16, 18, 22, 32], "Bob"),
    ([33, 31, 29, 35], "Charlie"),
    ([36, 37, 38, 40], "Derek"),
    ([23, 21, 19, 10], "Eddie")]
```

```
for motor in motor_control_pins:
    for pin in motor[0]:
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, 0)
```

```
halfstep_seq = [
    [1, 0, 0, 0],
    [1, 1, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
    [0, 0, 1, 0],
    [0, 0, 1, 1],
    [0, 0, 0, 1],
    [1, 0, 0, 1]]
```

```
def static_vars(**kwargs):
    """
```

```
    Custom decorator to allow declaration of static variables like
this:\n
```

```
    @static_var(variable = 0)\n    def foo(parameters):\n        foo.variable += 1
    """
```

```
def decorate(func):
    for k in kwargs:
        setattr(func, k, kwargs[k])
    return func
return decorate
```

```
# end def def reverse_motor(motor):
    return [motor[2], motor[1], motor[0], motor[3]]
# end def
```

```
def set_5_indents(levels):
    steps_per_level = 512
    motor_steps = [0] * 5
```

```
    for i in range(len(levels)):
        motor_steps[i] = levels[i] * steps_per_level
```

```
    for current_full_step in range(max(levels) *
steps_per_level):
        for halfstep in range(8):
            i = 0
            for motor in motor_control_pins:
                if (i + 1) > len(levels):
                    break
```

```
                print(i)
                for pin in range(4):
                    if (current_full_step < motor_
steps[i]):
                        if (levels[i] < 0):
                            GPIO.output(reverse_mo-
tor(motor[0])[
                                pin], halfstep_seq[half-
step][pin])
                        elif (levels[i] > 0):
                            GPIO.output(motor[0][pin],
                                halfstep_seq[halfstep]
[pin])
                            i = i + 1
```

```
                time.sleep(0.001)
            # end def
```

```
def load_indents_from_file(filename):
    """
```

```
    Load all indent information from the file and
returns a list of
indentations
    """
    filehandle = open(filename, 'r')
    ind = []
```

```
    while True:
        line = filehandle.readline()
        if (not line):
            break
        else:
            # assuming 4 spaces per indent level
            ind.append((len(line) - len(line.lstrip()))
>> 2)
```

```

filehandle.close()
return ind
# end def

def get_5_indent_values(in_buffer, start_line):
    "Returns five lines from the input buffer"
    return in_buffer[start_line:start_line+5]
# end def

def get_5_indent_deltas(old, new):
    return [(new[i] - old[i]) for i in range(5) if (i < len(new))
and (i < len(old))]
# end def

@static_vars(page=[0] * 5)
@static_vars(previous_page=[0] * 5)
def update_interface(start_line, visible_lines):
    "Calculates the new state of the interface"
    print("Showing lines: %d - %d" %
        (start_line + 1, start_line + visible_lines))
    update_interface.page = get_5_indent_values(indents,
active_line)

    print("prev\tnext\t\tdifference")
    for p1, p2 in zip(update_interface.previous_page, up-
date_interface.page):
        print(p1, p2, "|", p2 - p1, sep="\t")

    update_interface.previous_page = update_interface.page
# end def

indents = load_indents_from_file("mock_stepper_control.
py")

active_line = 0
display_lines = 5
total_lines = len(indents) + 1
userinput = ""

if total_lines <= 5:
    update_interface(0, total_lines)
    print("Showing lines: 1 - %d" % (total_lines))
else:
    while True:
        # User selected Next page
        if (userinput.upper() == "N") and ((active_line + 5)
<= total_lines):
            active_line = active_line + 5
            display_lines = 5

```

```

# User selected Previous page
elif (userinput.upper() == "P") and ((ac-
tive_line - 5) >= 0):
    active_line = active_line - 5
    display_lines = 5

# User selected Quit program
elif userinput.upper() == "Q":
    break

# User entered invalid character
else:
    pass

if active_line + display_lines > total_
lines:
    display_lines = total_lines - active_line

    update_interface(active_line, display_
lines)
    userinput = input("(N)ext | (P)revious |
(Q)uit > ")
# end while

GPIO.cleanup()

```

Leonardo code to send ASCII values

```

// Declare the pins for the Button and the
LED<br>int buttonPin = 12;
#include <Keyboard.h>

int button1 = 12;
int button2 = 11;
int button3 = 10;
int button4 = 9;
int button5 = 8;

int VRx = A2;
int xPosition = 0;
int mapX = 0;

void setup() {
    // Define pin #12 as input and activate the
internal pull-up resistor

```

```

Serial.begin(9600);
pinMode(VRx, INPUT);

pinMode(button1, INPUT_PULLUP);
pinMode(button2, INPUT_PULLUP);
pinMode(button3, INPUT_PULLUP);
pinMode(button4, INPUT_PULLUP);
pinMode(button5, INPUT_PULLUP);

Keyboard.begin();
}
void loop(){

  xPos = analogRead(VRx);
  //mapX = map(xPos, 0, 1023, -512, 512);

//Read words Backward
  if (xPos < 200) {
    Keyboard.press(130);
    Keyboard.write(98);
    Serial.println("Words reading backward");
  }

//Read words Forward
  else if (xPos > 337){
    Keyboard.press(130);
    Keyboard.write(102);
    Serial.println("Words reading Forward");
  }

  delay(1000);
  Keyboard.releaseAll();

// Read the value of the input. It can either be 1 or
0
  int buttonValue1 = digitalRead(button1);
  int buttonValue2 = digitalRead(button2);
  int buttonValue3 = digitalRead(button3);
  int buttonValue4 = digitalRead(button4);
  int buttonValue5 = digitalRead(button5);

//For multiple key presses use Keyboard.press()

  if (buttonValue1 == LOW){
    // If button pushed, turn LED on
    //Serial.println("Button 1 is pressed");

//M-g
    Keyboard.press(130);
  }

  Keyboard.press(103);
  //delay(50);

