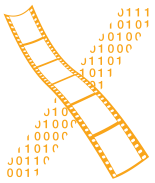


Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

AirPixel:

Prototyping a Pneumatically Actuated Shape Display with a Textile Surface

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

by
Mathis Beck

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Dr. Heiko Müller

Registration date: 26.07.2021
Submission date: 26.11.2021

Zentrales Prüfungsamt/Central Examination Office

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Name, Vorname/Last Name, First Name

Matrikelnummer (freiwillige Angabe)
Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Ort, Datum/City, Date

Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Ort, Datum/City, Date

Unterschrift/Signature

Contents

List of Figures

List of Tables

Abstract

These days, smart home devices get more and more popular. They integrate modern computers into our homes and day-to-day lives. This brings computer technology closer than ever to the vision of Weiser [1999] that "the most profound technologies are those that disappear" and "weave themselves into the fabric of everyday life" [1999]. However, the increasing number of smart home devices also implies an increasing number of interfaces like remote controls, touch panels, or smartphone apps. A promising approach to combine these interfaces and seamlessly integrate them into our homes are so-called shape displays. These displays can approximate arbitrary 3D shapes and objects in real-time. This makes them useful for rendering user interfaces that can be touched and explored with haptic sensations but can also dynamically change and adapt. However, these displays are rigid, bulky machines and the objects they display are formed from hard metal or plastic. Thus it is difficult to integrate them in soft environments close to humans, e.g. into a sofa.

To overcome this gap between the hardness of the machine and the softness of the human body, we present a novel prototype of a shape display featuring a soft and pliable surface. The prototype is actuated by a set of pneumatically driven silicone cells. The design of the control systems provides enough flexibility to make the integration in tight spaces possible. We provide insights into what considerations led us to the creation of our prototype, describe the implementation process in detail, share what we learned during the implementation, and evaluate our design with a user study.

Überblick

Smarthome Geräte werden heutzutage immer verbreiteter und lassen so moderne Computer mehr und mehr Teil unserer Wohnungen und unseres Alltags werden. Damit kommt Computertechnologie der Vision von Weiser [1999] immer näher, dass die tiefgreifendsten Technologien die sind, die in den Hintergrund unserer Wahrnehmung verschwinden und sich nahtlos in unser Leben einfügen. Jedoch bringt die steigende Zahl an Smarthome Geräten auch eine steigende Zahl an Steuerungsgeräten mit sich wie z.B. Fernbedienungen, Smartphone Apps oder Touchpanels. Ein vielversprechender Ansatz, der all diese Steuerungen zusammenfassen und subtil in unsere Wohnungen integrieren könnte sind sogenannte Shape Displays. Diese Displays können nahezu beliebige dreidimensionale Formen und Objekte darstellen. So können Benutzeroberflächen kreiert werden, die angefasst und erfühlt werden können, die aber auch dynamisch veränderbar und anpassbar sind. Allerdings sind diese Shape Displays starre, klobige Maschinen und die Objekte, die sie darstellen, sind aus hartem Metall oder Plastik geformt. Somit ist es schwierig, Shape Displays in weiche Umgebungen zu integrieren, in denen sich der Mensch in unmittelbarer Nähe befindet, wie es zum Beispiel bei einem Sofa der Fall ist.

Um diese Differenz zwischen den harten, unnachgiebigen Maschinen, die uns umgeben und unserem eigenen weichen Körper zu überwinden, stellen wir einen neuartigen Prototypen eines Shape Displays mit einer weichen, nachgiebigen Oberfläche vor. Der Prototyp wird durch aufblasbare Silikonzellen betrieben. Unser flexibles Design der Kontrollsysteme macht eine Integration in enge Bereiche möglich. Zunächst geben wir einen Einblick in die Überlegungen, die uns zur Erstellung des Prototypen bewegt haben. Dann beschreiben wir im Detail den Aufbau des Prototypen und was wir während der Konstruktion des Selbigen gelernt haben. Abschließend evaluieren wird unsere Arbeit mithilfe einer Nutzerstudie.

Acknowledgements

I thank my supervisors Oliver Nowak and Anke Brocker for providing the initial idea of this project, for their guidance, and for the welcoming and supportive environment, they provided throughout the whole project.

I thank Prof. Dr. Jan Borchers and Dr. Heiko Müller for examining the thesis.

And finally, I thank everyone who participated in the user study.

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in American English.

Download links are set off in coloured boxes.

File: myFile[a]

[a] http://hci.rwth-aachen.de/public/folder/file_number.file

Chapter 1

Introduction

Interfaces for smart home devices do not integrate well into our everyday lives

Computer technology in our homes becomes more and more ubiquitous. TVs, music systems, thermostats, dishwashers, and other smart home devices all include computers these days. Thanks to silicon technology our computers became so small and cheap they get integrated into all kinds of objects of our everyday life and we are already at a point where we are not actively noticing some of these integrated computers anymore. On normal days, people do not admire the processing power of their washing machines. As Weiser [1999] described it "the most profound technologies are those that disappear". However, this vision of computing technology operating in the background and assisting us without requiring our attention breaks when it comes to the interfaces we use to control all these smart devices. Most of them need their own separate control. Thus an increasing number of smart home devices also means an increasing number of remote controls, touch panels, smartphone apps, and other interfaces requiring our attention.

GUIs versus TUIs

All these common interfaces can be split into two groups. Either the interaction is performed via a graphical user interface (GUI) typically displayed on a 2D screen or a tangible user interface (TUI) like a remote control. Both offer distinct advantages. GUIs are flexible and can quickly adapt to the current program state. E.g. if you press a play button on a touchscreen it can change its appearance to a pause button. TUIs provide tactile sensations to the user. As Har-

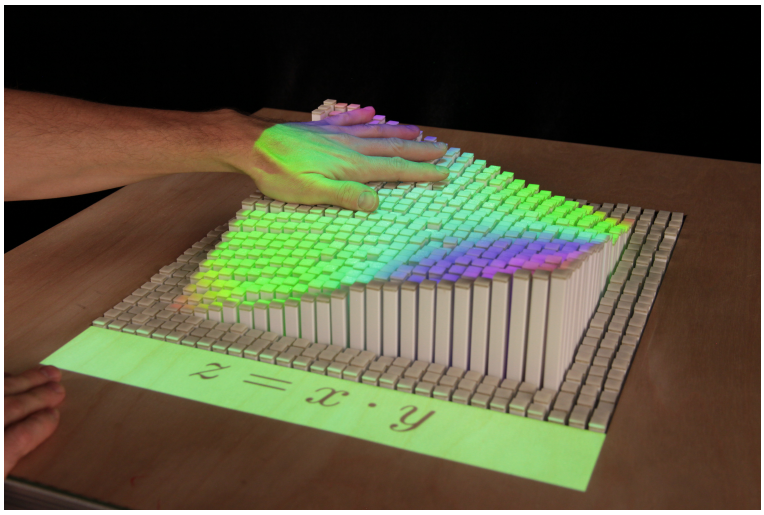


Figure 1.1: The inFORM Shape Display from Follmer et al. [2013], showing a math function. The shape is formed by a grid of square, plastic rods. The height of every single rod is adjusted so that they all together form a relief of the desired shape. Additionally, a projector projects a color image on top of the rods. Image taken from [Follmer et al., 2013]

rison and Hudson [2009] described, the haptic impressions perceived when touching a physical button make it easier to locate, it can be operated without looking at it and it even requires less attention to operate compared to the graphical counterpart on a touchscreen. As Follmer et al. [2013] mentioned, TUIs also provide real physical constraints. E.g. if you push a volume slider on a sound mixing board to its maximum, you can feel the physical constraint that prohibits you from pushing the slider further. GUIs often try to mimic real-world UI Elements such as buttons or sliders but they can only visually imitate these constraints. Consider e.g. the volume slider on a PC.

TUIs are static

However, the fact TUIs consist of real physical objects limits their abilities to be changed. Because of this typically static nature of TUIs, they are not able to provide much information about the program state to the user. E.g. the appearance of a button on a remote control can not change from a play to a pause button. This can lead to interfaces having lots of buttons all with specific different functional-

ity. A TV remote control can easily have 40 or 50 buttons most of them responsible for a specific function.

Shape displays combine the advantages of GUIs and TUIs

To combine the best of both worlds so-called shape displays got explored in human-computer interaction (HCI) research [Poupyrev et al., 2007, Follmer et al., 2013, Siu et al., 2018]. These shape displays pursuit the vision of Sutherland [1965] of a display that can manipulate matter and render any 3D object in space. Although technology is not on this level yet, today's implementations of such shape displays can approximate at least the reliefs of arbitrary objects. A common implementation approach is to build an array of densely packed rods that can be moved up and down as seen in Figure 1.1 from Follmer et al. [2013]. Similar displays were implemented by Leithinger et al. [2011], Ishii et al. [2015], Siu et al. [2018]. Each rod functions as a pixel in 3D space. The top parts of the rods form the relief of the rendered shape. User input in the form of pressing down on the rods can be recognized by constantly measuring their position.

Shape displays could invisibly integrate interfaces into our homes

These shape displays offer great potential to seamlessly integrate interfaces into our homes. They can adapt their interface to the requirements of different smart home devices eliminating the need for separate controls. And as they are real physical objects they can be integrated into the surroundings and eventually vanish into the background of our perception. Ishii et al. [2015] demonstrated that concept by building a table with three integrated shape displays.

The sofa-use-case

As stated by Weiser [1999] the location of such interfaces is of great importance. As the sofa is an important part of many living rooms we use this location as an exemplary use-case throughout this paper. When we think of a classical setup with a sofa and a TV in front, users could control the TV and other smart home devices directly from the sofa where they are sitting. To be easily accessible a shape display could be integrated into the armrest of the sofa.

Problems of shape displays: size and rigidity

However, as far as we know there are two main problems with current implementations of shape displays making them unsuitable for the above-described use case. The first one is their size and construction. The inFORM device

Figure 1.2: The part of our prototype users interact with - the bubblepad. A ring of six cells is currently inflated. A finger presses down on one of the cells.

from Follmer et al. [2013] is basically a rigid cuboid with the top face measuring about 0.7 m x 0.7 m and a height of about 1.1 m. This makes the device inflexible and principally impossible to integrate into e.g. the armrest of a sofa or other small furniture. The surface texture poses the second problem. The discrepancy between the rigidity of most technology we use and the softness of the human body led to a general interest in HCI research to explore more soft and pliable materials [Gohlke et al., 2016]. The hard and inflexible surface of shape displays is no exception and poses another hurdle integrating them into soft environments like a sofa or an armchair.

Our prototype

To overcome these two problems we propose a prototype of an adjustable interface featuring a soft, pliable surface that can be integrated into tight spaces.

The bubblepad is the part of our prototype a user interacts with. It is actuated by pneumatically driven silicone cells

The interface itself - from now on called bubblepad - can be seen in Figure 1.2. It is actuated by an array of silicone cells. Each cell can be inflated with pressurized air, raising the surface of the cell from a flat state to a dome-like shape. Additionally, each cell features an integrated pressure sensor allowing for user input by measuring how much an inflated cell is pressed down. By controlling the air pressure of each cell individually they act as a kind of pixel. Together they form a pattern of inflated and deflated cells on the bubblepad representing the current user interface. This pixel-based approach is why we called our concept AirPixel.

The silicone provides a soft and pliable surface. It gets covered with a sheet of fabric making it easy to seamlessly integrate into sofas, armchairs, or other furniture with textile surfaces. Because the bubblepad is controlled by pressurized air it only has to be connected to the control system with small, flexible silicone tubes. This loose coupling allows the bubblepad to be integrated into e.g. the armrest of a sofa while the control system can be stored elsewhere e.g. under the sofa. A more detailed description of our prototype is provided in chapter 3.2 “Overview Over the Bubblepad”.

Outline of our work

In this paper, we first provide insight into related work that inspired the creation of our prototype. Then we present the working principles of the prototype and discuss the implementation details. Finally, we present the result of a user study we conducted, discuss possible improvements for the prototype, summarize our contributions, and give an outlook for future research.

Chapter 2

Related work

This work is mainly based on knowledge from three research topics. While our prototype provides - to the best of our knowledge - a novel combination of these research areas, we draw heavily from the already existing work. Whereas two of three topics, namely shape displays and pneumatically actuated shape-changing interfaces, both belong to the HCI research area, the third topic, soft robotics, is a broader, interdisciplinary research area. This chapter explores for our prototype relevant aspects of these three topics and relates them to our work.

2.1 Shape Displays

Advantages of shape displays

The main inspiration for how the interaction with our prototype could work originates from the inFORM shape display from Follmer et al. [2013]. They demonstrated that shape displays can be used to implement dynamic TUIs offering real physical constraints. Unlike other TUIs that mostly consist of static objects, the interface created by a shape display can be rapidly adapted to the current state of the device or program. Nevertheless, they can leverage the potential of the physicality of the rendered shape to provide haptic sensations. When rendering e.g. a play button on a shape display, not only can a user feel the shape of the but-

ton, the shape can also change to a pause button whenever required.

The construction method of shape displays makes them bulky and inflexible.

In the current state of the art, these shape displays are constructed using an array of movable rods. [Follmer et al., 2013, Siu et al., 2018] The inFORM display uses for each of the 900 rods a motorized slide potentiometer for actuation and sensing. A kind of Bowden cable links the potentiometers to the rods. This control system requires a large space underneath the display surface. Together with the control electronics the whole device measures about 1.1 m in height and about 0.7 m x 0.7 m in width and length. This makes the device hard to integrate into existing objects, especially regarding the use case mentioned in the introduction with a sofa. More recent implementations like the one from Siu et al. [2018] manage to fit a display with 288 rods into a package with a height of about 27 cm. While this is a more reasonable size the integration into a sofa would still be a challenge as the system can not be split into separate parts but has to be integrated as a whole. None of the implementations we know addresses the second issue we mentioned in the introduction regarding the hard surface texture.

A visionary concept to combine GUIs and TUIs

Robinson et al. [2016] describe a similar approach they call Emergables to combine TUIs and GUIs. The goal they describe is that primitive UI elements like dials, sliders, etc. can emerge in a real 3D physical form from a touchscreen whenever they are needed. While still far away from this goal they created a low-resolution prototype similar to the inFROM shape display. The main difference is that user input can not only be generated by pressing on the rods, but also by tilting them in any direction or turning them around the z-axis.

2.2 Soft Robotics

Soft robots are driven by soft actuators often using pressurized air

To achieve a soft and pliable surface on our prototype we made use of principles commonly used in soft robotics. As the name suggests soft robots are made of soft materials. A common technique in soft robotics is the usage of soft actuators operated by pressurized air. Air has the advantage of

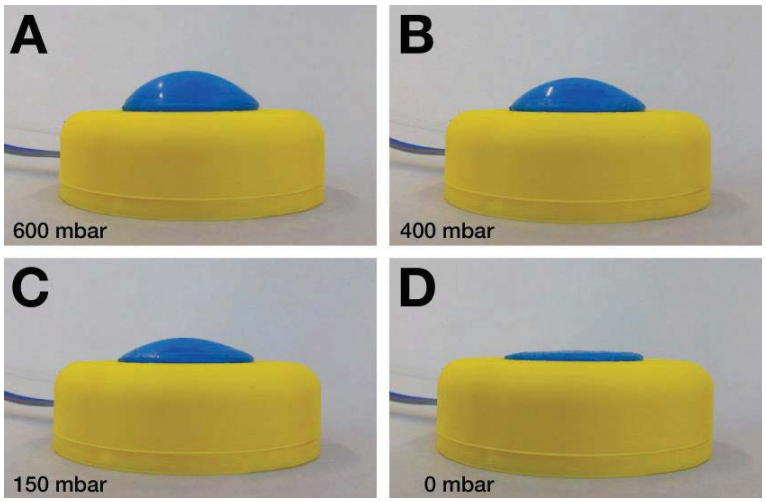


Figure 2.1: A pneumatically actuated button created by Gohlke et al. [2016]. As seen in the picture, the button can be inflated to different heights with different pressure levels. They used rapid changes in pressure while a user presses the button to generate pneumotactile feedback. A Hall sensor and magnet are integrated into the button to sense user input by measuring the height of the button. Image taken from [Gohlke et al., 2016].

being lightweight, compressible, easily available and due to the low viscosity, it supports fast transfer and actuation. [Ilievski et al., 2011]

A soft, pneumatically actuated button

The usage of soft pneumatic actuators eliminates the need for complex mechanisms and allows for continuous, organic shape changes. Gohlke et al. [2016] identified design parameters that are important to take into consideration designing such pneumatic actuators. This includes e.g. the hardness of the material, minimum and maximum pressure levels, maximum extension, and more. To demonstrate their findings they designed a button-like pneumatic actuator made of silicone (see Figure 2.1). The button was constructed by fusing two layers of silicone creating a zero-volume air chamber in between (explained later in Figure 3.3). The bottom part is embedded in a 3D printed base structure to limit the extension to one direction. To inflate the button air is forced in between the two silicon layers

with a small tube. The top silicone layer expands into a dome-like shape. The advantage of such a zero-volume air chamber is, that the button returns to its original flat shape as soon as the pressure is released. The air pressure is generated by a small diaphragm air pump. They used a combination of air tanks and solenoid valves to control the pressure inside the actuator. By rapidly changing the pressure inside the button by small amounts while a user presses it, they were able to create pneumotactile feedback. [Gohlke et al., 2016]

Detailed descriptions of methods how to manufacture such zero-volume air-chambers as mentioned above are featured in the work of Park et al. [2014]. We implemented one of their methods and describe it later in section 3.3.1 "Cells and Bubblepad" in more detail.