  Keyboard.press(130);
  Keyboard.press(103);
  //delay(50);
  Keyboard.press(103);
  Keyboard.releaseAll();
  delay(1000);

//line number
  Keyboard.write(49);
  Keyboard.release(49);
  delay(200);

//Return
  Keyboard.write(176);
  Keyboard.releaseAll();
}

  if (buttonValue2 == LOW){
    // If button pushed, turn LED on
    //Serial.println("Button 1 is pressed");

//M-g
    Keyboard.press(130);
    Keyboard.press(103);
    //delay(50);
    Keyboard.press(103);
    Keyboard.releaseAll();
    delay(50);

//M-g
    Keyboard.press(130);
    Keyboard.press(103);
    //delay(50);
    Keyboard.press(103);
    Keyboard.releaseAll();
    delay(1000);

//line number
    Keyboard.write(50);
    Keyboard.release(50);
    delay(200);
  }
}

```

Arduino Mega Code used for quick tests

```
#include <Stepper.h>
#include <Keyboard.h>

// steps value is 360 / degree angle of motor
#define STEPS 200

// create a stepper object on pins 4, 5, 6 and 7
Stepper stepper1(STEPS, 17, 15, 16, 14);
Stepper stepper2(STEPS, 18, 20, 19, 21);
Stepper stepper3(STEPS, 7, 5, 6, 4);
Stepper stepper4(STEPS, 11, 9, 10, 8);
Stepper stepper5(STEPS, 0, 2, 1, 3);

//initialise button - pins configuration
int button1 = 11;
int button2 = 12;
int button3 = 13;
int button4 = 14;
int button5 = 15;

int VRx = A1;
int VRy = A3;

int xPosition = 0;
int yPosition = 0;
int mapX = 0;
int mapY = 0;

//variables to denote live page number tracking
int pg = 1;

void StepperReset()
{
  int i;
  int reset=0;

  Serial.println("Resetting to 0");
  stepper1.setSpeed(120);
  stepper1.step(-1200);

  stepper2.setSpeed(120);
  stepper2.step(-1200);

  stepper3.setSpeed(120);
  stepper3.step(-1200);
```

```
stepper4.setSpeed(120);
stepper4.step(-1200);
```

```
  stepper5.setSpeed(120);
  stepper5.step(-1200);
  delay (1000);
}
```

```
int StepperPg1()
```

```
{
  Serial.println("Page 1");
  stepper1.setSpeed(120);
  stepper1.step(-240);

  stepper2.setSpeed(120);
  stepper2.step(640);

  stepper3.setSpeed(120);
  stepper3.step(600);

  stepper4.setSpeed(120);
  stepper4.step(640);

  stepper5.setSpeed(120);
  stepper5.step(420);

  return 1;
  //delay (100);
}
```

```
int StepperPg2()
```

```
{
  Serial.println("Page 2");
  stepper1.setSpeed(120);
  stepper1.step(640);

  stepper2.setSpeed(120);
  stepper2.step(800);

  stepper3.setSpeed(120);
  stepper3.step(640);

  stepper4.setSpeed(120);
  stepper4.step(-400);

  stepper5.setSpeed(120);
  stepper5.step(640);
  return 2;
```

```
//delay (100);
}

void setup()
{
  Serial.begin(9600);
  //Keyboard.begin();
  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  StepperReset();
}

void loop()

{

  xPosition = analogRead(VRx);
  yPosition = analogRead(VRy);
  //mapX = map(xPosition, 0, 1023, -512,
512);
  //mapY = map(yPosition, 0, 1023, 0,
1023);

  if (yPosition < 201)
  {
    //Call for StepperPg1 function
    StepperReset();
    pg = StepperPg1();
  }
  else if (yPosition > 201)
  {
    //Call for StepperPg2 function
    StepperReset();
    pg = StepperPg2();
  }
  delay(1000);
}
```


Symbol : Footprint Assignments

| | | |
|----|-----------|---------------------------------------------------------------------------------------|
| 1 | 5V1 - | Barrel_Jack : Connector_BarrelJack:BarrelJack_CUI_PJ-063AH_Horizontal |
| 2 | A1 - | Arduino Connector : Module:Arduino_UNO_R3 |
| 3 | Driver1 - | ULN2003A : Package_DIP:DIP-16_W7.62mm_LongPads |
| 4 | Driver2 - | ULN2003A : Package_DIP:DIP-16_W7.62mm_LongPads |
| 5 | Driver3 - | ULN2003A : Package_DIP:DIP-16_W7.62mm_LongPads |
| 6 | Driver4 - | ULN2003A : Package_DIP:DIP-16_W7.62mm_LongPads |
| 7 | Driver5 - | ULN2003A : Package_DIP:DIP-16_W7.62mm_LongPads |
| 8 | J1 - | JoystickConnector : Connector_PinSocket_1.00mm:PinSocket_1x05_P1.00mm_Vertical |
| 9 | M1 - | Stepper_Motor_unipolar_5pin : Connector_JST:JST_XH_S5B-XH-A-1_1x05_P2.50mm_Horizontal |
| 10 | M2 - | Stepper_Motor_unipolar_5pin : Connector_JST:JST_XH_S5B-XH-A-1_1x05_P2.50mm_Horizontal |
| 11 | M3 - | Stepper_Motor_unipolar_5pin : Connector_JST:JST_XH_S5B-XH-A-1_1x05_P2.50mm_Horizontal |
| 12 | M4 - | Stepper_Motor_unipolar_5pin : Connector_JST:JST_XH_S5B-XH-A-1_1x05_P2.50mm_Horizontal |
| 13 | M5 - | Stepper_Motor_unipolar_5pin : Connector_JST:JST_XH_S5B-XH-A-1_1x05_P2.50mm_Horizontal |
| 14 | R1 - | 10K : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P15.24mm_Horizontal |
| 15 | R2 - | 10K : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P15.24mm_Horizontal |
| 16 | R3 - | 10K : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P15.24mm_Horizontal |
| 17 | R4 - | 10K : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P15.24mm_Horizontal |
| 18 | R5 - | 10K : Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P15.24mm_Horizontal |
| 19 | SW1 - | SW_Push : Button_Switch_THT:SW_PUSH_6mm_H5mm |
| 20 | SW2 - | SW_Push : Button_Switch_THT:SW_PUSH_6mm_H5mm |
| 21 | SW3 - | SW_Push : Button_Switch_THT:SW_PUSH_6mm_H5mm |
| 22 | SW4 - | SW_Push : Button_Switch_THT:SW_PUSH_6mm_H5mm |
| 23 | SW5 - | SW_Push : Button_Switch_THT:SW_PUSH_6mm_H5mm |
| 24 | U1 - | RpiC : Connector_PinHeader_1.00mm:PinHeader_2x20_P1.00mm_Vertical |

Table 4 List of through hole components for PCB assembly, along with their designators

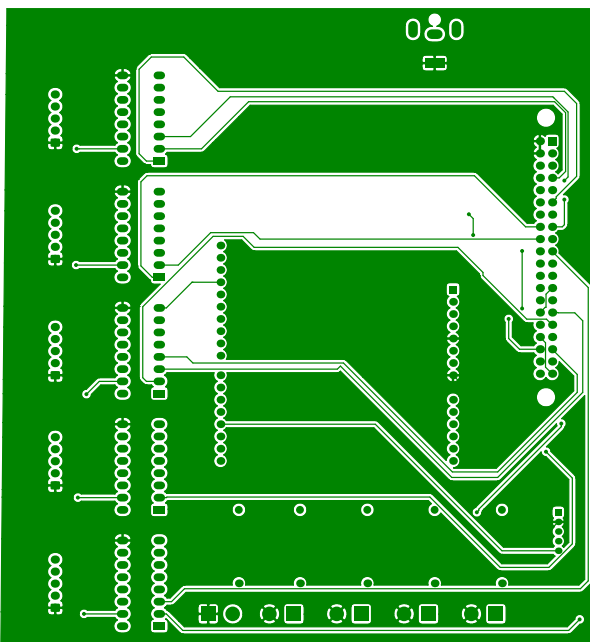


Figure 58 Top copper layer of the PCB layout

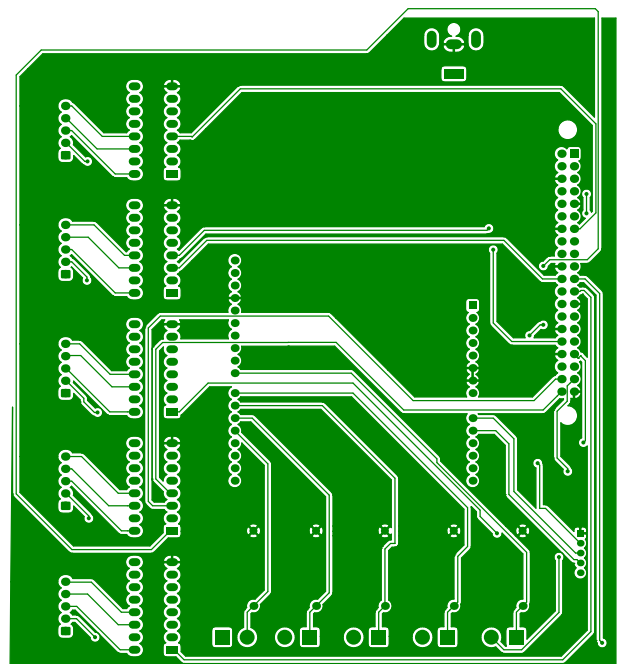


Figure 59 Bottom copper layer of the PCB layout

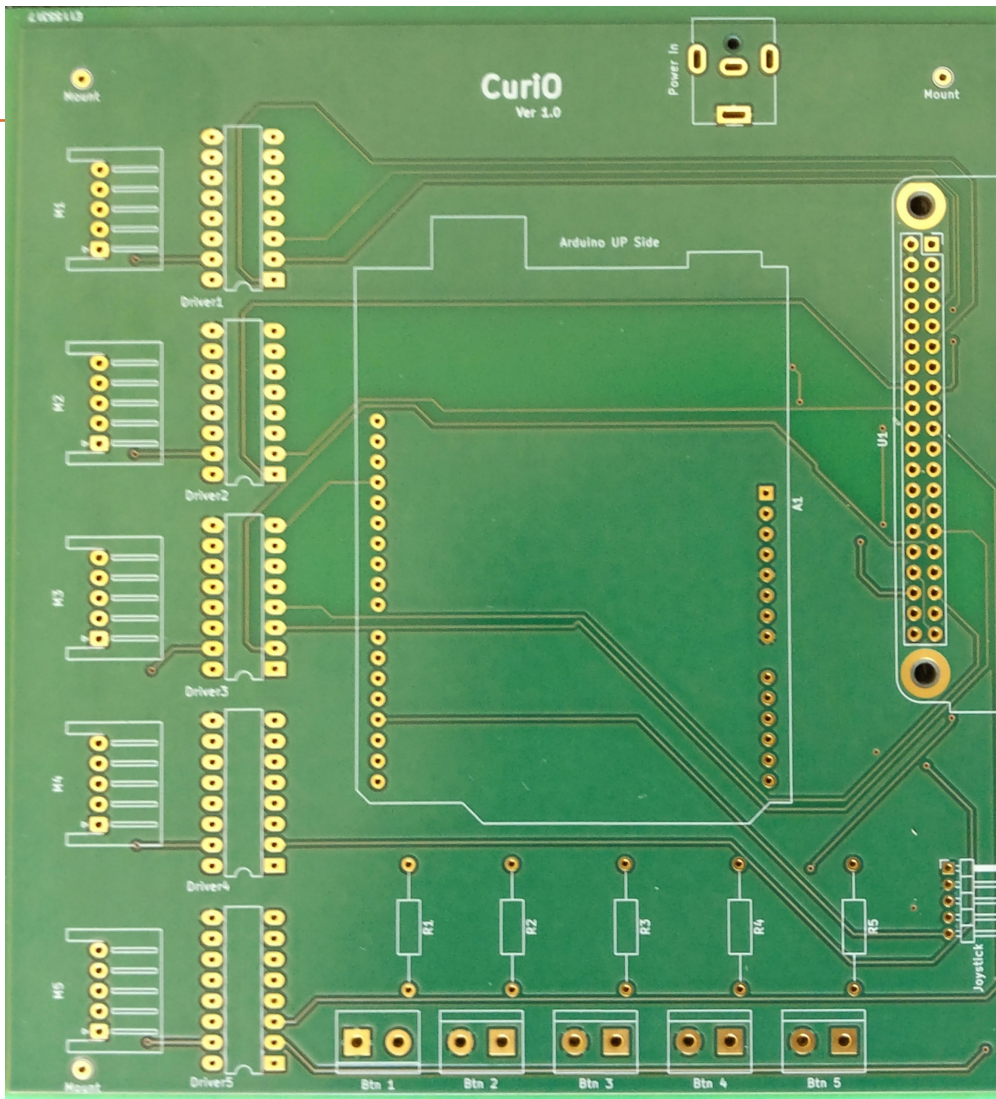


Figure 60 PCB fabricated using the Eurocircuits online service.

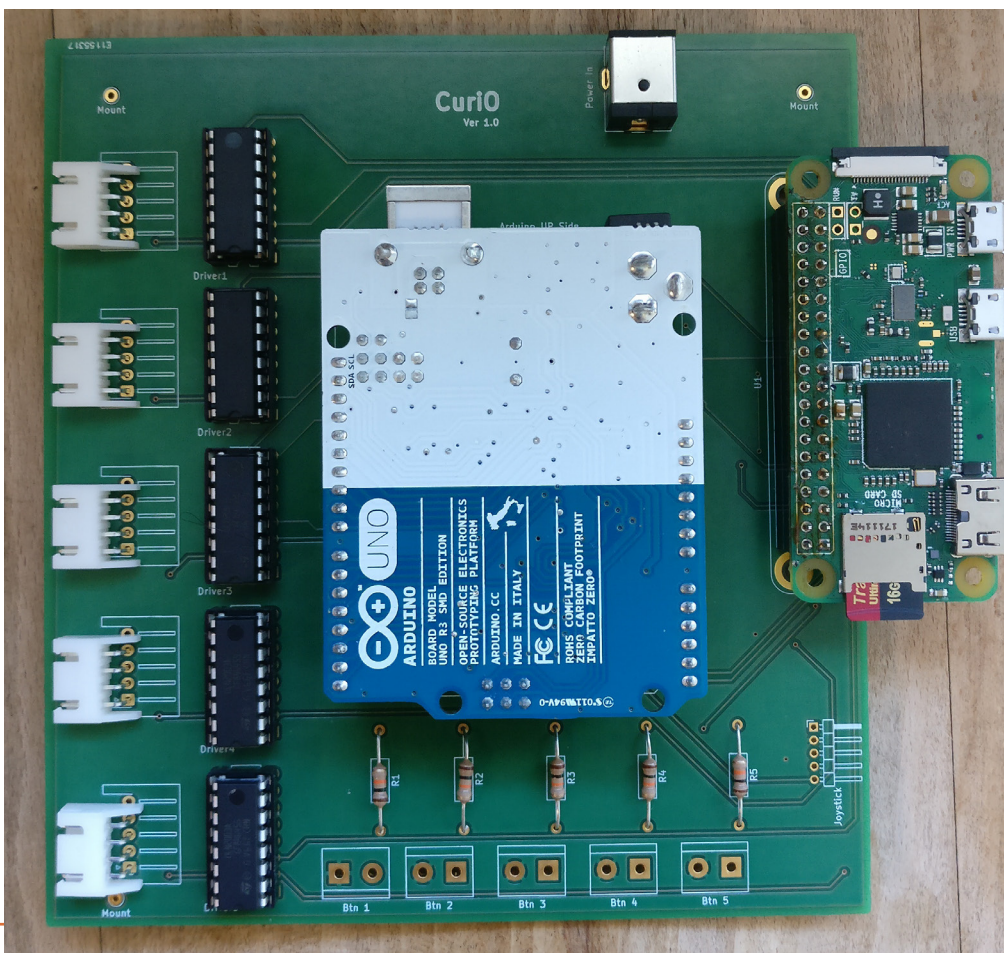


Figure 61 Assembled PCB ready for use

4.8 Product Interaction

The overall product experience vision was to give a sense of trust and reliability. At the same time, it increases productivity and playfulness of programming. The interaction design of the product was based on a combination of Nielsen Heuristics and Tognazzi's Principles of Interaction Design.