2.3 Pneumatically Actuated Shape-Changing Interfaces

2.3.1 PneUI: Multi-Layer Composite Materials

Using composite materials to create shape-changing objects

Regarding pneumatically actuated shape-changing interfaces, the work from Yao et al. [2013] provided a lot of the fundamental knowledge used in more recent work in this area e.g. the pneumatibles paper by Gohlke et al. [2016]. They call their approach PneUI and describe how multi-layer composite materials can be used to create objects able to change their shape and sense user input. Their proposed material consists of three main layers. The first layer is made of an elastomer and can contain embedded air channels. This layer expands evenly when pressurized. If the expansion should behave differently in certain directions a second layer is needed. This layer consists of rather inelastic materials like e.g. paper or fabric and features special surface structures. After these two structural layers, a third sensing layer can be added containing e.g. liquid metal or conductive thread to sense deformation in certain directions.

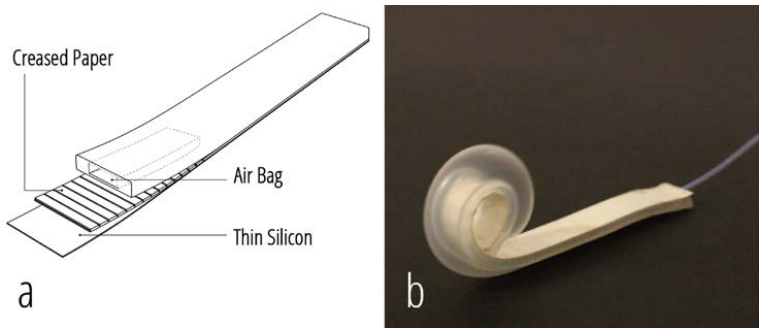


Figure 2.2: An example demonstrating the PneUI composite material concept from Yao et al. [2013]. It demonstrates how to create a curving pneumatic actuator. The actuator features two main layers. The first one is molded from silicone and features an integrated air channel. The second one is a sheet of paper with a special crease pattern. When the air channel is pressurized the actuator bends around itself. Image taken from [Yao et al., 2013].

Example of a curving actuator

An example of use can be seen in Figure 2.2. When the air channel in the first silicone layer is pressurized the silicone expands. However, the paper layer prohibits expansion on one side. This causes one side to be longer than the other one and results in a curvature of the actuator. The curvature can be influenced both by the amount of air pressure and the paper crease pattern.

All kinds of shape changes are possible

In their work, they describe how to achieve numerous primitive shape changes like e.g. different curvatures or expansions into certain directions. Furthermore, they provide methods on how to achieve texture changes by using very small scale shape changes and how to integrate sensors for structural changes to e.g. sense if an actuator gets stretched. They conclude by demonstrating possible applications of their technology with four different prototypes. [Yao et al., 2013]

Similar to the layer concept of PneUI our cells are made of an elastomer expanding uniformly into all directions and a rigid base structure limiting the expansion into one direction.

2.3.2 3D Printing Pneumatic Actuators

3D printing allows the creation of nearly arbitrary shapes but removing support material is difficult

Vázquez et al. [2015] present a quite different approach to manufacturing elastic pneumatic actuators. They used multi-material photopolymer 3D printing to create the rigid parts as well as the elastic parts of an actuator. 3D printing provides a lot of freedom regarding the geometry of objects to fabricate. Additionally, with a multi-material printer rigid and elastic parts can be directly fused together during the printing process. When printing hollow structures like air chambers, current 3D printing technology requires the use of support material to support overhanging geometry. However removing the support material posed a severe problem, especially in thin structures like tubes and on elastic surfaces. They described two methods how to avoid this problem. The first one is to split the object to print into multiple parts in order to provide better access to inner parts and ease the process of removing the support material. The single parts can then later be assembled together, e.g. by integrating threads into the design. Alternatively, they replaced the support material by permanently placing prefabricated support structures into the object while printing.

Examples for 3D printed interactive devices

With their technique, they manufactured a variety of interactive devices. This includes e.g. a turning knob or a slider both using air chambers to control the posed resistance while turning or sliding. To sense user input they used off-the-shelf components like rotary or slide potentiometers. They also printed a small button, similar to the cells we produced, and used a pressure sensor to measure user input.

Our cells also use 3D printed parts

To construct our cells we also used 3D printing to print the rigid base structure of the cells. However, we did not print the elastic air chamber on a 3D printer but instead molded it from silicone. This simplified the printing process and eliminated the problem of having to deal with the support material removal. Thus we were able to use a lower resolution FDM printer instead of a photopolymer printer, leading to faster print times and a faster prototyping process.

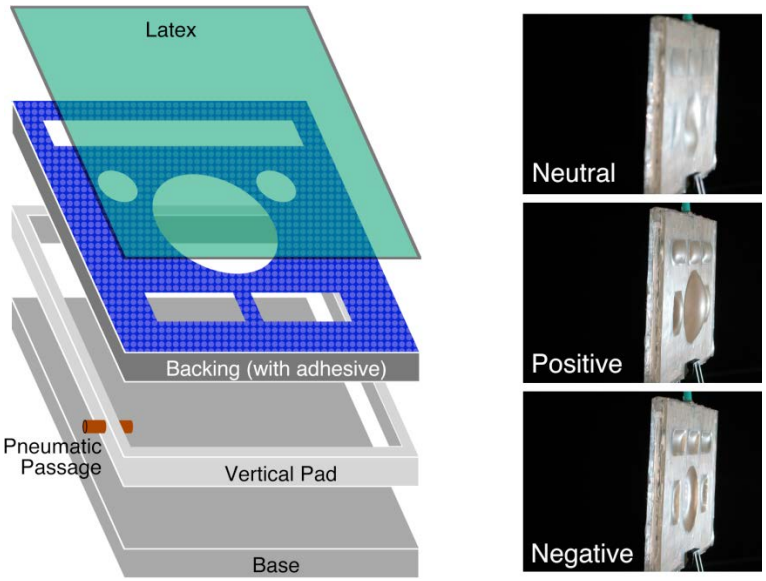


Figure 2.3: A display with inflatable buttons created by Harrison and Hudson [2009]. The shape of the interface is determined by the laser-cut acrylic backing plate by cutting out the pressable features from the acrylic. A sheet of latex is glued over the backing plate. Behind the backing plate is a rigid, airtight chamber with a small inlet. By either pressurizing or depressurizing the air chamber, the latex creates the indented positive or negative detents on the surface, as seen on the right of the image. Image taken from [Harrison and Hudson, 2009].

2.3.3 A Static Inflatable Display

Closely related to our approach Harrison and Hudson [2009] created an adaptable interface with a pliable surface and pneumatic actuation. They use laser-cut acrylic plates to determine the shape of the interface. A latex sheet provides a pliable surface. The exact construction method is described in Figure 2.3.

With the help of a rear projector, an image can be projected onto the surface. Touch input gets recognized with the help of an infrared light source and infrared camera. Advantages of the system are the simple and cheap construction and large dimensions in which it can be produced. A dis-

UI elements can be inflated or deflated but are otherwise static

advantage of the approach is the static placement of the UI Elements. In the simplest form, the only way the interface can be changed is by inflating or deflating all UI Elements simultaneously. By separating the air chamber so that each UI ELement has its own airtight chamber, individual UI Element can be independently inflated or deflated. But position and shape of the UI Elements stay fixed. [Harrison and Hudson, 2009]

Barometry can be used to sense user input

In their work Harrison and Hudson [2009] also evaluated how barometry can be utilized to measure user input. Their findings were a major influence on the way we are measuring user input for our prototype. When a user presses down on an inflated compartment the pressure in the air chamber rises. The force exerted to a compartment when it gets pressed is directly related to the displacement distance and the increased pressure in the air chamber. By analyzing the pressure data over time, press and release events can be extracted.

2.3.4 Controlling Airflow

Solenoid valves control the airflow

The pneumatic control systems used to control the airflow to pneumatic actuators are very similar amongst all the approaches. There is a source of pressurized air, either provided by an air compressor or another type of electric air pump. Air tanks are used to smooth out pressure fluctuations (created by the compressor or pump) or to have different levels of air pressure quickly available. Ultimately solenoid valves are used to enable or disable the airflow to the actuators. They can be controlled electronically by microcontrollers making the whole system computer-operated. [Gohlke et al., 2016, Delazio et al., 2018, Harrison and Hudson, 2009, Kim et al., 2008, Teng et al., 2018, Yao et al., 2013, Vázquez et al., 2015] For our prototype, we use the same approach.

2.4 Relation to Our Prototype

The foundation of our prototype

One way to look at our prototype is it being a mix of the inFROM shape display from Follmer et al. [2013], the inflatable button from Gohlke et al. [2016], and the adaptable display from Harrison and Hudson [2009]. We tried to combine the benefits from these approaches. Like Harrison and Hudson [2009] our prototype features a soft, pliable surface and different UI Elements can be displayed. However, we did not want the position and shape of the UI elements to be static. Therefore we used an array of cells to act as kind of pixels like the rods in the inFORM shape display. The cells themselves are similar to small versions of the inflatable button from Gohlke et al. [2016].

Chapter 3

Our Prototype

In this chapter, we describe our prototype from a top-down view. First, we summarize the findings presented in the previous chapter we gained from related work by formulating a set of design goals. Then we describe the prototype in increasing levels of detail. We start by presenting an overview of the end result, describing the appearance and capabilities of the bubblepad (the part of the prototype users interact with). In the second step, we provide an overview of all components that make the bubblepad work and describe the underlying working principles. In the final step, we give a detailed explanation of the whole implementation, providing references for how to build and assemble each part.

Additional resources like 3D models and source code can be found in the following repository on the RWTH GitLab. We will reference the contents of this repository later in the text at appropriate places.

git.rwth-aachen.de/i10/thesis-mathis-beck-airpixel[a]

[a] https://git.rwth-aachen.de/i10/thesis-mathis-beck-airpixel/-/tree/main-mathis

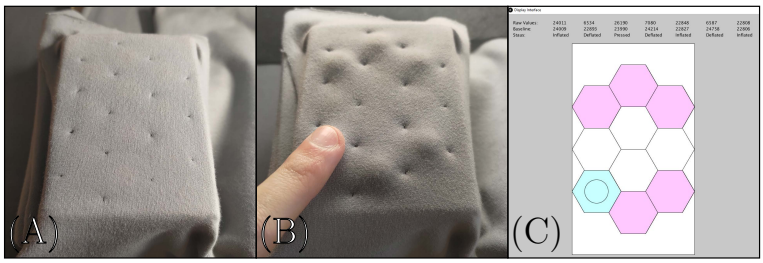


Figure 3.1: (A) shows the bubblepad in a flat state. No cell is inflated. (B) shows a pattern of inflated cells on the bubblepad. One cell gets pressed. (C) shows our desktop application rendering the state of the bubblepad in (B). The color-coding of the hexagons is explained later in table 3.1.

3.1 Design Goals

By reviewing the related work presented in the previous chapter we drew up a list of design goals for our prototype.

- The rendered interface should be dynamically changeable in real-time
- It should be possible to render arbitrary shapes
- The output should be of visual and haptic nature
- Detecting user input should be possible
- The surface should be soft and pliable and it should be possible to cover the prototype with a sheet of fabric

3.2 Overview Over the Bubblepad

The bubblepad is a rectangular textile surface covering ten inflatable cells

Figure 3.1 (A) shows the finished bubblepad. The textile-covered surface is about 7.5 cm x 10 cm in size. It features ten individually controlled, inflatable cells arranged in a hexagonal pattern. The cells are placed directly under the visible, top layer of fabric. When a cell is deflated it is completely flat, when it is inflated though it creates a bump on

the surface. More cells can be inflated in specific patterns to create intricate UIs (Figure 3.1 (B)). The rendered cell pattern can be both visually perceived by looking at the bumps and through haptics by feeling the bumps with the fingertips.

Every cell has an integrated pressure sensor

Every cell is equipped with a pressure sensor to provide user input. Users interact with the prototype by pressing on inflated cells. Thus a single inflated cell acts kind of like a button. However the input is not binary but a continuous pressure value, therefore we can not only tell if a cell is pressed but also how strong the user presses. As every cell has its own pressure sensor it is also possible to detect multiple presses on different cells at the same time.

The prototype connects to a desktop application

The prototype is controlled by software running on a desktop PC (Figure 3.1 (C)). It shows the current state of the display and controls the inflation states of the cells. The whole system responds in real-time. Pressure values are received and processed with a delay $\leq 100\ \mathrm{ms}$ and rendering a new cell pattern takes $\leq 200\ \mathrm{ms}$.

3.3 Working Principles

The whole prototype can be split into four major subsystems.

- The cells, actuating the bubblepad
- The pneumatic control system, regulating the airflow to the cells
- The electric control system, responsible for switching the solenoid valves and integrating the pressure sensors
- The software running on microcontrollers and PC, managing dataflow and tieing everything together

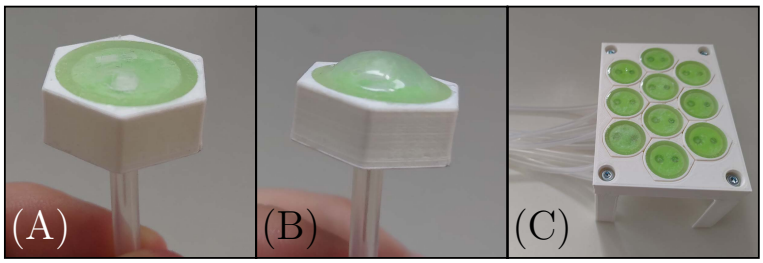


Figure 3.2: (A) and (B) show a single cell. In (A) the cell is deflated and the two silicone layers touch each other. In (B) the cell is inflated and the top silicone layer (transparent silicone) expands into a dome-like shape and separates from the bottom silicone layer (green silicone). (C) shows the bubblepad frame with all ten cells inserted.

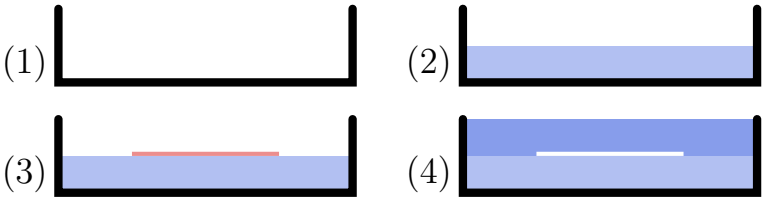


Figure 3.3: Schematic 2D depiction of how to create a zero-volume air chamber from two silicone layers similar to the method described by Park et al. [2014]. (1) shows the empty mold. In step (2) the first layer of silicone gets poured into the mold. When cured in step (3) a layer of release agent (red) is applied to the center region. In step (4) the second layer of silicone gets poured on top. When cured the two layers have bounded at the outer rim but not in the center where the release agent separated them.

3.3.1 Cells and Bubblepad

Appearance of a cell

All the cells in the bubblepad are identical. A cell consists of two layers of silicone molded in a riding 3D printed base structure. A finished assembled cell can be seen in Figure 3.2 (A)-(B). It can extend up to 8 mm, has an outer diameter of about 21 mm and the inner inflatable section has a diameter of 15 mm.

A cell consists of a zero-volume air chamber

The two silicone layers form a zero-volume air chamber. This is the same principle Gohlke et al. [2016] and Park et al. [2014] used in their work. They layered two flat layers of silicone on top of each other and bonded them together only at the outer rim as can be seen in Figure 3.3. When forcing air between the two silicone layers the middle parts separate and expand. The advantage of this design is that as soon as the pressure is released the chamber deflates itself because of the spring force of the extended silicone, pulling it back to its original zero-volume shape. It quickly turned out though, that with the small scale of the cells the bonding between the two layers posed a significant weak spot. This was mainly because of the small width of the outer ring where the layers were held together. However, we found a solution to mitigate this problem described later in the implementation part (3.4.1).

Structure of a cell

To limit the extension to one direction the lower silicone layer gets molded into a rigid 3D printed base frame. Inflating the air chamber now only causes the upper layer to expand forming a dome-like shape (see Figure 3.2 (B)). Two small silicone tubes are molded into the bottom layer of the silicone and fed through holes in the bottom of the base frame. One short tube connects a pressure sensor to the cell and the other longer one serves as a connection to the pneumatic control system. It would also suffice to mold only one tube into the cell and then connect the pressure sensor with e.g. a T-connection piece later. However molding a second tube into the cell did not complicate the manufacturing process, so we decided to mold two tubes into the cell eliminating the need for additional connection pieces.