Comfort in familiarity, in designing motion-based interaction of the sliders and buttons. Moving beads to count in an abacus was an inspiration. During the shadowing session, it was observed how blind users interact with everyday things like smartphones, coffee machines, and kitchen knives. They use Allocentric processing of their haptic Space, which means they measure relative distances from a fixed point of reference to seek the position of any object.

Skeuomorphic Design: Users translate experiences in the physical world to cyber-physical interactions. The top panel and the surface etching to indicate different indentation levels were inspired by textured tiles used to indicate different rooms, as observed from the Visio school.

Use of metaphors in designing directional buttons or joystick movement. The vertical positioning of the joystick allowed to indicate pushing away from the user as the previous page and towards the user (+Y) as next page. Also, the right and left (X-axis) was metaphorically related to reading from left to right.

Use of consistency and standards: Placement of physical buttons and interface outlets, in the positions that follow the mental models of a user. So that they know where to find different points of interaction, to find them, such as the function button in the top left, like in game consoles and power switches in the top right, as in iPhones and other smartphones.

User Control: To help users leave an unwanted state immediately without facing

a long procedure or roundabout to do so.

Visibility of system status: To allow users to know if their interaction was successful through vibration based feedback. This allows them to feel in control and translate into better decision making and create trust. The vibration of the mechanical drive served as the feedback that the system was in operation.

Recognition, not Recall, in the type of information being provided as feedback (tactile and audio) to users. Tactile feedback provides quick recognition of how the code is structured per page or per block, while information is recalled while tapping on the button.

Recognition is easier because it involves less number of cues, and the user does not have to remember previous lines used interactions. He or she can quickly recognize the position of the slider and the relative information readout.

Discoverability of features: If a user cannot find an interaction feature of the product, it equals to him/her that it does not exist. Use of perceptual cues: Dependable tactile landmarks that help users to navigate around the product. For instance, blind users find their lockers easily by locating how many lockers is it away from the edge. This was taken as an inspiration to include a workspace with tactile edges so that the user knows where he/she can find the sliders and the joystick. Also, the buttons onto the sides were marked with tactile edges, to indicate it was a button.

Fitt's Law: Use of bigger buttons for frequently used interactions and smaller ones for less. The function button was designed to be bigger than other controls such as volume and joystick, more prominent (placing it at a distance from the surface) to indicate interaction priority. Also, according to Fitt's law, the

most accessed points by a user are the corners of the interface. Hence, to position essential elements of interaction at these positions.



Figure 62 Navigation



Figure 63 Skimming and Comprehension

5.2 System Usability

System usability System (SUS) (Brooke, 1986) was used to determine how interaction friendly the detailed concept is and how much it serves the design goal. The plan was to test the concept with one real blind person and two blindfolded participants.

Participants

Six TU Delft students from different study backgrounds volunteered in the test.

Method

The participants were blindfolded with a sleeping mask prior to the start of the test. A form prototype was placed in front of the participant, and they were asked to perceive the logic of the program.

Since by then a real time interface was not developed yet, the slider buttons were manually moved to positions as per the indentations in the code. Here, two programs were used- Fibonacci series and Singly Lists. The task for the participant was to interact with the product, comprehend the code logic, and draw the logic diagram after the task. Each row in the prototype was mapped onto sprites (see figure 64) in a scratch program (see figure1). When the participant (acted) tapped the slider buttons on the prototype to simulate real interaction, recorded screen reader sounds were played from a separate computer at the same instance, from the scratch program.



Figure 67 A blindfolded participant interacting with the prototype

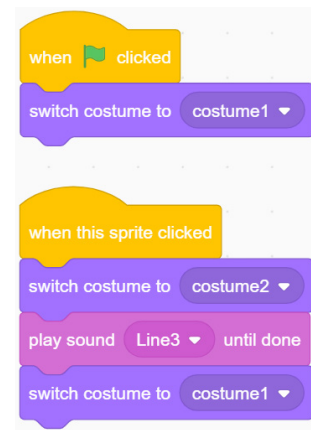


Figure 64 Scratch program to play pre-recorded sounds when sprites are clicked

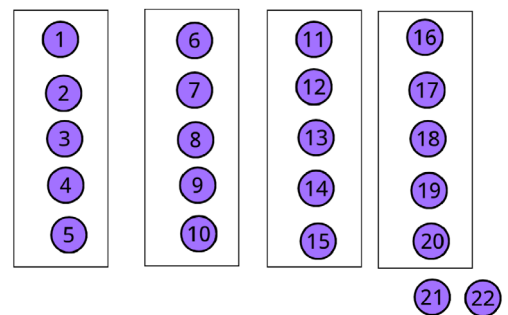


Figure 65 sprite buttons, each corresponding to each code row

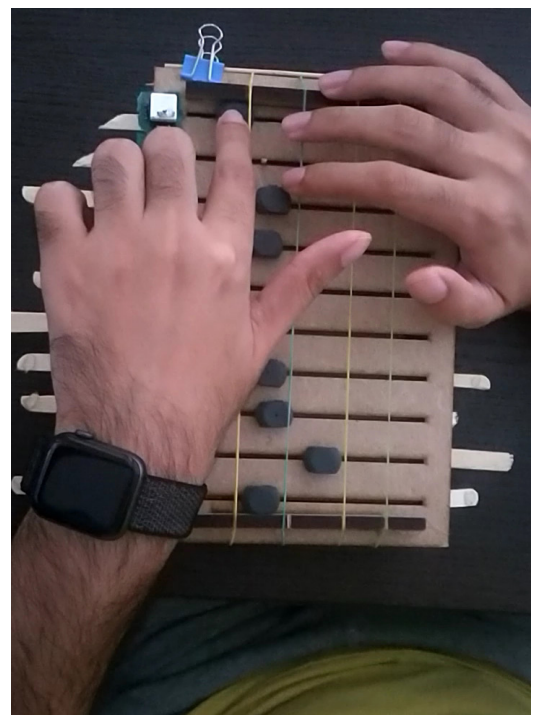


Figure 68 MDF variant prototype was used in the study

Evaluation

The questionnaire (see figure 1) consisted of six questions, three of which were positive, and the other three are negative. All of them have a 1-5 point for the judgment (from strongly disagree at 1 to strongly agree at 5). For questions 1, 3, 5, the score is subtracted by one. For questions 2, 4, 6, the score is subtracted from five statements. For the total score, the sum of scores is multiplied with the 4.5 to get an overall value for 100 points

System Usability Questionnaire

Researcher: KT Rajagopal

Date: 28/09/2019

I found it confusing to comprehend the hierarchy
Strongly Disagree Strongly Agree
 1 2 3 4 5

I found it difficult to understand the code logic with the system
Strongly Disagree Strongly Agree
 1 2 3 4 5

There was lot of discomfort using this
Strongly Disagree Strongly Agree
 1 2 3 4 5

I found this system unnecessarily complex
Strongly Disagree Strongly Agree
 1 2 3 4 5

I was able to learn to use this system quickly
Strongly Disagree Strongly Agree
 1 2 3 4 5

I could navigate to the required word quickly
Strongly Disagree Strongly Agree
 1 2 3 4 5

Table 5 System usability Questionnaire adapted from Brooke, 1986

| SUS Score | Adjective Rating |
|-----------|------------------|
| > 80.3 | Excellent |
| 68 - 80.3 | Good |
| 68 | Okay |
| 51- 68 | Poor |
| < 51 | Awful |

Table 6 System usability Scoring