The bubblepad frame arranges the cells in a hexagonal pattern

All the space between the cells is wasted because we are neither able to change the shape here nor measure pressure inputs. To minimize this 'dead space' between the cells we decided to arrange them in a hexagonal pattern and also give the cell base frame a hexagonal shape. We decided against a round shape of the cells to fill up all the space between the cells and prevent them from potentially sliding around. The cells are then placed into another 3D printed frame (see 3.2 (C)) providing the structural stability for the bubblepad and locking the cells in a fixed position. In Figure 3.2 (C) it can be also seen that we added four legs to the

frame so that the tubes from the cells bend to the side and the bubblepad can be steadily placed on a flat surface.

Mainly due to the cost of the valves, we were limited to a number of around ten cells. To decide how we arrange the cells we created cell patterns for some basic UI primitives that the bubblepad should be able to display. These are basically the same patterns we then used in the first four tasks of the study (see later Figure 4.1). The arrangement we ended up with (see Figure 3.2 (C) and 3.1 (A)) consists of three vertical rows of cells with three, four, and three cells. It is the only arrangement of ten cells in a hexagonal grid that supports these primitive cell patterns and is still symmetrical.

Structure of the bubblepad

The frame gets cushioned with sponge rubber to provide a soft feeling consistently over the whole surface of the bubblepad. Ultimately, the whole frame gets covered with a layer of elastic fabric. The fabric is sewed to the bubblepad frame between the cells (small black dots in Figure 3.1 (A)-(B)). This is done on the one hand to ensure that a well-defined outline is visible of an inflated cell. On the other hand, this prevents 'ghosting effects', e.g. in the case when two cells in a row are inflated with a deflated cell in between. If the fabric would not be fixed to the frame the two inflated cells would raise the fabric also over the deflated cell in the middle making it impossible to distinguish visually if the cell in the middle is inflated or not. However, the way we sewed the fabric to the frame might pose a significant disadvantage as discussed in chapter 4.7.1 "Fabric Attachment", because neighboring cells get visually separated when inflated.

3.3.2 Pneumatic Control System

The heart of our pneumatic control system consists of an air compressor supplying pressurized air and solenoid valves controlling the airflow to the cells.

SOLENOID VALVES:
Solenoid valves allow controlling the flow of a gas or fluid with the help of electric current by making use of an electromagnet. When current is passed through the coil of the electromagnet the created magnetic force lifts a plunger inside the valve either opening or closing the valve. If the valve is closed when no current is flowing the valve is called normally closed, otherwise normally opened. There are two important properties used to classify solenoid valves. The working principle of the plunger and the configuration of outlet/inlet ports and states.

Direct operated solenoid valves lift a plunger inside the valve directly with the electromagnet. However, the plunger only shuts a small orifice inside the valve. Thus these valves typically feature a small flow rate but require no pressure difference to operate. On the other hand, indirectly operated solenoid valves use an existing pressure difference to handle larger flow rates. We use only directly operated solenoid valves as flow rate is not a concern in our case and we can not guarantee a big enough pressure difference for indirectly operated valves.

2/2-way and 3/2-way solenoid valves are common types of port configurations. 2/2-way (or just 2-way) valves have two ports and two different states. They act like a simple switch either connecting the two ports or disconnecting them. 3/2-way (or just 3-way) valves have three ports and two states. One of the ports is a common port. Depending on which state the valve is in, either one of the other two ports is connected to the common port. The symbols for both valve types are depicted in Figure 3.4.

Figure 3.4: Symbols for 3-way (left) and 2-way (right) solenoid valves. The left boxes indicate the state when energized, the right boxes the 'normal' state. An arrow indicates that flow from one port to the other is possible in the direction of the arrow. The $\top$ symbol indicates a blocked port.

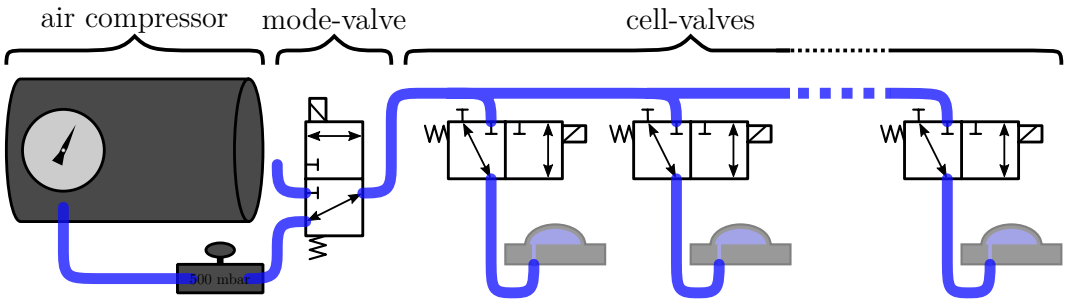


Figure 3.5: Schematic depiction of the pneumatic control system. All valves are depicted in their de-energized state. In this state, all the cells (light grey) are isolated from one another and the mode-valve connects the system to the pressure source. The pressure is generated by an air compressor and fine-adjusted with a pressure regulating valve (dark grey).

Solenoid valves control the airflow

A schematic overview of the system is depicted in Figure 3.5. Starting from the cells every cell is connected to a separate solenoid valve. These valves are normally closed and thus they isolate the cells from one another in the idle state. This ensures that the pressure sensors only read the pressure from the cell they are attached to. Even though we use 3-way solenoid valves, we blocked the third port so that they act as simple 2-way valves only letting air pass when the valve gets energized. More detailed information about which valves we have chosen are outlined in section 3.4.2. The other sides of these cell-valves are all connected together and ultimately connect to the common port of another 3-way solenoid valve, we call the mode-valve. In the idle state, the mode-valve connects the system to the pressure source. When energized it connects just to an open port, allowing the air pressure to leave the system. Thus this mode-valve allows us to switch between inflation and deflation mode.

A compressor serves as the pressure source

To provide pressurized air to the system we use a small air compressor typically used for airbrushing. There is no particular reason why we have chosen this pressure source other than already having it available. Other pressure sources are also possible or even better suited as they are quieter. The compressor maintains an internal pressure between 3 and 4 bar. As this pressure level is way too high for our purposes we use a pressure regulating valve and a digital barometer to set a constant output pressure

of 400 - 500 mbar. This output gets connected to the mode-valve.

Changing the pattern of inflated cells is a two-step process

With this system, changing the rendered cell pattern is a two-step process. In the first step, the mode-valve remains in the de-energized state pressurizing the system. The cell-valves of all cells that should get inflated get energized for a short amount of time, connecting them to the pressure source and allowing air to flow into the cells. In the second step, the mode-valve gets energized, letting air escape from the system. The mode valve remains energized until the second step is completed. Now all the cell-valves of the cells meant to be deflated get shortly energized, connecting them to the ambient air pressure and allowing air to flow out of the cells.

Other approaches like Delazio et al. [2018] use two 2-way valves for each air chamber, one for inflation one for deflation. The corresponding valves are always connected to their target pressure level. This allows the inflation and deflation process to happen at the same time, however, double the amount of valves is needed. We decided to use only one valve per air chamber due to the lower cost, the decreased space requirements, and the valves having a low enough response time of about 10 ms.

3.3.3 Electric Control System

The electric control system mainly consists of an Arduino Uno, another standalone Atmel microcontroller, multiple digital pressure sensors, and complementary components switching the current through the solenoid valves.

The Arduino controls the solenoid valves with the help of a transistor circuit

The Arduino is responsible for controlling the valves. However, typical microcontrollers can only switch currents way under 100 mA. This is far too little to power the solenoid valves directly. Additionally, the valves operate in most cases with higher voltages than the 5 V an Arduino uses. To switch the current for controlling the valves we used bipolar junction transistors and flyback diodes. The transistors allow a small current from the Arduino to

switch the larger current at a higher voltage necessary for the valves. When the valves get switched off the magnetic fields of the electromagnet coils collapse and induce a reversed current. The flyback diode provides a way to dissipate this current and avoid damage.

The atmega8 microcontroller communicates with the pressure sensors

The pressure sensors we used to measure cell pressure provide the output via a digital interface. However, they operate at 3.3 V, and thus the Arduino operating at 5 V can not directly communicate with the pressure sensors. Hence we used another standalone Atmel AVR atmega8 microcontroller running at 3.3 V to communicate with the pressure sensors via SPI. The atmega8 sends the pressure values to the Arduino also via SPI. To communicate with the control software running on a PC we used the serial library provided by the Arduino framework allowing us to easily establish a communication channel from the Arduino via USB to the control software.

3.3.4 Software

The software of our prototype consists of three different programs. One running on the Arduino, one on the atmega8, and a desktop application. All data processing and control logic is implemented in the desktop app.

The microcontrollers forward data from the sensors to a desktop application and receive instructions

As mentioned before the atgema8 microcontroller communicates with the pressure sensors. It continuously reads the data from the pressure sensors and sends the pressure data to the Arduino. This happens every 50 - 60 ms. The Arduino buffers the data, forwards it to the desktop application, and receives the intended inflation states for the cells. It then compares the received states to the current cell states and inflates or deflates cells that should change. To synchronize the communication between PC, Arduino, and the atmega8 a simple custom protocol is used that groups data into frames with well-defined start and endpoints. This is done to easily map the pressure values to the sensors. E.g. the first pressure value in a data frame always corresponds to the same pressure sensor. Otherwise, when sending the pressure values just in a continuous stream without any en-

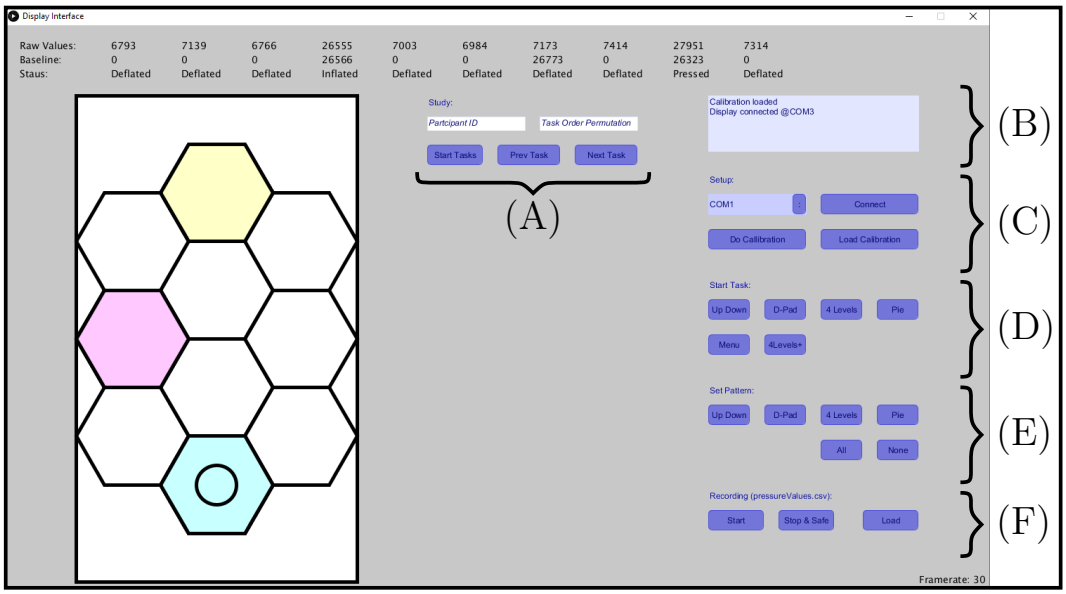


Figure 3.6: A screenshot of the desktop application. The GUI consists of three main parts. On the left, it shows a graphical representation of the bubblepad. The color coding is explained below in table 3.1. In the top row, the raw sensor values get listed. In the middle and on the right different controls are situated. (A): Controls for conducting the user study. (B): A text area for important messages. (C): Controls to set up a connection to the bubblepad. (D): Buttons to individually start the tasks used in the user study. (E): Buttons to apply predefined cell patterns. (F): Controls for recording the input to the bubblepad or playback a recording.

capsulating frames, it is difficult to match the received values to the sensors. Both programs for the microcontrollers are implemented in C++.

The desktop application depicts and controls the current state of the bubblepad

The desktop application provides a GUI to control the prototype. A screenshot is depicted in Figure 3.6. On the left side of the screen, it provides a graphical overview of the current state of the bubblepad depicting the cell states from a top-down view. Each cell is represented by a hexagon. The states are color-coded as seen in table 3.1. When pressed a circle shows the strength of the applied pressure. The larger the circle, the higher the pressure.

The application is created with Processing

The application is written in java using the Processing environment. Processing was chosen because it allows to rapidly prototype graphical interfaces, it is easy to use, it is a freely available open-source project, it is operating sys-

Color	State
White	Deflated
Yellow	To be inflated
Purple	Inflated
Blue	Pressed

Table 3.1: Color coding of the cell states in the desktop application

tem independent (runs on Windows, Mac, and Linux) and it features a simple serial library.

A more detailed description of how the desktop application and the software running on the microcontrollers operate is provided in appendix B "A detailed description of the software's operating principles".

3.4 Implementation Details

In the last section, we described how all the different parts of the prototype function and how they work together. On this basis, this section describes in more detail how each part is manufactured and which design decisions were taken. This includes especially the manufacturing process of the cells and the bubblepad but also the solenoid valves and the electric control system.

3.4.1 Cells and Bubblepad

Cell Manufacturing

We went through multiple design iterations constructing the cells

Before arriving at the final cell design we went through a few prototypes, improving the design in each iteration. One of the first prototypes can be seen in Figure 3.7. It was manufactured very closely to what was described by Gohlke et al. [2016], Park et al. [2014]. The 3D printed base frame was designed to act both as a mold and a base

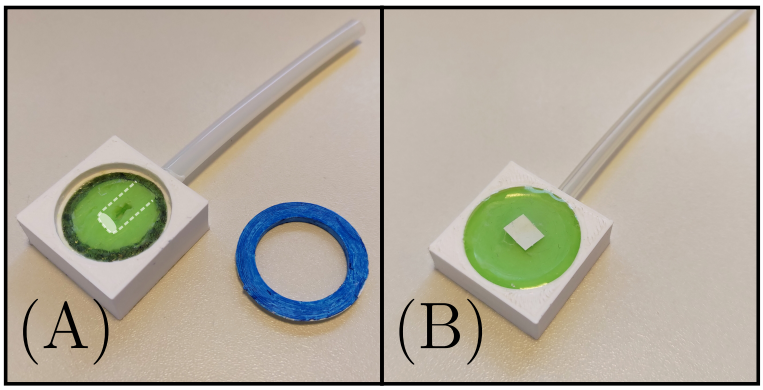


Figure 3.7: A early prototype of a single cell. (A) shows an unfinished state of the cell. The first layer of green silicone has already been cured. The white outline depicts the tube underneath the silicone. A hole through the silicone connects to the tube. The black ring marks the area where the mask (blue ring) covers the silicone and no release agent gets applied. (B) shows the finished cell. The second transparent layer of silicone has been cured. The small piece of paper that can be seen between the two layers covered the hole to the tube.

structure for the cell. In the first step, a silicone tube was inserted horizontally through a hole in the side into the frame. To prevent fluid silicone from entering the tube, it was plugged with a drop of hot glue at the end inside the mold. Then silicone was poured into the mold, completely covering the tube. After the silicone was cured, a small hole was cut through the middle of the silicone layer into the tube beneath to open a path for the air to enter the cell. This state is depicted in Figure 3.7 (A). Now a mask, consisting of a 3D printed ring, was placed on top of the cured silicone layer and a release agent was applied to the cured silicone sparing the outer rim. The hole to the tube was covered with a small piece of paper. The last step was to remove the mask, pour the second layer of silicone on top of the first layer, and let it cure. See Figure 3.7 (B).

Problem of the original air chamber design

As mentioned in the previous section this method of joining the two silicone layers was not very reliable, especially when we tried to make a smaller version of the cell. The

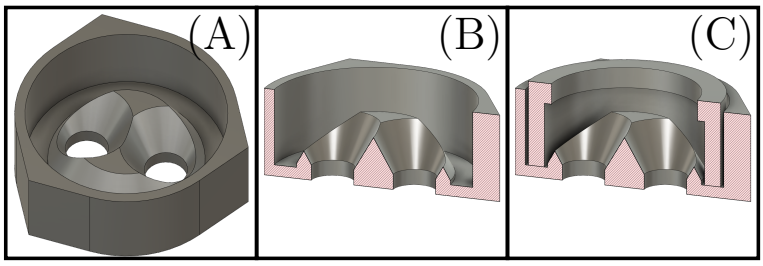


Figure 3.8: 3D model of the final cell frame design. (A) shows the whole frame. (B) shows the frame cut in half. (C) shows the half frame with the mask inserted. Before molding two tubes get inserted through the bottom holes. The first layer of silicone gets poured inside the mask. The second layer gets poured over the first layer and into the empty space the mask leaves behind when removed.

width of the ring where no release agent was applied and where the two silicone layers bonded together was only about 1 mm wide. Thus dirt particles or small imperfections when applying the release agent significantly weakened the bond. Widening the ring was not desirable, as this shrinks the part of the cell that is able to expand and thus increases the 'dead space' between the cells. As stated in section 3.3.1 this 'dead space' should be minimized.

New air chamber design

However, we found another solution to increase the bonding area between the two silicone layers. By treating the bottom silicone layer not as a flat surface but as a 3D cylinder, we concluded to use the vertical side of this cylinder to bond the layers together. Apart from this we also made several changes to the base model. As the cells should later be placed next to each other, the tube could not leave the cell horizontally. Thus we moved the hole for the tube to the bottom so it is inserted into the cell vertically and also added a second tube hole. We changed the overall shape to a hexagon and later removed every second edge to create small support pillars in the bubblepad frame. The model of the base for the final cell design is depicted in Figure 3.8. [1]

[1] The 3d models can be found in the repository under 3d_models.

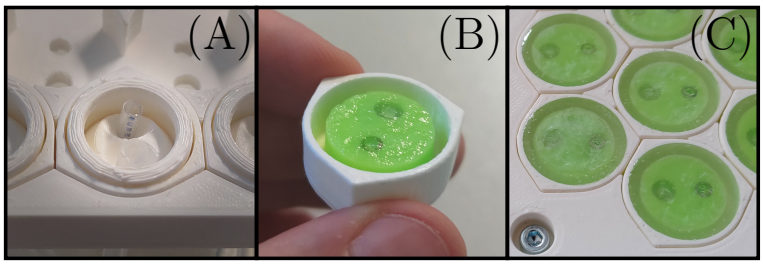


Figure 3.9: Shows three steps of the cell molding process. In picture (A) the mask coated in release agent was put into the frame and two silicone tubes were inserted from below. In picture (B) the first layer of silicone has already been cured and the mask was removed. The top part of the first layer was cut flat and coated with a release agent. In picture (C) the cells are finished. The second layer of silicone was poured over the top layer, filling the gap the mask left behind.

Molding the first layer of silicone for the final cell

Apart from the model the molding process also slightly changed. In the first step, the two silicone tubes are inserted into the base frame. The part of the tubes that stick inside the cell can be lightly sanded beforehand to ensure a better bounding with the silicone later. The tubes should stick out a few millimeters from the top of the cell. The next step is to prepare the mask by applying a release agent to the inner part and the bottom. The mask has the shape of a hollow 3D cylinder (see Figure 3.8 (C)). It is then inserted into the base as depicted in Figure 3.9 (A). Now the first layer of silicone is poured into the base. It should reach at least the level of the small overhanging edge of the mask. The surface tension of the fluid silicone prevents a flat surface from forming as the fluid is rising up at the edges of the mask and tubes. After the silicone has cured the mask can be removed leaving a gap between the base and the silicone. Now the silicone has to be cut to a flat surface. A sharp, flat wire cutter or nail scissors can be used for this. The imprint of the mask's overhang serves as a height reference. The tubes also get cut to the same level.

Molding the second layer of silicone for the final cell

Residues of release agent left by the mask on the side of the cured silicone cylinder can be removed with a small piece of paper towel soaked in cleaning alcohol. Now release

agent gets applied to the top surface of the cured silicone. The release agent we used was viscous enough to also cover the tube holes. If this is not the case small pieces of thin paper can be used to cover the tubes. The cell should now look like in Figure 3.9 (B). Now the second layer of silicone is poured in until it is flat with the top rim. It should fill the gap between the base and the cured silicone and cover the cured silicone in a 1 mm thick layer. After the second layer of silicone has cured the cell is finished and should look similar to Figure 3.9 (C).

Used materials

Similar to Gohlke et al. [2016], we used silicone with a hardness of 32 Shore A (TFC Silikon Kautschuk Typ 15) for the top layer. For the bottom layer, we used slightly harder silicone (Shore A 35, TFC Silikon Kautschuk Typ 2), as it was faster to cure and the hardness did not impact the performance of the cell. Our silicone tubes had an inner diameter of 2 mm and an outer diameter of 4 mm. As a release agent, we used a silicone mold separating cream (Glorex Vaseline Trennmittel). The base and the mask were printed with PLA plastic on an Ultimaker S5 Pro with a layer height of 0.1 mm. Support is not needed.

Bubblepad Assembly

Mainly limited by the cost of the solenoid valves we decided to have an array of ten cells for our prototype. With the method described above ten out of eleven manufactured cells turned out working, only one cell was leaky.

Cells are placed into a 3D printed frame

To combine all the cells into a single unit we designed and 3D printed a frame for the bubblepad in which the cells get inserted (see Figure 3.10 (A)). It keeps the cells in place and provides structural stability to the bubblepad. We also added four legs to the sides to offset the frame from the underground. The legs get screwed on with M3 bolts and nuts. To keep the surface soft and pliable we cut two sheets of 3 mm sponge rubber so that it fits precisely on the top surface of the frame and inside the compartment of the cells (see Figure 3.10 (B)). [2]

[2] A stencil can be found in the repository under

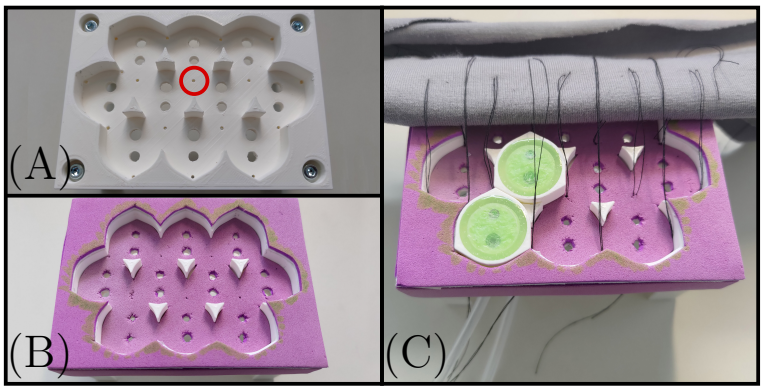


Figure 3.10: Shows three different stages of the bubblepad assembly process. (A) shows the empty 3D printed bubblepad frame. It has ten slots for the cells to be inserted in. Every slot is surrounded by three small holes (red circle) helping to sew the fabric to the frame. (B) shows the frame covered with two fittingly cut layers of sponger rubber. In (C) the threads that hold the fabric in place are already passed through the holes marked in (A). Now the cells can be inserted underneath the fabric.

The fabric is sewed to the frame in between the cells

Next, the covering fabric has to be sewed to the frame. This is a bit tricky as it has to be done before the cells are placed into the frame. Otherwise, the needle does not fit through the frame anymore. On the bottom side, the frame has holes located at the positions of the three corners of each cell (one of these holes is marked with a red circle in Figure 3.10 (A)) through which a thread can pass through. To make the seams in the right position the fabric is placed flat onto the frame. From underneath, a needle gets inserted orthogonally through one of the holes in the bottom of the frame, pulling the thread through the surface of the fabric. Then a millimeter away the needle pierces through the fabric again in the opposite direction and feeds back through the hole in the frame. The thread can now be cut. The two ends of the thread should both come out of the same hole at the bottom. It is important the thread is long enough so the fabric can later be pulled up to insert the cells without the thread completely slipping through the frame. See Figure 3.10 (C) for reference.

3d_models/bubblepad-frame_stencil.pdf.

The cells get inserted under the frame

Now the cells can be inserted into the frame under the fabric. To tighten the threads coming out of the bottom, the frame gets turned around. Each pair of threads gets pulled until it is under tension and then fixed to the bottom of the frame, e.g. with a drop of hot glue. At last, the edge of the fabric can be stretched out a little bit and glued to the edge of the frame. The bubblepad is now finished.

The used fabric should be elastic

We used two layers of fabric with a total thickness of about 1 mm. The fabric should not be much thicker as it dampens the tactile feeling of the cells as well as it impedes their clear visibility. To make it even possible that the inflated cells are noticeable under the fabric, the fabric has to be stretchable so that it can be deformed by the expanding cell.

3.4.2 Pneumatic Control System

A problem with solenoid valves is that they do not block pressure equally in both directions

Most of the pneumatic control system is already explained in section 3.3.2 “Pneumatic Control System”. However, one interesting aspect not covered so far is our choice of the solenoid valves. It was clear we needed direct-operated valves because of the low pressure differences in our system (see info box in section 3.3.2). However, the construction method of these valves poses a problem. In the closed state, they can typically handle a much greater pressure difference before leaking in one direction than in the other. This phenomenon can be easily understood by looking at how these valves are constructed. A schematic of a 2-way direct operated solenoid valve can be seen in Figure 3.11. When the pressure on the inlet side is greater than the pressure on the outlet side, the pressure difference assists the spring force pushing the plunger down. The valve only leaks if the pressure gets so high the materials fail. If it is the other way around, i.e. the pressure on the outlet side is higher than on the inlet side, the pressure difference counteracts the spring force. As soon as the pressure overcomes the spring force, the plunger gets pushed up and the medium can leak through the orifice. How much pressure a valve can handle in the sub-optimal direction depends on its construction.

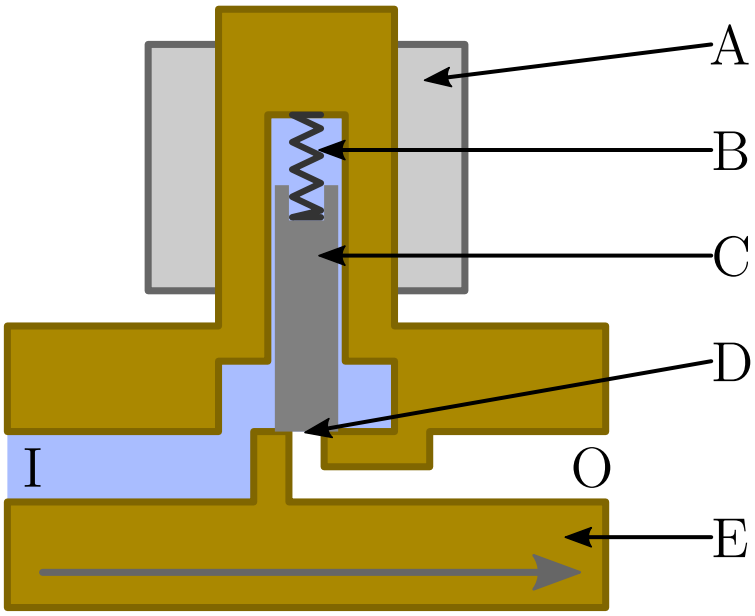


Figure 3.11: A 2D schematic showing the design of a 2-way solenoid valve. A: Electromagnet, B: Spring, C: Plunger, D: Orifice, E: Valve Body, I: Inlet, O: Outlet. The proper flow direction is from left to right. The valve is depicted in its normally closed state. The medium (blue) enters through the inlet but is blocked by the plunger and can not reach the outlet. The plunger is forced into this position by the spring. If the electromagnet gets powered, the magnetic force pulling up the plunger overcomes the spring force, and the plunger lifts. This allows the medium to flow through the orifice to the outlet.

We need solenoid valves working bidirectionally

However, we needed the valves to handle pressure differences with the higher pressure on both sides. Consider the following two situations a cell-valve can be in. When the cell is deflated the side of the cell-valve connecting to the mode-valve is under higher pressure (because the mode-valve connects to the air compressor). The cell-valve should not let air pass into the cell. When the cell is inflated the pressure on both sides of the cell-valve is equal. However, when a user presses on a cell, the pressure on the side of the cell-valve connecting to the cell is higher. This time the cell-valve should not let air pass out of the cell.

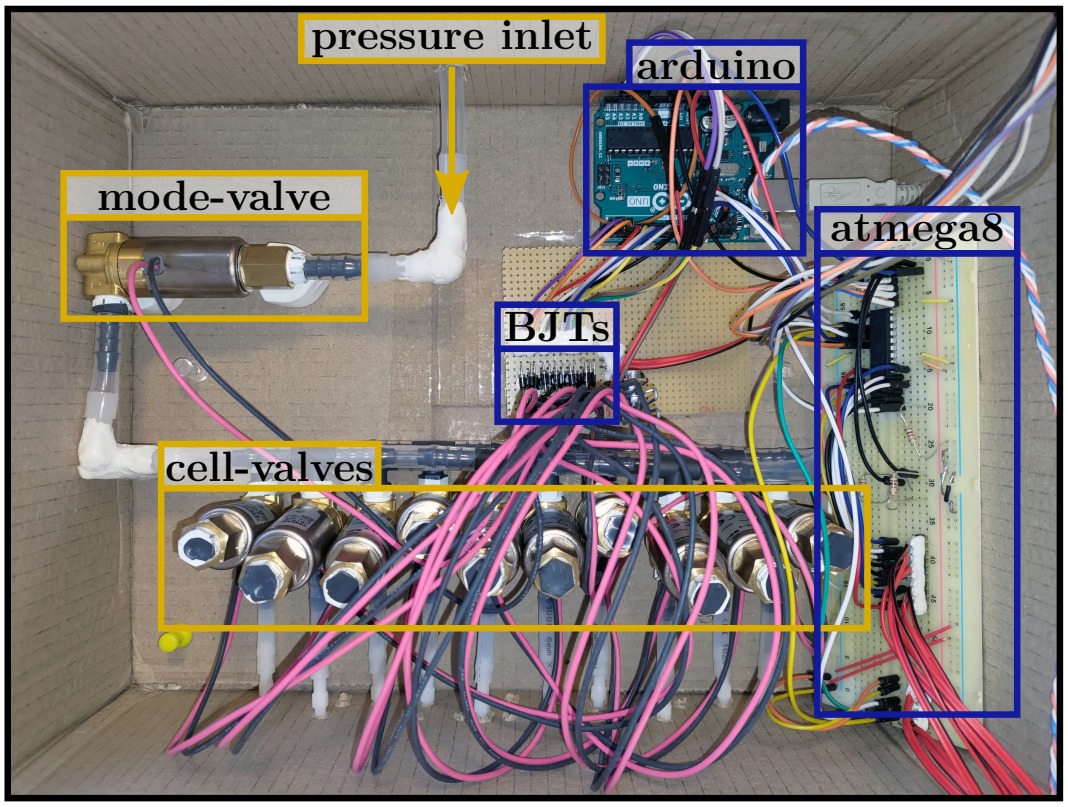


Figure 3.12: Shows the finished assembly of the pneumatic (orange) and electric (blue) control system and gives an overview of the most important components. The system was built into a cardboard box. The electric control system gets explained in the following chapter.

Our choice for the valves

As most direct acting 2-way or 3-way solenoid valves are designed this way to only work unidirectional manufacturers often do not state how much pressure contrary to the operating direction (back pressure) their valves can handle. In the end, we decided to use 3-way valves from SMC out of their old VDW product line (VDW2505G201FQ). They were relatively cheap ($\leq$ 20€) and it is explicitly stated they can handle at least one bar of pressure regardless of the flow direction. Another option we considered where STC 2V025 valves. They can be bought even cheaper ($\leq$ 10€) and the test unit we had could handle a back pressure of 2 bar. However, due to the COVID-19 pandemic, we could not order them and went with the former option.

The system gets assembled with silicone tubing and connectors

The assembly of the pneumatic control system is straightforward and mainly consists of connecting everything with tubes and connectors as described in section 3.3.2 "Pneumatic Control System". To connect all the cell-valves to the mode-valve we used threaded T-connection pieces and 6 mm (inner diameter) silicone tubing connecting them all in series. The cells get connected to the cell-valves with simple straight threaded connectors. Figure 3.12 depicts the finished assembly.

3.4.3 Electronics

Switching the Valves

The transistors switching the valves need to fulfill certain electrical requirements

To switch the electric current necessary to power the solenoid valves it is common practice to use transistors. We used so-called bipolar junction transistors (BJTs). In a nutshell, BJTs allow a small current I_B to switch a larger current I_C. The relation is described by the DC current gain $\beta_F = \frac{I_C}{I_B}$. Our valves were rated for a voltage of 24 V DC with a maximum power consumption of 3 W, meaning we had to supply a maximum current of $\frac{P}{U} = \frac{3\ \mathrm{W}}{24\ \mathrm{V}} = 125\ \mathrm{mA}$. To be on the safe side even when all eleven valves get powered simultaneously a single output pin of the Arduino should only deliver a maximum of 10 mA [Microchip Technology Inc. [2018] p. 323]. Thus the transistors have to feature a current gain of at least $\beta_F = \frac{125\ \mathrm{mA}}{10\ \mathrm{mA}} = 12.5$. In general, BJTs feature much higher current gains of at least 50. A more limiting factor is that the transistor itself has to be able to handle the current and voltage requirements of a valve. We have chosen the NPN BC337-40 BJT because it can handle a collector current I_C of 800 mA and a Collector-Emitter-Voltage of 45 V, easily fulfilling the 125 mA, 24 V requirement to drive a valve. In the worst case, they feature a current gain of $\beta_F \geq 170$.