5.4 Aesthetic Perception

Introduction

A buyer's perception of aesthetics was evaluated to determine if the product experience vision of reliability and trust is reflected in its form. Also, since the primary channel of procurement of assistive aids is through e-commerce (appendix 7.2), it is important what meaning is conveyed to a customer (not the end user) so that he/ she thinks its a good fit for the end user. The economic buyer or customer here can be a friend, family member or an special needs educator in a blind institution.

Participants

The test participants were students from TU Delft, from different disciplines in the age group of 21- 29 years old. It was ensured to have an equal proportion of male and female participants.

Method

They were shown a rendered image (see figure 1) of the product on a computer screen, and presented a questionnaire with 5 point Likert scale. The task was to look at the picture and grade the different visual criteria based on the first impression.

The participants were provided with very little information on what the product is about and its features, so as to reduce any bias during the evaluation. One blind programmer was also included as a participant (see figure 1), to test the haptic aesthetic of the product. The haptic aesthetic helped gain insights on whether a blind end user would prefer to buy the product after the experiencing its aesthetics and the emotions it conveys. After each grading, the participants were asked to explain the rationale behind their scores. Their answers were voice recorded for post-evaluation research.



Figure 69 Rendered picture of the final product, shown to participants during the Aesthetic perception test



Figure 70 3D printed form prototype presented to a blind programmer, for tactile perception of the product.

Evaluation

| | 1 | 2 | 3 | 4 | 5 |
|----------------|---|---|---|---|---|
| Robust | | | | | |
| Playful | | | | | |
| Intuitive | | | | | |
| Ergonomic | | | | | |
| Reliable | | | | | |
| Safe to use | | | | | |
| Gender Neutral | | | | | |

Table 7 Evaluation table used to evaluate the aesthetic perception

Results

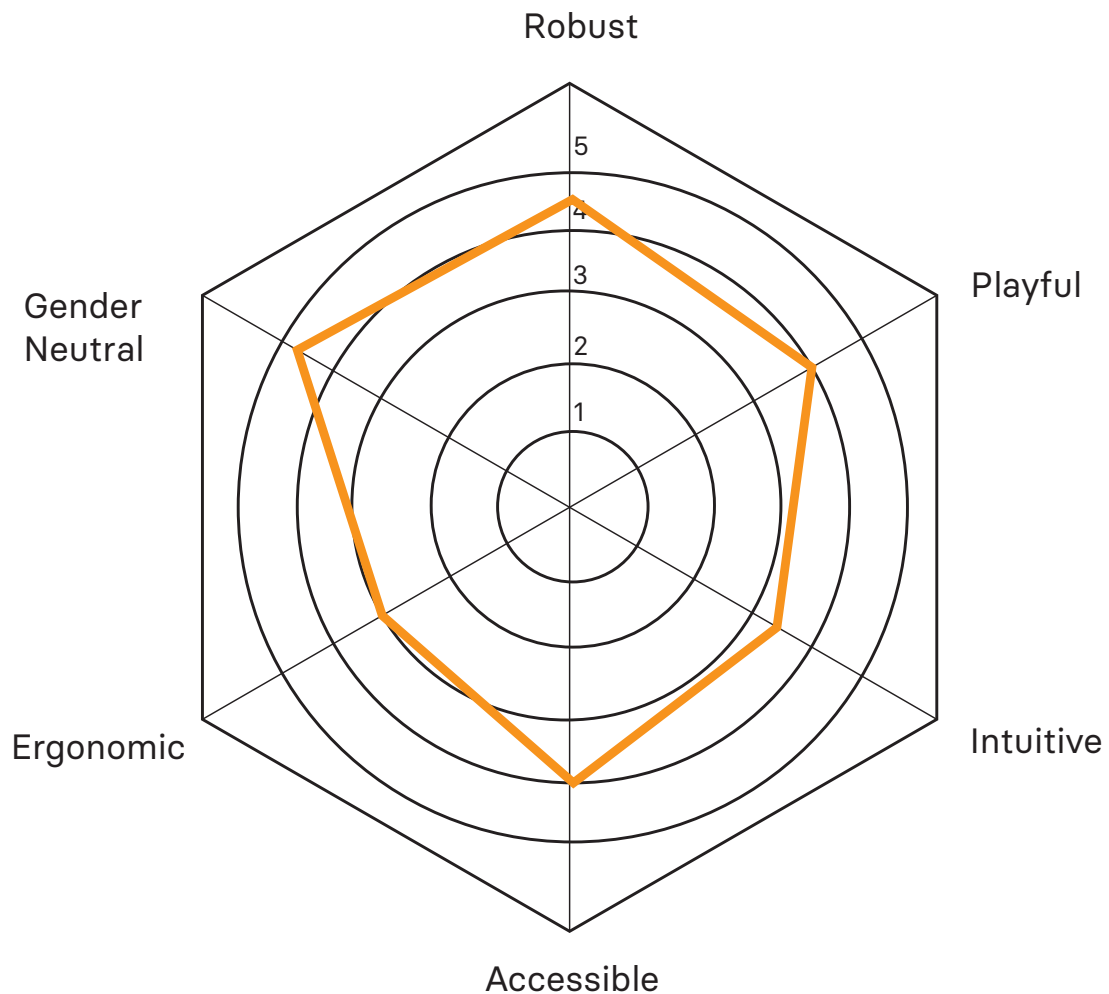


Figure 71 Radar chart showing the average score for each aesthetic criteria

Appendix 6

6.1 Assumptions and Validations

Assumptions in terms of perceptions, market trends and general preferences, according to their impact to the business model were mapped. They were validated through the different stages of the project, through user interviews, and reviews from blind professional programmers

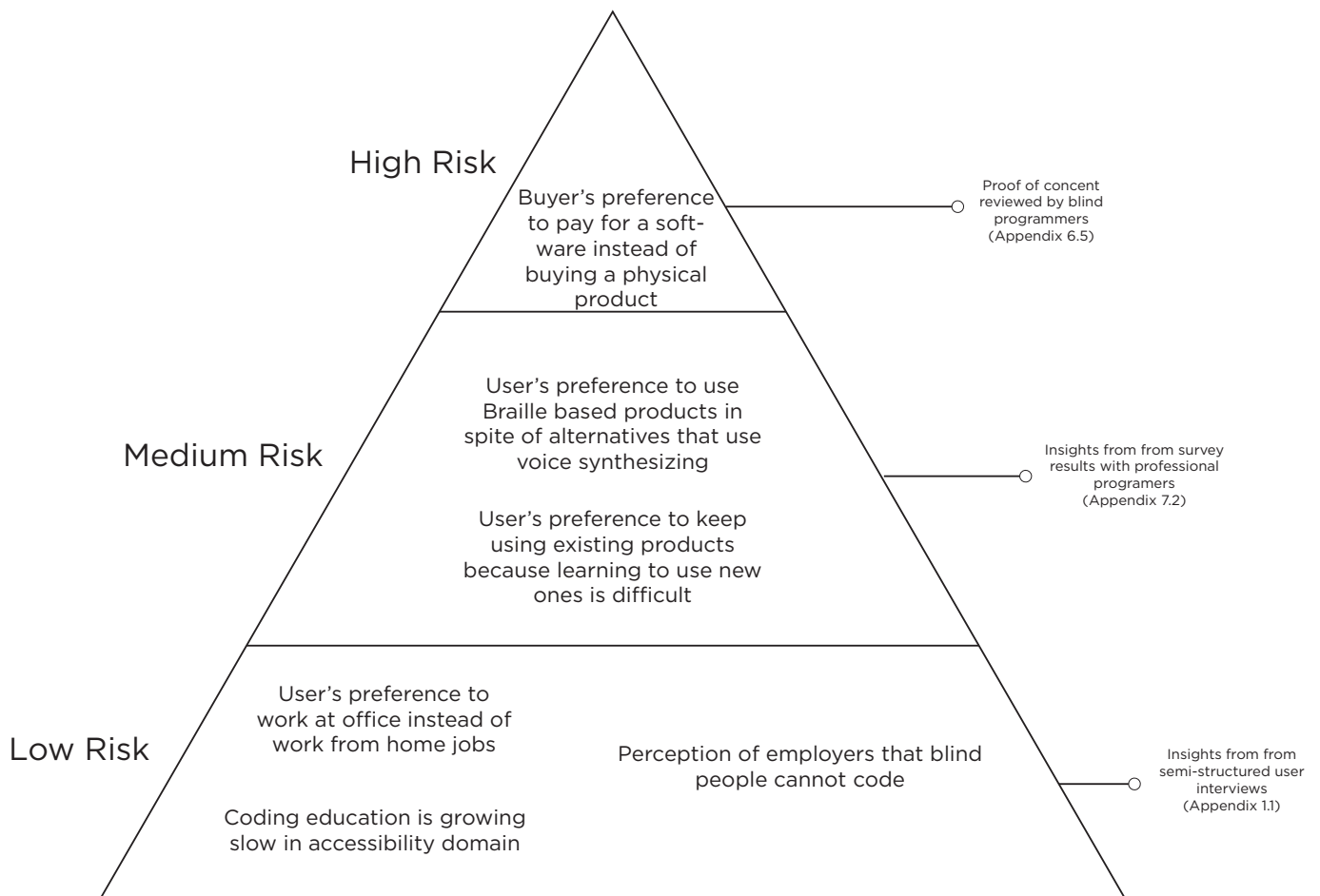


Figure 72 Risky Assumptions Pyramid

6.2 Costing and Pricing

| Overview | | |
|--------------|------------------------|---------------|
| S no | Particulars | Price 1 unit |
| 1 | Electronics | 62.27 |
| 2 | Embodiment | 64.69 |
| 3 | Shipping and Logistics | 3.22 |
| Total | | 130.18 |

Table 8 Overview of Cost of Goods Sold (COGS)

| ELECTRONICS | | | | | | |
|-------------|---------------------------|-------------------------------|-----|-----------------|--------------|---------------------------------------------------|