A suitable value for the base resistor

The limit the current the Arduino supplies to the transistor a base resistor R_B is used. We chose a resistance of $R_B = 1\ \mathrm{k\Omega}$. As the Arduino operates with $V_{CC} = 5\ \mathrm{V}$, this allows a maximum current of $I_B \leq \frac{V_{CC}}{R_B} = \frac{5\ \mathrm{V}}{1\ \mathrm{k\Omega}} = 5\ \mathrm{mA}$ to flow which is less than the

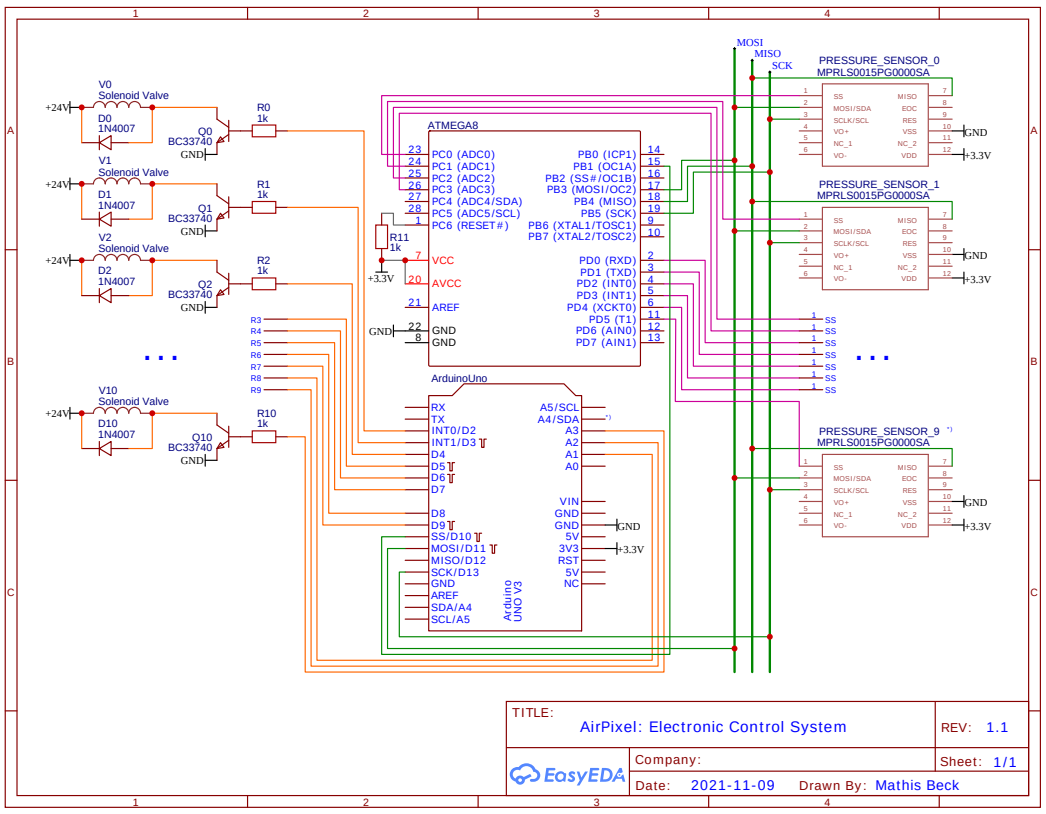


Figure 3.13: Circuit diagram of the electronic control system. The main components are an Arduino Uno and an atemga8 microcontroller (middle). The pressure sensors (right) and both microcontrollers are connected to the SPI bus (green). The atmega8 controls the SS pins of the pressure sensors (purple). The Arduino controls the BJTs switching the electromagnets inside valves (orange).

**) The 9th pressure sensor failed and had to be replaced with an analog one connected to the Arduino on pin A4.*

maximum 10 mA per output. However, there is a voltage drop of $V_{BE} \leq 1.2$ V. Thus the voltage dropping across the base resistor V_{R_B} is smaller than the 5 V V_{CC} from the Arduino: $V_{R_B} \geq V_{CC} - V_{BE} = 5\text{ V} - 1.2\text{ V} = 3.8\text{ V}$. Now the base current can be as small as $I_B \geq \frac{V_{R_B}}{R_B} = \frac{3.8\text{ V}}{1\text{ k}\Omega} = 3.8\text{ mA}$. But even this would allow the valves still to draw up to $I_C \geq \beta_F * I_B = 170 * 3.8\text{ mA} = 646\text{ mA}$ which is more than enough. The circuit diagram in Figure 3.13 shows how all the parts are exactly connected.

When the current flow through the valves stops, the coil of the electromagnet induces a large reverse voltage spike

(due to Faraday's law of induction the coil resists the current change). To avoid damage to the BJTs or Arduino a so-called flyback diode gets connected across the coil. In normal operation, this diode is not conductive. However, it becomes conductive when the reverse voltage is applied. This shorts the reverse voltage allowing the energy to dissipate through the resistance of the coil. We used 1N-4007 diodes.

To avoid the electromagnet damaging the transistors or microcontrollers a flyback diode is used.

Pressure Sensors

Besides controlling the valves, the electronic system also has to read the values from the pressure sensors. Even though pressure is an absolute physical quantity some pressure sensors measure the pressure relative to a specific pressure level. There are three kinds of pressure sensors commonly available. Absolute pressure sensors measure the absolute pressure value (so to speak, vacuum is the reference point), gauge pressure sensors measure the pressure relative to the ambient pressure, and differential pressure sensors measure the difference between two pressure levels. For our application, the used reference point is not important as long as it does not drastically change during operation. Other important parameters to consider when choosing pressure sensors are their operating range, accuracy, and output methods. Analog sensors output their reading as a voltage potential. This voltage has to be converted with an analog-digital-converter (ADC) to a digital value. This can be done e.g. with the ADC already integrated into the Arduino. However, the Arduino only features six ADC channels and we want to use ten pressure sensors. To avoid using an external ADC with more channels we used digital pressure sensors instead. They have their own integrated ADC and output the pressure directly as a digital value.

There are absolute, gauge, and differential pressure sensors outputting their data either in analog or digital form

In the end, we decided on Honeywell MPR sensors (MPRLS0015PG0000SA). These are gauge pressure sensors, have an operating range from 0 to 1 bar, and feature a high-resolution ADC. To read out the data they support both the I2C and SPI bus. Using I2C, to address multiple sensors of

We use digital gauge pressure sensors

the same type a separate address multiplexer is needed. To avoid this, we decided to go with the SPI option. A quick summary of the working principles of SPI can be found in the box below.

Definition: *SPI*

SPI:
SPI is short for Serial Peripheral Interface. It is a serial communication bus often used to handle communication between microcontrollers and peripheral devices or other microcontrollers. The attached devices are organized in a master-slave hierarchy. There is exactly one master on the bus controlling one or more slaves. All devices share the following three common wires:

- Master Out Slave In (MOSI). Data gets transmitted from the master device to a slave.
- Master In Slave Out (MISO). Data gets transmitted from a slave device to the master.
- Clock signal (SCK). A common clock signal created by the master device synchronizing the communication on the MISO and MOSI lines.

To specify which slave is communicating with the master, in addition to the three wires from above, every slave is connected to the master with a separate Slave Select (SS) wire (active low). The communication is always initiated by the master by pulling down the SS line of the desired slave and starting the clock on the CLK line. Now slave and master simultaneously transmit data over the MISO and MOSI lines until the master finishes the communication by pulling the SS line high and stopping the clock. Every slave not selected by the SS line ignores the CLK and MOSI lines and does not output anything on the MISO line. [Microchip [2018]]

The separate atmega8 microcontroller communicates with the pressure sensors on the required voltage level via SPI

The MPR pressure sensors we use operate at a voltage of 3.3 V. Though the Arduino operates at 5 V making a direct connection impossible. In a first attempt, we tried to use a voltage-level translator (TXB0104 from Texas Instruments). This chip could theoretically be used to translate between the 3.3 V logic level from the pressure sensors to the 5 V logic level from the Arduino. While this worked

partially the communication between the Arduino and the pressure sensor was highly unreliable. We are not entirely sure why this issue occurred. We suspect that either the voltage-level translator messed up the timing of the SPI bus or the pressure sensors do have not enough power to drive the inputs of the translator. Nevertheless, we bypassed this problem by using a separate atmega8 microcontroller running at 3.3 V. The atmega8 can directly communicate with the pressure sensors.

The atmega8 sends the collected pressure data via SPI to the Arduino

The transfer of the data from the atmega8 to the Arduino is also done via SPI. The difference that allows an SPI communication despite the voltage level difference, in this case, is that the communication between the Arduino and atmega8 is only unidirectional. The atmega8 is configured as the SPI master, sending the pressure values to the Arduino whenever they are available. Communication in this direction is not a problem. When the atmega8 sends a logical 1 corresponding to 3.3 V on the MOSI line, the Arduino interprets this correctly, because everything above 3.0 V is considered a logical one by the Arduino. Arduino and atmega8 both use 0 V to represent a logical 0, so there is also no problem in this case. Communication in the other direction from the 5 V device to the 3.3 V device would be an issue because sending a logical 1 with 5 V would destroy the input of the 3.3 V device. As mentioned this direction is not needed to send data from the atmega8 to the Arduino, however, it would be needed for the pressure sensors, because the master has to send a special command to them, to start the pressure reading.

3.4.4 Data Transmission Protocol

Data is grouped into frames before transmission

For the transmission of the pressure values from the atmega8 to the Arduino and from the Arduino to the PC, a simple protocol is used. The data from all pressure sensors gets grouped into one data frame. Every frame starts with the control byte 0x01 and ends with the control byte 0x02. To distinguish control from data bytes, 0xFF is used as an escape byte prepending every control byte. Thus 0xFF is not allowed in the payload, however, this does not pose

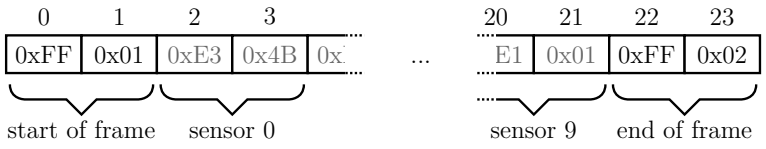


Figure 3.14: An exemplary data frame with a length of 24 bytes containing the pressure values from all ten pressure sensors. The first two bytes mark the beginning of the frame. Then next 10 byte-pairs (byte 2 to 21) are the pressure values. The last two bytes mark the end of the frame. The byte 0xFF serves as an escape byte.

any problems. The pressure values get sent as a two-byte number. The first, most significant byte can never reach 0xFF because the maximum output is 0xE666. To avoid the other byte of being 0xFF the least significant bit is always set to zero. In theory, this alters the pressure reading, however, the last bit is so insignificant it is anyway totally overshadowed by noise. Figure 3.14 illustrates how such a data frame can look like.

The receiver extracts the data from the frames

To split an incoming data stream into individual frames the receiving program remembers the current state of the protocol. The state can either be WAIT, COMMAND, or READ. WAIT is the initial state and indicates that the program does not know where we currently are in the data stream. Thus the program waits until it receives the start of frame command (0x01). In COMMAND mode the last received byte was the 0xFF escape byte, meaning the upcoming byte is a special command. READ means that a data frame gets received. In this mode, the program keeps track of how many bytes of the current frame were already received. When receiving a new byte, first it gets checked whether it is the 0xFF escape byte. If so the program switches to the COMMAND state. If not the action depends on the current state. In WAIT mode nothing happens, the received byte gets discarded. In COMMAND mode the received byte is interpreted as a command. When receiving 0x01 in COMMAND mode (start of frame), the program switches to READ mode and resets the number of received bytes. When receiving 0x02 in COMMAND mode (end of frame), the program switches to WAIT mode.

Chapter 4

User Study and Discussion

Because of the novel visual and tactile appearance of our user interface we wanted to study reactions from users to our prototype and gather feedback about the concept. Gohlke et al. [2016] already conducted a user study evaluating their single pneumatic button. This study primarily focused on pressure levels and the tactile feedback they generated. Harrison and Hudson [2009] also conducted a study of their inflatable display, comparing it to other user interfaces. Although these studies provide some hints on how our prototype will perform, our design is too different to extrapolate from the available data to how users will interact with it. Thus, we conducted our own study to find out what are the problems and strengths of our design and where should be the focus when further developing the concept. Note that in this chapter whenever we speak of 'the prototype' or 'the device' in a context addressing the participants, we actually mean only the bubblepad. As this is the only part of the prototype the participants interacted with, we did not want to introduce a new separate term to them.

Results of other studies from related work are only marginally applicable to our prototype

4.1 Research Goals

We want to find out how do users interact with the prototype and what problems there are with the current implementation

One of the two main goals of the study was to get insights into how users perceive the interaction with the prototype. This includes the following qualitative questions: Is the bubblepad even perceived as an interface? Is the interaction intuitive? Does it pose a problem that the UI can change all of a sudden? What kind of a conceptual model do users have in their minds? What kind of UIs rendered on the bubblepad are users imagining? The other main goal is to find out what users think about the concept in general and specifically our implementation. What are the problems of our implementation? How can the current prototype be improved? Is the prototype suitable for the sofa-use-case described in the introduction?

4.2 Experimental Design

The study consists of two parts

The study can be split into two parts: In the first part, the participants solve a sequence of tasks shown on a computer screen with the help of the bubblepad. The participant is not told beforehand how the bubblepad works or how to interact with it. The second part consists of a semi-structured interview.

4.2.1 Tasks

First part of the study: solve five tasks displayed on the computer screen using the bubblepad

There are in total five tasks. The first four tasks have a simple goal always described at the top of the screen (see Figure 4.1). Every participant had to complete 8 repetitions of these tasks. The rendered interface on the bubblepad stayed constant during the repetitions and only changed in between the tasks. The order of the first four tasks was randomized between the participants to mitigate learning effects. The fifth task was more complex than the first four and was split into five sub-tasks. All participants completed this task last, so they could use the experience they gathered when completing the first four tasks. Also, the

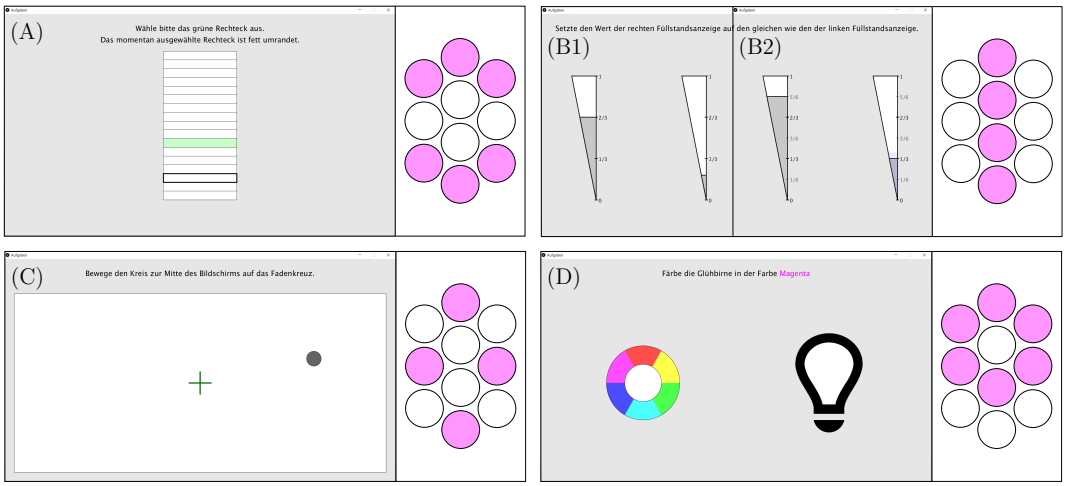


Figure 4.1: Screenshots from the tasks used in the first part of the study. Next to the tasks the cell pattern that was rendered on the bubblepad for this task is depicted. (A) UpDown Task. (B) 4Levels Task. (C) DPad Task. (D) Pie Task.

interface did not remain constant during this fifth task but was adapted after each interaction.

The following list explains the first four tasks. The letters after the task names relate to Figure 4.1.

- UpDown (A): The participant had to select the green rectangle. To move the selection two areas were inflated on the top and bottom of the bubblepad. The areas should resemble up and down arrows. By pressing a cell of the upper area the selection moves upwards and vice versa.
- 4Levels (B): The participant had to adjust the level of the right level indicator so it matches with the left level indicator. The level indicators have four equally spaced levels (0, $\frac{1}{3}$, $\frac{2}{3}$, 1). Four cells in a vertical line get inflated on the bubblepad. Every cell corresponds to one level of the level indicator, i.e. pressing the topmost cell completely fills the level indicator, the cell one below sets the level indicator to $\frac{2}{3}$, and so on. Starting from the fifth repetition the level indicators got expanded by three more in between levels ($\frac{1}{6}$, $\frac{3}{6}$, $\frac{5}{6}$) (B2). However, the cell pattern on the bubble pad did not change. To reach one of the in-between lev-

els the participant had to press the cell corresponding to the level below with greater pressure. To suggest that pressure now influences the tasks, the level indicator also resembled the currently applied pressure by showing a light blue overlay.

- DPad (C): The participant had to move a circle on a 2D surface to the middle of the screen. The four outer cells on the bubblepad in the middle of each side get inflated. The top cell moves the circle up, the left cell moves it to the left, and so on. The speed of the movement is determined by how much pressure gets applied to the cells. Harder pressure results in the circle moving faster. It was also possible to press multiple cells at once to move the circle diagonally.
- Pie (D): The participant had to select the right color on the color wheel. A ring of six cells gets inflated on the bubblepad. There is a 1:1 spatial mapping between the colors on the color wheel and the cells on the bubblepad.

The fifth task is more complex and the interface changes

The fifth task (named Rooms) should resemble a simplified smart home control interface. The participant had to navigate through a three-layer menu and change specific parameters. The active layer is highlighted by a white background. The procedure how to navigate through this task is explained in the following list with the help of Figure 4.2:

(A) A room can be selected with the top and bottom cells. Pressing the top cell moves the selection up and vice versa. The right cell confirms the selection and proceeds to state (B).