| S NO | Particulars | Specification | Qty | Unit Cost (EUR) | Total | Reference |
| 1 | Arcade Button | 16mm | 1 | 1.25 | 1.25 | |
| 3 | Audio Jack male to female | 35mm | 1 | 2.4 | 2.4 | |
| 5 | Thumb Joystick | Analog 2 axis | 1 | 6.95 | 6.95 | https://www.kiwi- |
| 7 | Micro stepper motors | - | 10 | 0.33 | 3.39 | https://www.aliex |
| 8 | Power Bank | 5000 mAh, 2.1A | 1 | 14 | 12 | https://www.ama |
| 9 | Rocker Switch | MRS-101-9 | 1 | 0.75 | 0.75 | https://www.kiwi- |
| 10 | Push Buttons | Round, 12mm | 10 | 0.46 | 4.63 | https://www.kiwi- |
| 11 | OTG MicroUSB | OTG to Female USB hub | 1 | 5.95 | 5.95 | |
| 12 | SD card | 16GB | 1 | | | |
| 13 | Rpi power supply | 5V DC, 2.5A | 1 | 10.95 | 10.95 | https://www.kiwi- |
| 14 | ULN2003a | | 10 | | | |
| 15 | Raspberry Pi Zero W | V1.1 | 1 | 10.71 | | |
| 16 | MDF Wood sheet | 3mm thickness 900 x 300 mm | 1 | | | |
| 16 | Poplar wood sheet | 5mm thickness 900 x 150 mm | 1 | 5 | 5 | |
| 17 | PTFE Lubricant Spray | - | 1 | 9 | 9 | |
| | | | | Total | 62.27 | |

Table 9 Cost of electronic components

| Embodiment | | | | | | |
|------------|----------------|---------------------------|--------------|-----------------|--------------|-------------------------------------------------------------------------------------------------------------------------------|
| S No | Particulars | Specification | Qty | Unit Cost (EUR) | Total | Reference |
| 1 | Laser cut | 814 x 508mm | 1 | 19.92 | 19.92 | https://cotter.co |
| 2 | Material | MDF 3mm | 1 | 5.36 | 5.36 | https://cotter.co |
| 4 | Lead screws | SLS 3D printing, Nylon | 5 | 5.46 | 27.3 | https://www.3dhubs.com |
| 5 | Travelling Nut | 3D printing, Nylon | 5 | 1.58 | 7.91 | https://www.3dhubs.com |
| 6 | Fasteners | MS, 2in M2 machine screws | 6 | 1.23 | | https://www.gamma.nl/ |
| | | MS, 2in M4 nuts | 6 | 1.27 | | https://www.gamma.nl/ |
| 7 | PCB | 2 Layer, Copper | 1 | 1.6 | | https://www.seeedstudio.com/fusion.html |
| 8 | Materials | ABS Plastic, Nylon 6 | | 3.79 | | http://www.custompartnet.com/estimate/injection-molding |
| 9 | Mould | Injection Moulding | | 24.48 | | http://www.custompartnet.com/estimate/injection-molding |
| | | | Total | 64.69 | 60.49 | |

Table 10 Cost of manufacturing various components and embodiment

| Shipping | | | | | | |
|--------------------------------------|-----------------------------------------|------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Particulars | Specification | Prices per 1000 pieces | | Reference | | |
| | | Unit Cost (EUR) | Total | | | |
| Shipping from Shenzhen to Rotterdam | Freight charges | 0.69 | 690.00 | | We are assuming a 20 feet container with dimensions 20 into 8 into 8ft Based on our calculation, approximately 1645 of our products can fit into one such container The estimated cost of shipping from Shenzhen port to Rotterdam port is 1142 euro for 1645 products. Therefore, it is 0.69 euro per product in terms of transportation | |
| Cardboard boxes or Folding box board | 30 x 15 x 24 cm Solid Bleached Board | 0.53 | 530 | https://www.biopack.ro/en/boxes-price-calculator.html#calcul | | |
| Packing Tape | 50mm x 66m x 35my | 1 | 50 | 0.05 | https://www.kortpack.nl/en/custom-made-packaging-tape/custom-made-vinyl-packaging-tape-4-colors/printed-vinyl-packaging-tape-50mm-x-66mtr-4-colors-printed | |