(B) A context menu pops up for the selected room. The middle vertical row of cells is used to choose an option from the menu. The left cell closes the menu and goes back to (A). Choosing the first entry 'Bearbeiten' (eng. edit) leads to (C). Choosing the second entry 'Kopieren' (eng. copy) copies the settings from the selected room and goes back to (A). The copied settings can be applied to another room as shown in (G). The third option 'Einfügen' (eng. paste) can not be

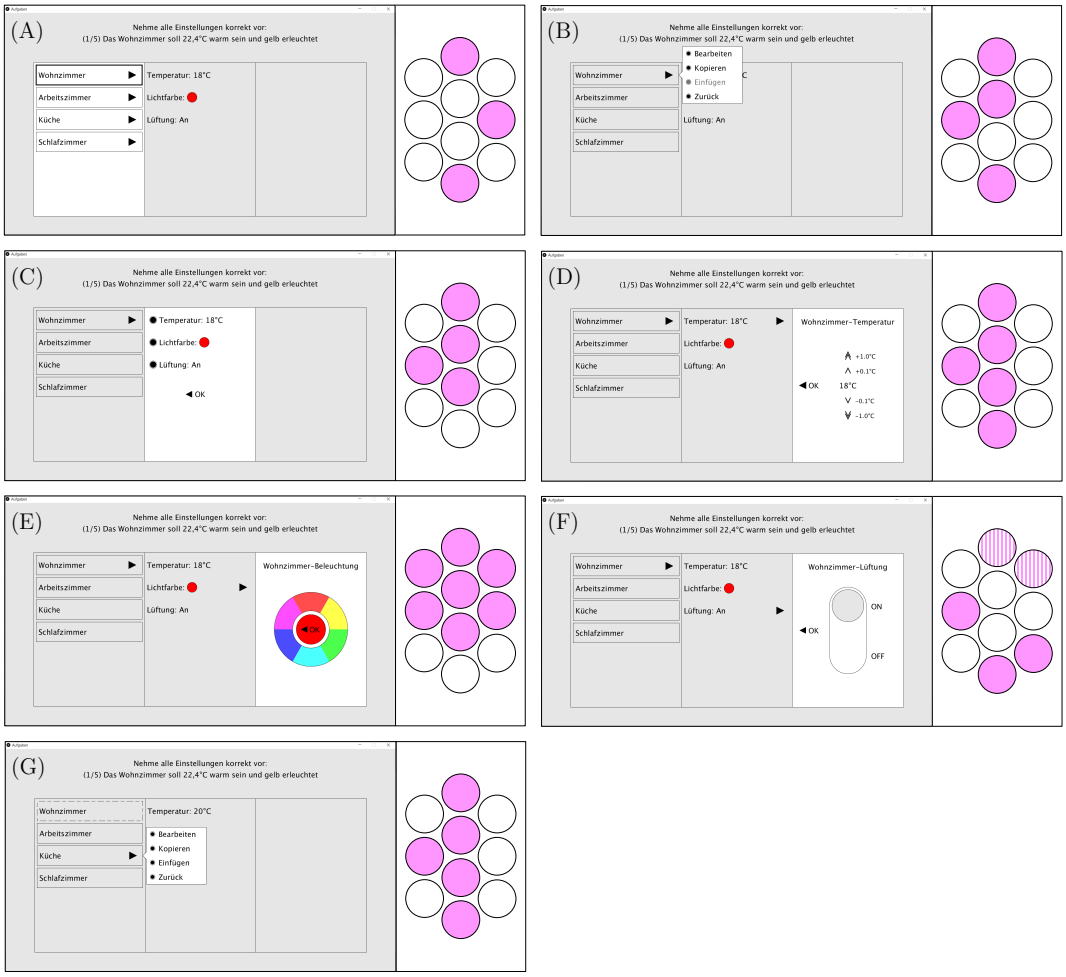


Figure 4.2: Shows all the different interfaces states of the Rooms task together with the associated cell patterns. The task consists of a three-layer menu that should mimic a typical smart home control interface. In the first layer (A), a room gets selected. Through a context menu (B, G) the second layer (C) gets accessed. There it can be chosen which parameter (temperature, light color, or ventilation) of the selected room the user wants to edit. The third layer (D, E, F) provides specific controls to edit the chosen parameter.

chosen because no settings were copied before. The last option 'Zurück' (eng. back) is another way of just closing the menu.

(C) Shows the current values for the three parameters 'Temperatur' (eng. temperature), 'Lichtfarbe' (eng. lighting color), and 'Lüftung' (eng. ventilation) and lets the user choose which one to edit. Pressing the

left cell leads back to (A). The three cells in the middle vertical row select one of the three parameters that should get changed (leading to (D, E, or F)).

(D) Controls for editing the temperature. The left cell confirms the current values and leads back to (C). The four cells in the middle vertical row incrementally increase or decrease the temperature.

(E) Controls for editing the lighting color. The outer ring changes the color similar to the Pie task. The middle cell confirms the current color and leads back to (C).

(F) Controls for turning the ventilation on or off. The left cell confirms the current state and leads back to (C). The bottom two cells turn off the ventilation, the top two turn it. They are only inflated in the opposite state.

(G) Like in (B) the context menu is shown. However, the settings from 'Wohnzimmer' (eng. living room) were copied before and could now be applied to 'Küche' (eng. kitchen). Note that now all four cells in the middle vertical row are inflated because all options are available.

To complete the task a participant had to complete five subtasks described on the top of the screen. The first three subtasks required the participant to set parameters of specific rooms to specific values. The fourth and five asked the participant to copy all the parameter values from one room to another room. (1/5: 'Wohnzimmer', 22.4 °C, yellow lights. 2/5: 'Küche', ventilation off. 3/5: 'Arbeitszimmer' (eng. study), 18.0 °C, red lights, ventilation on. 4/5: 'Schlafzimmer' (eng. bed room) should equal 'Arbeitszimmer'. 5/5: 'Wohnzimmer' should equal 'Küche').

4.2.2 Interview With High-Level Tasks

Second part of the study: participants were interviewed and developed their own cell patterns

After the participants completed the tasks a semi-structured interview was conducted. First, the participants were briefly explained how the bubblepad works. Then

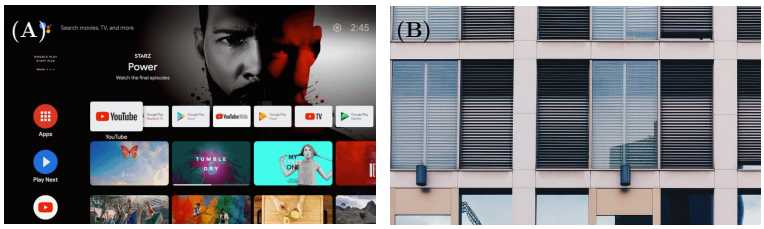


Figure 4.3: The images were used to visualize two of the high-level tasks used in the second part of the study. The participants were shown these images. (A) should represent a typically smart-tv interface. (B) was used to describe electric window blinds.

they should rate certain statements on a Likert scale from 1 ”strongly disagree” to 5 ”strongly agree”. The first question was whether ”it was fast and easy to understand how to use the prototype to solve the tasks”. Then they were asked individually for each task whether ”the prototype helped you to solve the task”. About the Rooms tasks, they were additionally asked whether it confused them that the interface changed while they interacted with the prototype. Further questions focused on the participant’s experience during the interaction, what use-cases for the prototype participants could think of, what they liked and disliked about the bubblepad, what could be changed about or added to the prototype, and whether they think the sofa-use-case is practical. A more complete outline of the interview can be found in appendix A.

Four high-level tasks were introduced for which the participants should develop cell patterns

Part of the interview was also that the participants were asked to develop their own interfaces on the bubblepad. To do this four high-level imaginary tasks were presented to the participants and it was asked how the cell pattern on the bubblepad should look like to solve these tasks. The participants were given a template on a sheet of paper where they could draw the cell patterns they came up with. The template can be found in appendix A. It was explicitly not part of the task to design an interface to bring the bubblepad in the correct state to solve the actual task. It was assumed that the bubblepad magically knows what the current task is and what the user wants to do. The four high-level tasks were described as follows:

- Imagine you are sitting on a sofa with a modern TV in front of you. Next to you, the bubblepad is integrated into the armrest of the sofa. You want to change the volume of the TV.
- Imagine now you want to navigate through this smart-tv interface shown in Figure 4.3 (A).
- Imagine there is a smart lighting system in the room with LED lamps that can be changed in color and brightness. You want to change both of these parameters.
- Imagine similar electric window blinds as shown in Figure 4.3 (B). You want to control the height and the opening angle of the slats.

4.3 Participants and Language

Ten participants went through the study in either English or German

The study was conducted with ten participants with an age between 20 and 30 years. None of them had a background in computer science or HCI and it was their first time encountering the prototype. However, all of them had used a smart-tv before. The study was conducted in either English or German based on the participant's preference. The introduction and interview were translated into both languages. The consent form was only available in English. The descriptions shown on the screen for the tasks were only displayed in German, however, the investigator translated them into English when necessary. Six participants preferred to do the study in German, four in English.

4.4 Variables and Recordings

There are two independent variables. In the first part of the study the five tasks together with the cell patterns and in the second part the different imaginary high-level tasks. The tasks and cell patterns were both controlled by the

desktop application. The investigator guided through the high-level tasks.

The software recorded pressure values and timings during the study

During the study, several dependent variables were measured. All inputs to the bubblepad were recorded, i.e. all pressure values from the sensors were stored. Also the time it took a participant to complete a task was recorded. For the DPad Task, the applied pressure and distance to the target were continuously recorded. The recording of all the values was handled by the desktop application. Time intervals were recorded in ms. Pressure values were normalized in relation to the maximum cell pressure (pressing down a cell completely).

4.5 Procedure

Participants gave their informed consent

The study is conducted in person with one investigator and one participant at the time. During the whole study, the participant sits in front of a desk with a computer screen. Frist the participants got a short introduction making them aware of the purpose of the study, the procedure they pass through, the involved risks, and how their data will be handled. They were asked to sign a consent form presenting the same information to them in text form (for the consent form see appendix A).

The investigator guided the participants through the process

Next, the setup was explained. On the desk to the right of the participant, the bubblepad was situated. They were told that this is the interface they have to use to solve the tasks. No other input method was allowed. The tasks will be shown on the screen. When ready the participant pressed enter on the keyboard to start the first task. To proceed to the next task or repetition after completion the participant also had to press the enter key. After all tasks were completed the interview was conducted. For each high-level task, the investigator provided a separate template to the participant to draw the cell patterns on.

4.6 Results

4.6.1 The Bubblepad Was Recognized as an Interface

Starting with one of the fundamental research questions we asked at the beginning of this chapter, it is safe to say that users indeed recognized the bubblepad as an interface. All of the participants solved all the tasks. The tasks were also solved in an adequate amount of time (more about the completion times below), showing that the participants not just completed the tasks by random inputs but had a clear vision of what they are doing. Nine out of ten participants also immediately associated the inflated cells with some kind of button that can be pressed to invoke an action.

4.6.2 The Interaction Was Intuitive

On average participants agreed to the statement that "it was fast and easy to understand how to use the prototype to solve the tasks". On the Likert scale, the average rating was 4.1 out of 5 (SD 0.7) as depicted in Figure 4.4. Based on the participants' statements, it also became clear that the association between the cell pattern on the bubblepad and the graphical interface on the screen worked well. In addition, the learning curves of the tasks were quite steep. While on average the first iteration of each task took in comparison rather long to complete (up to three times as long as for later repetitions), already the second iteration was completed much faster, see Figure 4.5. All in all, it seemed like the interaction with the bubblepad was experienced quite intuitive which is also consistent with what participants stated in the interview. E.g. one participant said "[Die Interaktion] war sehr intuitiv. Ich konnte wirklich ganz einfach Zusammenhänge erkennen zwischen dem Muster auf dem Bildschirm und dem Muster auf dem [bubblepad]. Das war sehr einleuchtend." (eng. *The interaction was very intuitive. I could easily associate the pattern on the screen with the pattern on the [bubblepad]. That was very obvious*).

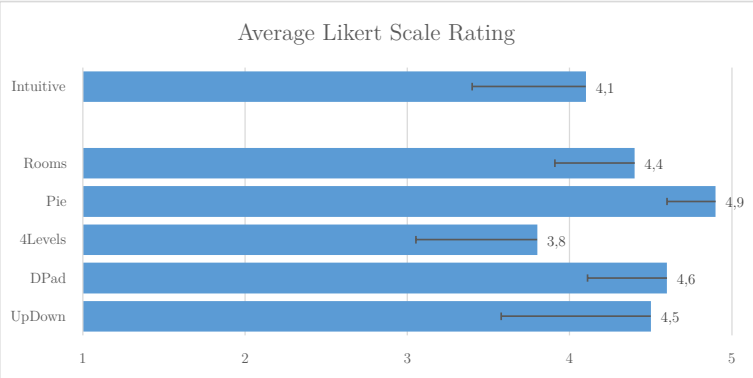


Figure 4.4: The diagram shows the average ratings on a Likert scale from all participants. The scale reaches from 1-'strongly disagree' to 5-'strongly agree'. The first dataset labeled 'Intuitive' refers to the statement "it was fast and easy to understand how to use the prototype to solve the tasks". The five datasets below refer to the question of whether "the prototype helped [the participant] to solve the task". The error bars indicate the standard deviation.

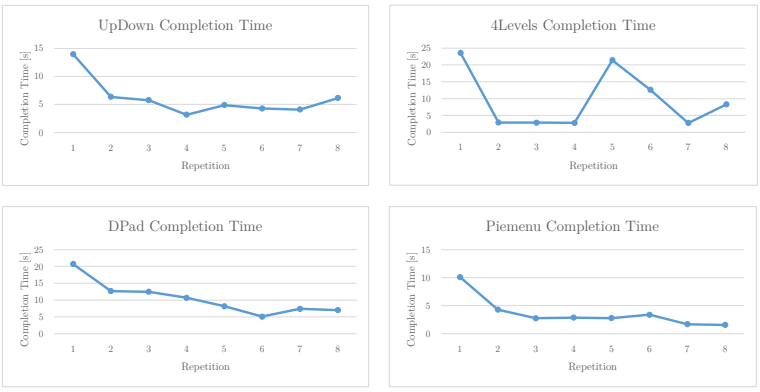


Figure 4.5: Each diagram shows the average completion times of a task for all repetitions. For all tasks, the completion time drops drastically after the first repetition. The 4Levels task shows an anomaly at the fifth repetition. The completion time increases again to the level of the first repetition. This can be attributed to the task changing in the fifth repetition (in between levels have to be selected).

4.6.3 Combining Cells Did Not Work

In the UpDown task, we tried to combine multiple neighboring cells to form larger buttons with a distinctive shape. The upper three cells should resemble an arrow pointing up and the lower three an arrow pointing down (see cell pattern in Figure 4.1 (A)).

However non of the participants understood the concept of multiple cells joined together. They only pressed individual cells to complete the task. When directly asked why there are multiple cells inflated the participants either said they expected the cells on the sides to somehow move the selection diagonally or they could not think of a reason and said it confused them. One participant even rated the task with a 2 on the Likert scale because he was very confused and could not figure out why there are multiple inflated cells for the same action. However, he very confidently solved the task and stated that apart from this confusion he would give a rating of 5.

It got very clear that the participants associated the cells on the bubblepad with individual round buttons that could either be active or not. Also when developing the cell patterns for the high-level tasks in the interview, none of the participants suggested a cell pattern using multiple cells to form a single button.

A reason for this could be the visual separation of the cells. Due to the way how the fabric is attached to the bubblepad, even neighboring cells are visually separated, as the fabric between the cells is fixed to the base frame of the bubblepad. Thus it is not possible to display a continuous shape by inflating neighboring cells (the issue and a possible solution are discussed in more detail below in section 4.7.1).

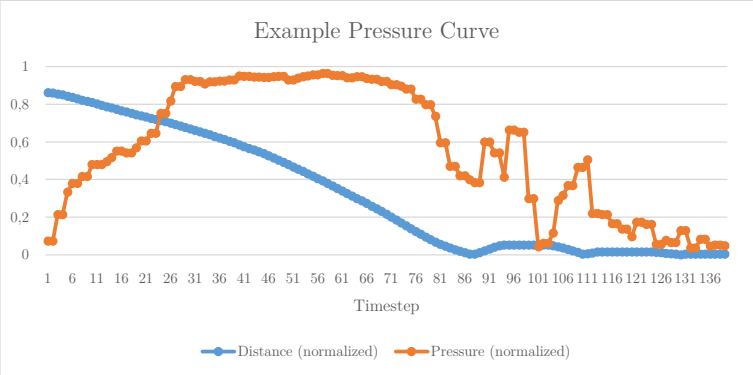


Figure 4.6: Pressure and distance curve over time from the DPad task. The eighth repetition from one of the participants is shown. This example shows particularly well that the participant decreased the pressure based on the distance to the target. Both pressure and distance are normalized (in relation to the maximum cell pressure and the maximum distance possible on the screen).