| Bubble wraps (LDPE) | 50cm x 100m, 60r | 1 | 500 | 0.50 | https://www.kortpack.nl/en/protective-packaging/bubble-foil-bubble-wrap-50cm-x-100m-60my-1-roll | |
| | | 3.22 | 1,770.00 | | | |

Table 11 Shipping and Logistics Costs

6.3 Revenue Model



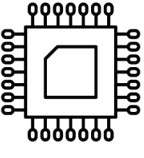
| Customer Segment | Desired Value | Production Method | Production Costs | Gross Profit /Gain |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------|-------------------------|------------------|--------------------------------------------------------------|
| End users Educators Blind schools |  Finished Product | Bulk manufacturing | 250 Euros | 425 Euros (Margin: 175 Euros) |
| Researcher Designer Developer |  Contribution based discounts | Bulk manufacturing | 200 Euros | 350 Euros (Margin: 150 Euros) + Product Development |
| Maker/Tinkerer |  Technical Designs | On-demand Manufacturing | 450 Euros | 0 Euros + Product Development |

Table 12 Revenue model for Discount based on contributions

6.4 Features and Issues

The features included in the final concept were mapped along with its corresponding issues. The plan is to solve the issues in the next version after graduation.

| Phases | Version | Features | Issues |
|-----------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 Product Development | V1 | <p>Interaction</p> <ul style="list-style-type: none"> • Haptic Skimming: 5 lines • Read line number • Move cursor to specific line • Read indentations • Continuous scrolling <p>Chassis</p> <ul style="list-style-type: none"> • Organic Hinges for orientation • Material variants - MDF & PLA • Glued joints • DFMA design for easy DIY <p>Electronics</p> <ul style="list-style-type: none"> • Assembled on protoboard • Powered with battery bank to handle current fluctuations | <ul style="list-style-type: none"> • User has to wait for the time delay before he can skim to next 5 lines • User can only skim through 5 lines at a time • Glued joints does not allow easy reparability • Living hinges tend to break under perpendicular forces. Need internal reinforcements • Protoboard can pose wiring failures and is difficult to assemble • Battery bank needs to be switched on, opening the backcover |
| | V2 | <p>Interaction</p> <ul style="list-style-type: none"> • Higher resolution: 10 lines • Haptic motor feedback for debugging on specific line • Fast scrolling: Wait for user joystick input release to read lines <p>Chassis</p> <ul style="list-style-type: none"> • Internal supports for living hinges • Lead screws printed with self lubricating Nylon 12 using SLS printing • Screwable joints in both variants <p>Electronics</p> <ul style="list-style-type: none"> • Assembled on PCB • Micro stepper drive | <ul style="list-style-type: none"> • SLS 3D printing may be not available as commonly as FDM printing • Lead screws for Micro steppers may be weak with perpendicular loads from user's touch and push switching • Micro stepper may not have enough holding torque for handling user's touch |

Table 13 List of issues in the current version and its possible solution