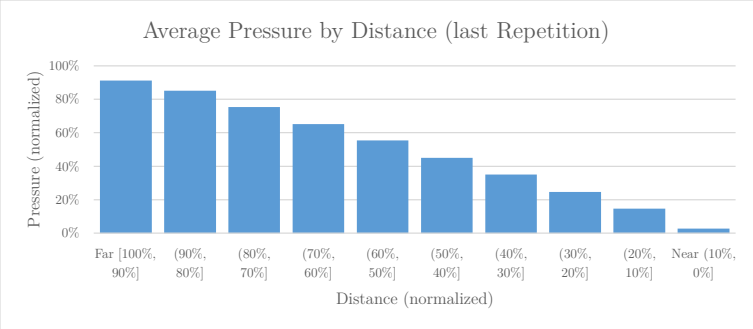


Figure 4.7: Diagram of the pressure distribution for the DPad task. Averaging over the last repetitions from all participants the pressure distribution in relation to the distance to the target is shown. The distance is categorized into ten segments of equal size. For each segment, the average of all pressure values was calculated that were recorded with a distance falling into the segment. The last repetition was chosen because the participants already learned how to solve the task efficiently at this point. It clearly can be seen that the applied pressure decreases when the distance gets smaller. Like in Figure 4.6, pressure and distance are normalized.

4.6.4 Pressure Input Helps Users

In the DPad task, the pressure of the cells determined how fast the circle moves on the screen (higher pressure means faster movement). At least eight of ten participants adapted their pressure during the task and used less pressure when the distance to the target got smaller. This behavior is illustrated in Figure 4.6 with the last repetition from one exemplary participant. Figure 4.7 visualizes the relation between distance and pressure averaging over all participants. Interestingly five of the eight participants mentioned above could later not recall that pressure had an influence on the task. This indicates that using pressure input was intuitive for the participants and helped them to solve the task.

4.6.5 Complex, Changing Interfaces Are Possible

The Rooms task provided a complex interface with multiple layers and a variety of UI elements. The cell pattern on the bubblepad constantly adapted to the state of the interface. Despite the increased complexity, it did not seem that participants had difficulties solving this task. Also, the rating of the Rooms task does not stick out worse in comparison to the other tasks (see Figure 4.4). The fact the cell pattern changed during the interaction did not seem to pose any problems for the participants. When specifically asked all participants described the changing cell patterns as more helpful than confusing.

At first sight, this contradicts the finding of Follmer et al. [2013] that ”rapid shape transitions were jarring to users”. Reasons for these different observations could be that our shape transition happens on a much smaller scale. Or that the participants started to expect the shape transition after they understood the concept that their current action not only changed the graphical interface but also the interface on the bubblepad adapting to the new state of the graphical interface.

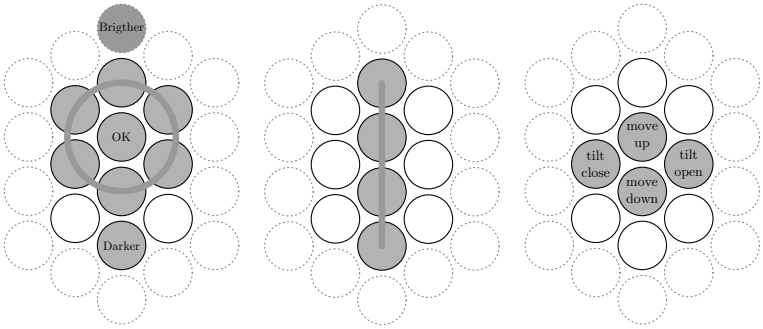


Figure 4.8: Three representative cell patterns participants suggested to solve the high-level tasks. Pattern (A) sets the color and brightness of a LED lamp. This or similar patterns were suggested by five participants. Six cells in the middle form a color wheel similar to the Pie task where every cell is associated with one color. The two discrete buttons on the top and bottom incrementally increase or decrease the brightness. The OK button confirms the selection. Three of the six participants suggested implementing the color wheel as some kind of slider to be able to select from a continuous spectrum of colors. Pattern (B) was suggested by one participant to change the TVs volume. A sliding gesture over the cells from top to bottom decreases the volume by a certain amount, sliding in the other direction increases the volume. To adjust the electric window blinds almost all participants suggested an interface similar to pattern (C). Two buttons move the blinds up and down. Two buttons tilt the slats. A summary of all suggested patterns can be found in appendix A.

4.6.6 Sliders Would Be Beneficial

When the participants were asked to suggest cell patterns for the high-level tasks four suggestions involved the usage of sliders. To set the color of a LED lamp five participants suggested using some sort of color wheel similar to the Pie task. Three of them also said that it would be beneficial if the bubblepad could continuously track the finger position also between the cells so that not only six fixed colors can be selected but also all the colors in between on the color

circle. A representative cell pattern suggested by these five participants is depicted in Figure 4.8 (A). A similar suggestion came up for changing the TVs volume. One participant suggested inflating cells in a vertical row and using gestures. Sliding up over the inflated cells would increase the volume and sliding downwards would decrease it (see Figure 4.8 B).

Apart from the sliders, the other suggested patterns did not reveal any interesting or unexpected interaction techniques. Most of them were just an arrangement of buttons serving a single purpose. An example can be seen in Figure 4.8 (C).

4.6.7 Visual Clues

When asked how the prototype could be improved six participants suggested improving the visual feedback. Suggestions were to label the cells, place an LED under each cell lighting it up, and create buttons with different shapes or sizes. At the moment the only visual output the bubblepad offers are the inflation states of the cells. Inflated cells can only be distinguished by their relative or absolute position on the bubblepad. It seemed like the participants used the relative position of the cells to identify the overall shape of the UI and associate it to the graphical interface on the screen. The absolute cell position played a role in consecutive cell patterns. E.g. in the Rooms task, the 'OK' buttons were always placed at the same absolute position on the bubblepad with one exception (when selecting the light color). Three participants had significant problems adapting to the altered position of the 'OK' button.

4.6.8 The Sofa-Use-Case

In general, the participants thought the concept of the bubblepad is appropriate to be integrated into a sofa. However, there were both positive and negative aspects participants brought up when using the bubblepad in this scenario.

+ Nine out of ten participants think the prototype is fitting the sofa-use-case

+ Eight out of ten participants liked the feeling of the surface and would not change it even when integrated into a sofa

+ Six out of ten participants said an advantage against a normal TV remote control is the easier usage because the bubblepad has fewer buttons or because the interface can adept

$\sim$ Four out of ten participants think it is a disadvantage that the device is stationary and can not be moved around. However, four participants as well said this is an advantage in comparison to a remote control because it can not get lost

- Six out of ten participants were worried about accidental inputs

- One participant disliked that only one person on the sofa can use the device

4.7 Discussion and Improvements

Based on the user study and the experience we gained when building the prototype we identified a few problems with the current design. In the following section, we address these problems and suggest possible solutions.

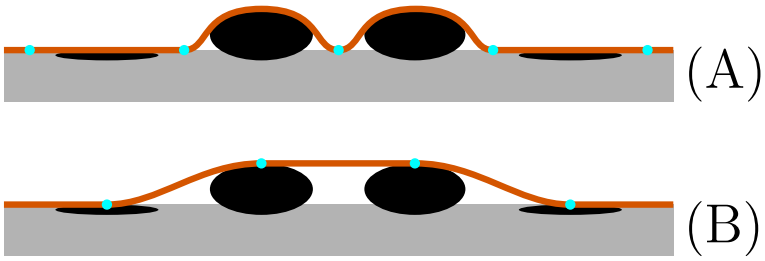


Figure 4.9: A schematic cross-section of the bubblepad illustrating two different ways of attaching the sheet of fabric. Grey: bubblepad frame; Black: cells; Orange: fabric; Cyan: anchor points. The two inner cells are inflated. (A) depicts the current implementation. The fabric is fixed to the frame between the cells. The contour between the cells is sharp, but two neighboring cells do not form a continuous shape. In (B) the fabric is fixed directly to the cells. This allows two neighboring cells to form a consistent shape but the contour is less sharp.

4.7.1 Fabric Attachment

A different way of attaching the fabric

As described in section 3.4.1 "Bubblepad Assembly" the sheet of fabric covering the bubblepad is fixed in between the cells to the bubblepad frame (see Figure 4.9 (A)). However, there might be a better way to attach the fabric. It could be directly attached to the top layer of silicone of each cell as depicted in Figure 4.9 (B). Maybe glue or a drop of uncured silicone can be used for the bonding. This could solve the following two problems at once.

Continuous shapes with multiple cells

Initially, we thought it would be possible to combine multiple cells to a larger button with a distinct shape. We tried this e.g. with the cell pattern for the UpDown task (see section 4.2.1). However, in the study, it turned out that participants do not recognize neighboring cells as a single button (see section 4.6.3). Funnily enough in the study participants even suggested having buttons with different shapes and sizes. When using the alternative way of attaching the fabric the contour of neighboring cells would form a continuous shape. This could lead to better recognition of buttons composed out of multiple cells. Having differently shaped

buttons would be one way to improve the visual feedback of the prototype.

Inputs from pressing in between cells

Secondly, the redesign could help to detect presses in between cells. When pressing in between two inflated cells in the current design, one basically just presses to the bubblepad frame without generating any input. However, with this alternative design, pressing in between two inflated cells would cause the fabric to stretch and exert a force on the two cells. A sophisticated enough algorithm could detect these kinds of in-between inputs. This could make sliding interaction as suggested by the participants in 4.6.6 "Sliders Would Be Beneficial" possible.

4.7.2 LEDs

Inflated cells can be difficult to see and distinguish

Even with the new way of attaching the fabric the visual feedback of the bubblepad has another problem. It is dependent on the lighting of the environment. When directional light falls onto the bubblepad from an angle the inflated cells cast a shadow and are easily visible. However when the light is very diffuse, it hits the bubblepad from the top, or it is just dark, inflated cells cast a weaker shadow making them hard to see. In the study, it was suggested to illuminate inflated cells with LEDs from below. In our opinion, this could be a good solution to the problem that even could be easily implemented with only a minor redesign of the cells.

A LED illuminating each cell could improve visibility

When using transparent silicone the LED could be directly cast into the bottom layer of the silicone similar to the tubes. The covering fabric of course would also have to be semi-transparent. When using the right fabric it could even be possible to use RGB LEDs and color the cells differently. This would improve the visual feedback even further. E.g., cells could be grouped by illuminating them in the same color.

4.7.3 Cell Resolution

Higher cell resolution has advantages but users might not prefer smaller cells

A higher cell resolution could both help to render more detailed composed button shapes and to sens user input in between the cells as more fine detailed data is available. A higher resolution could be achieved by decreasing the cell size. However, this complicates the production of the cells. Based on the production method we used, we are currently near the limit of what we could achieve regarding miniaturization without using more sophisticated tools. Additionally, eight out of the ten participants from the study stated they liked the current cell size. As a cell approximately fits the size of a fingertip, they would not make the cells smaller. Though the participants put a single cell on a level with a single button. Having smaller cells would allow joining multiple cells together to a larger button, so it is questionable if smaller cells would impair the user experience or not.

Decreasing the cell spacing

Anyway, removing the 3D printed cell frame could be another method of slightly increasing the cell resolution by decreasing the 'dead space' between the cells. The size of the inflating part of the cell would stay the same. The frame of the cells only is there to provide structural stability. However, the bubblepad anyways is designed to be pliable. We observed that the first layer of silicone could provide enough stability even without the cell frame especially when using a harder silicone for the first layer. By applying the release agent also to the cell frame before pouring in the silicone, the frame could just be removed later and the silicone could directly be placed into the bubblepad frame. This would both decrease the distance between the cells and improve the surface texture.

4.7.4 Operating Noises

One drawback that quickly comes to one's mind when using the bubblepad is the operating noise caused by the air compressor and the valves. However, if necessary these noises can be largely eliminated by using small air pumps and soundproof containers similar to Gohlke et al. [2016].

4.7.5 Design Goals

We fully reached our goals of the prototype responding in real-time and having a soft surface

Looking back we reached the design goals we mentioned earlier in section 3.1, although there is still room for improvement. When using the bubblepad there is no noticeable delay between the press on a cell and the desktop app recognizing the input. Even though changing a cell pattern takes a little more time, the system still feels responsive even if the pattern gets changed. Thus we reached our goal of the UI adapting in real-time. As mentioned above the participants of the study liked the surface texture and feel of the bubblepad. So the goal of having a soft and pliable surface we consider also as fulfilled.

We also reached our goals of detecting user input, providing visual and haptic output, and rendering arbitrary shapes. But there is more room for improvement

Another goal was to be able to detect user input. While the pressure measurement inside a cell worked well to detect presses to the cell, we can not detect any input between the cells. As mentioned this makes sliding inputs difficult to capture. We also wanted our prototype to render arbitrary shapes. The rendered cell pattern can be arbitrary, however the fact that combining multiple inflated cells did not work limits the number of possible shapes that can be rendered. The last goal was to provide both visual and haptic output. Inflated cells can both be seen and felt on the bubblepad. But as mentioned above the visibility depends on the lighting conditions.

Chapter 5

Summary and Future Work

5.1 Summary and Contributions

This work contributes a prototype of a TUI that has a soft and pliable surface and can dynamically change its appearance and the rendered UI.

Related work

First, we reviewed related work about shape displays and soft robotic principles in HCI research. Even though the work from Follmer et al. [2013] is over eight years old, the state of the art regarding shape displays has not developed much further in comparison to their inFORM shape display. Using principles from soft robotics Gohlke et al. [2016] created an inflatable, soft button that has both input and output capabilities. Based on this we focused on approaches with pneumatically actuated shape-changing interfaces. With their PneUI composite material concept, Yao et al. [2013] contributed a publication a lot of the later research expanded on. The inflatable display from Harrison and Hudson [2009] features a similar way of interaction to what we had in mind with our prototype with the significant difference that our UI elements are not static.

Our implementation of the prototype

We continued by describing the implementation of our prototype from a top-down perspective. We gave an overview of the look and feel of our prototype, introduced all the necessary components that make the prototype work, and lastly described in detail how every part is implemented. The implementation of this prototype is the main contribution of this work. Especially the method we developed to manufacture the inflatable cells could be useful also for other projects. For someone who wants to build a similar prototype or who wants to improve on the one we build, our work provides an overview of all the necessary aspects that should be considered and gives concrete indications for how an implementation can look like and what specific components and materials can be used.

How the bubblepad works

The bubblepad is the part of our prototype handling the user interaction. An array of ten individually controlled, inflatable cells is covered by an elastic sheet of fabric. The pattern of inflated cells creates the user interface. An arrangement of solenoid valves regulates the flow of pressurized air to the cells. The valves are operated by microcontrollers making the inflation state of the cells computer controllable. The microcontrollers are also responsible for reading the values from pressure sensors integrated into the cells. The pressure measurement allows the recognition of user input. Everything comes together at the desktop application interpreting the pressure data and controlling the inflation states of the cells.

A user study to evaluate the prototype

We evaluated our prototype by conducting a user study. The participants had to solve a variety of tasks using the bubblepad. They were then interviewed about their experience interacting with the prototype and asked to suggest their own cells pattern to solve some high-level tasks with the prototype. The study showed that the basic interaction with the bubblepad worked well and even was quite intuitive. We also found out that taking the amount of pressure into consideration can help users solve certain tasks. It did not seem that complex interfaces with frequently changing cell patterns posed significant problems to users. However, the study also revealed flaws in our prototype. Combing multiple cells to a large button did not work with the current implementation. Also, the lack of the ability to sense

user input in between the cells and the visual feedback caused problems. Nevertheless, we reached our goal of creating a dynamically changeable interface with a soft and pliable surface.

5.2 Future Work

5.2.1 Other Approaches

TUIs that use shape change to adapt their appearance and constraints can come in many different forms as outlined in section 2.3 “Pneumatically Actuated Shape-Changing Interfaces”. Maybe our work serves as inspiration for other forms of such TUIs or provides some useful details that help with the implementation. We think that especially the method we developed to manufacture the cells could be used also in other scenarios.

As a unique feature of our prototype is that it is covered with a sheet of fabric, it could also serve as a reference for future research about interactive textiles.

5.2.2 Carrying the Prototype Forward

Developing this prototype further could also be worthwhile as we believe that the full potential of our concept has not yet been reached. Possible hardware improvements that could be implemented like lighting the cells with LEDs, fixing the fabric directly to the cells, or decreasing the cell spacing are highlighted in section 4.7 “Discussion and Improvements”.

In this work, we mainly focused on the input capabilities of the system. However, it also would be interesting to study the output capabilities. Especially when the improvements mentioned above for better visual output get implemented. One could compare the role of tactile feedback and visual feedback in different situations. This could e.g. include

the question of how the roles of tactile and visual feedback change when users get more familiar using the bubblepad. Or focusing on the tactile output one could study how well users can tell different patterns on the bubblepad apart.

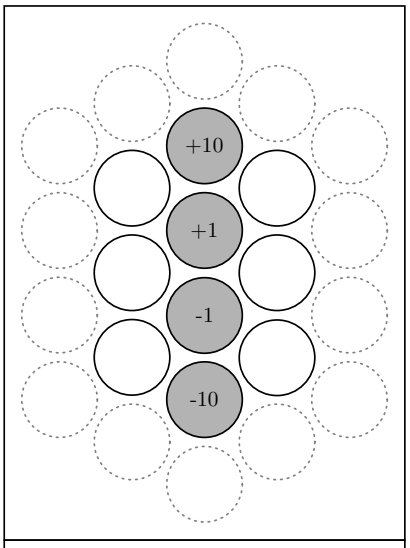
Another important question to ask is how to select a target. In our work, we supposed that the bubblepad already knows which device it should control. However, it is not clear so far how to come to this state and how to figure out what device the user currently wants to control.

Appendix A

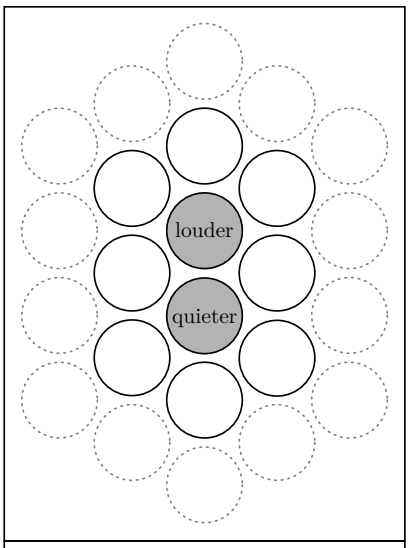
Additional material about the preliminary user study

This appendix contains all patterns suggested for the high-level tasks, the informed consent form, and a script of the introduction and the interview. Note that the interview was conducted in a semi-structured way. Thus the script serves only as a guideline. The interview went slightly differently for each participant dependent on the given answers.

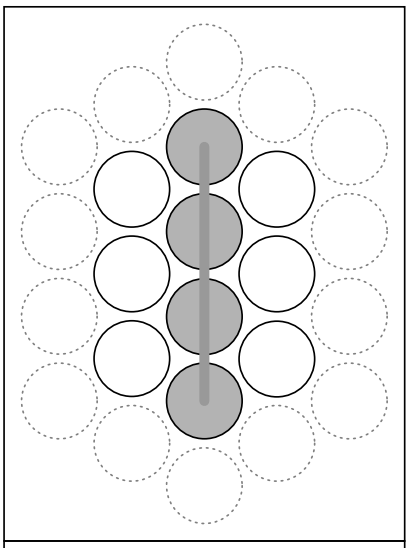
Suggested patterns for the high-level tasks



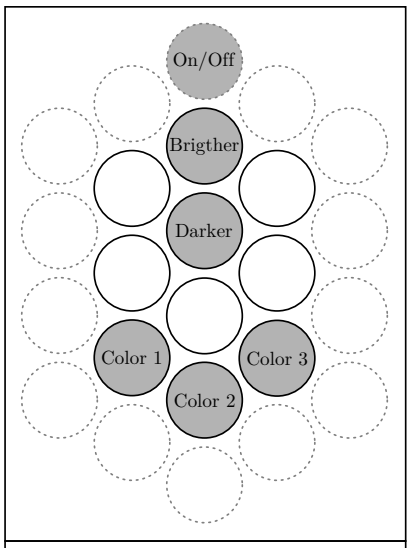
tv volume, pattern 1/3: 4 participants
incrementally change volume



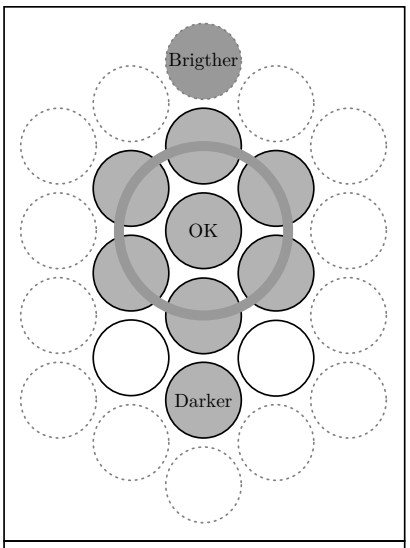
tv volume, pattern 2/3: 5 participants
incrementally change volume
1 participant with extra mute button



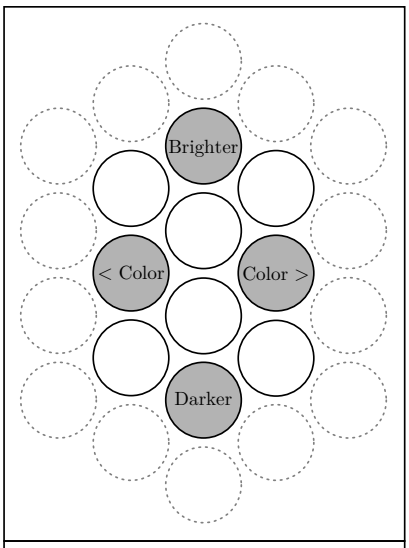
tv volume, pattern 3/3: 1 participant
slide up to increase colume (by a fixed increment) and v.v.



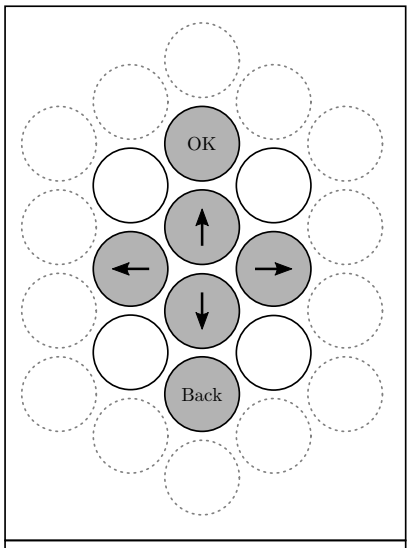
led, pattern 1/3: 3 participants
3 preset colors, incrementally change brightness, turn on/off
1 participants had 4 preset colors
1 participants had 8 preset colors



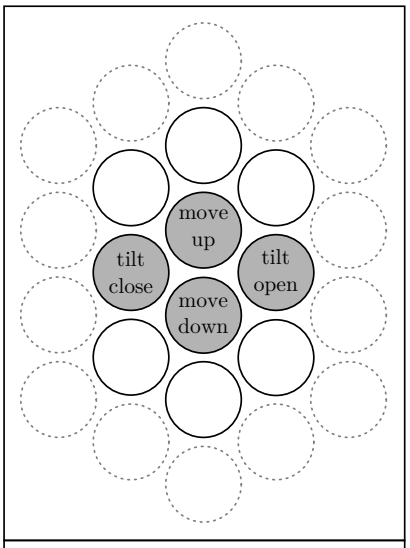
led, pattern 2/3: 5 participants
color wheel, incrementally change brightness
1 participant had a larger color wheel with 10 cells
1 participant added on/off button
1 participant: pressure changes speed of brightness change



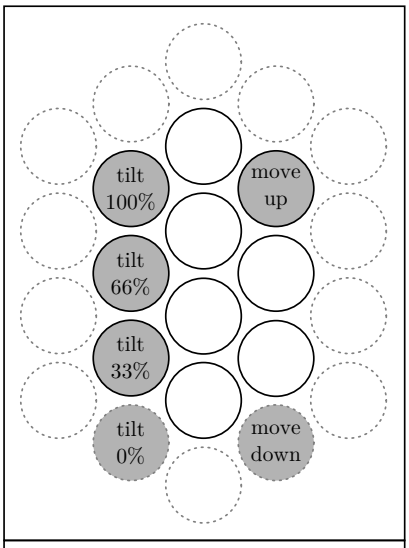
led, pattern 3/3: 2 participants
incrementally traverse through color spectrum
incrementally change brightness



smart tv, pattern 1/1: 10 participants
up/down/left/right to navigate; ok/back to enter/leaf
4 participants had the dpad spaced out
2 participants added extra buttons to control the sidebar
1 participant added a home and voice assistant button



window blinds, pattern 1/2: 8 participants
incrementally change height and tilt
the exact placemanet of the buttons was slightly different for each participant
2 participants added two buttons to totally open/close the blinds (1 height and tilit; 1 only height)



window blinds, pattern 2/2: 2 participants
incrementally change height, 4 buttons for preset tilt angles
1 participant had only two tilt preset (fully opened/closed)

Informed Consent Form

Bachelor's Thesis AirPixel: User Study

Principal investigator: Mathis Beck
RWTH Aachen University
Email: mathis.beck@rwth-aachen.de

Purpose: The goal of our study is to identify problems and strengths of our prototype for different use-cases, collect suggestions about how to improve the design concept, and learn more about how the interaction with the prototype works.

Procedure: The study consists of two parts. In the first part, you directly interact with the prototype and perform different tasks using the prototype. If anything comes to your mind during this phase don't hesitate to communicate it. If you get stuck, you can always ask for help. In the second part, you will be asked questions about the first phase and your thoughts about the prototype in general. During the study, we make an audio recording and record all inputs to the prototype.

Risks: If the prototype somehow fails it does **not** pose any risks. It is operated by pressurized air; however, the pressure is too low to do any harm (< 500 mbar). The study will take about 30 min to 45 min. If you feel exhausted at some point in the study, it is no problem to do a break. Of course, you can also terminate the study at any point.

Confidentiality: No personally identifying information will be published. Some part of the input recording might get published as part of the results. The audio recordings are only used as a reference in the evaluation process. They will be kept private at any point and will only be stored on this computer.

Costs and Compensations: You can get some sweets after the study 😊

☐ I have read and understood the information on this form.

☐ I have had the information on this form explained to me.

Participant's Name	Participant's Signature	Date
	Principal Investigator	Date

Introduction and Interview

- *Disinfect Hands*
- Welcome and thanks again for participating
- What is this study about: We want to test a prototype of a special kind of input device. What exactly this is you can find out in a moment yourself.
- The study is split into two parts:
    - In the first part you can try out the device by yourself. There will be different tasks displayed on the screen. I'll explain this in more detail in a moment
    - In the second part I'll ask you some questions about what you think about the prototype
- The device is operated by air pressure (< 500 mbar). It is not dangerous
- *Recordings:* All inputs to the device, private audio recording, notes.
- *Hand out consent form.* Take your time and read the form, it should say roughly the same what I just told.
- Are there any questions before we start?

- *Show bubblepad*
- You can ignore everything else attached to the prototype. The air compressor and other noises during operation should be ignored as good as possible.
- In a moment you'll see several tasks on the screen. The only thing you need to solve them is the prototype. There are 4 types of tasks, each with 8 repetitions. And at the end there's another single 5th task.
- Time not relevant, you don't have to be fast. You can also just try some things out to see what happens and to see what is required to solve the task. Especially if you encounter a new type of task.
- You can't do anything wrong.
- What to do is always written at the top of the scrren. Unfortunately, the text is in German. But there are only a few sentences, I can just translate them to you when they come up
- If you completed a task, press ENTER on the keyboard to go to the next task. This you'll do until all tasks are completed
- If you have some questions in between, of course just ask
- If you want you can try to speak out lout what you are thinking e.g., if something is strange to you or surprises you. Often this is a little weird in the beginning to vocalize your thoughts. You don't have to do this; just however you feel comfortable.
- If there are no more questions and you are ready, press ENTER to start.

- *… participant completes tasks …*
- *Briefly explain how the bubblepad works*
- Now I would like to ask you some questions about how you would describe your interaction with the prototype.
- Did you quickly realize what needed to be done? Or more confusing?
- Now it would like you to rate some statement on a scale from 1 to 5 stars. 1 strongly disagree; 2 disagree; 3 neither agree nor disagree; 4 agree; 5 strongly agree
    - It was fast and easy to understand how to use the prototype to solve the tasks
    - The prototype helped you to solve the task *(for all tasks)*
- About the fifth task with the different rooms
    - Where the offered controls / cell patterns where helpful?
    - How did you feel about the cell pattern changing while you were interacting with the prototype? More helpful, more confusing?
- In general, anything else that surprised you or came to your mind during you solved the tasks?

- Do you can think of any use-case/scenario where this device could be used?
- Imagine the following scenario: You are sitting on a sofa, in front of you there is a modern smart tv. Next to you, into the armrest, this device is integrated. It does not have to be exactly this device. It could be a fictional better version. E.g., with more/less/bigger/smaller cells, etc.
- Suppose you want to change the tv volume. How would you like to do that with this device? What kind of pattern could you imagine in this case?
- *Hand out template*
- *Show picture of smart tv interface.* Now you would like to navigate through this interface. Maybe to start one of these movies. Same question as before: How can this be accomplished with this device? What cells should be inflated?
- *Hand out template*
- You probably know those LED lamps or light bulbs where you can choose a color. E.g., Phillips Hue. You would now like to change the color and brightness.
- *Hand out template*
- *Show picture of window blinds.* Some buildings have these automatic window blinds. They can be moved up and down and the slats can tilt to open and close.
- *Hand out template*

- Is there anything else you can think of you can control from your sofa?
- Imagine a very primitive version of paint on your tv. Maybe you can choose different colors, different pens and there is a canvas to draw on. Can you think of any way to control this simple version of paint with this device?
- Is there anything you would change about or add to the prototype?
- In comparison to a normal remote control can you see any advantages or disadvantages?
- Now suppose your tv is off. How should the state of the device be in this case? How would you turn on the tv?
- Do you think the device is suitable for the sofa use case?
- Do you have anything else you want to add?
- That's it. Again, thank you very much

Appendix B

A detailed description of the software's operating principles

The complete source code can be found in the repository. [1] All the important parts of the code are documented with comments. For the desktop app, there is an additional javadoc documentation available. This appendix explains how the integral parts of the software are implemented.

B.1 Program running on the atmega8

The atmega8 microcontroller reads the pressure values from the sensors and sends them to the Arduino via the SPI bus. The software is split into two main parts: a class named MPR representing a pressure sensor and the main program loop.

The MPR class

The MPR class mainly consists of two attributes storing the location of the SS pin and two methods used to read a pressure value. The first method `prepareReadRaw()` sends the command to the pressure sensor to start a new pressure

[1] The source code can be found in the repository under software.

reading. The pressure sensor needs up to 5 ms to finish a pressure reading. The second method `finishReadRaw()` sends the command to output the last reading, then reads the 24-bit pressure value from the sensor and returns it as a 32-bit integer number.

The main loop

Starting the microcontroller first all sensors are initialized by specifying the corresponding SS pins. Then the SPI module is initialized and the main loop starts. To handle the SPI communication as a master device we use the Arduino SPI library. In the main loop first, all sensors get told to start a new pressure reading (by calling the `prepareReadRaw()` method). After 5 ms the results get collected by invoking `finishReadRaw()` on all sensors. The returned 24-bit pressure values get truncated to 16-bit values by cutting of the least significant 8 bit. We do not need the whole 24-bit level of precision and in our setup, these last bits anyway contain only noise. After ensuring none of the remaining two bytes are 0xFF they get stored in a buffer. With the help of the data transmission protocol (see section 3.4.4) the buffered values get sent to the Arduino combined in a single data frame. The main loop repeats about every 60 ms.

B.2 Program running on the Arduino

The Arduino gets the pressure values from the atmega8, relais them to the desktop application, receives the desired cell inflation states from the desktop application, and controls the valves. Similar to the atmega8 the software is split into two main parts: a class named ValveArray representing all the solenoid valves and the main loop.

The ValveArray class

The ValveArray class stores the current inflation state of the cells and the locations of the output pins controlling the BJTs. To alter the cell inflation state the class provides the `applyState(newState)` method. As an argument, the method receives the desired new inflation state. It compares the new state to the current state. If any cells should get inflated the appropriate valves get turned on for a short amount of time inflating the cells. Then the mode-valve is

turned on, and the same process happens for the cells that should get deflated. The inflation states are represented by a 16 bit long bit array. I.e. if bit i is set to 1, cell i (connected to valve i) is or should get inflated. Respectively 0 represents the deflated state.

Initialization and ISR

Before the main loop starts, the valve array gets initialized by declaring all the BJT control pins as outputs, the program waits until the USB serial connection is ready, and the SPI module is put into slave mode. The SPI module gets also set up to invoke an interrupt every time a new byte of data arrives. The interrupt stops the main program and jumps to a special interrupt service routine (ISR). In this routine, the received data gets stored in a buffer. It is important that the ISR is finished as quickly as possible. As the Arduino has no control over how fast the data comes in, data can get lost when a new interrupt occurs while the ISR from the previous transmission still gets executed. To avoid this the atmega8 waits a short amount of time after the transmission of each byte to give the Arduino time to buffer the data.

Receiving data from the atmega8 and relaying it to the desktop application

The main loop first checks if there are any received bytes from the atmega8 in the buffer that were not already processed. This is done by keeping track of two pointers for the buffer. One points to the position of the last read byte, the other one to the position of the last received byte. The buffer is implemented as a ring buffer. When receiving too many bytes before processing them the received-pointer would pass the read-pointer and we would lose data. However, the buffer is large enough to store two full frames with pressure data, giving the Arduino plenty of time for reading the data from the buffer, while the atmega8 is collecting new data from the sensors. If the two pointers are not equal, the program reads as many bytes from the buffer until the two pointers are equal again. The read bytes get sent via the serial USB connection to the desktop application.

Receiving data from the desktop application

The main loop continues by checking if there is any data received from the desktop application. The desktop application transmits the desired valve states with the same protocol used to transmit the pressure values. Here a data frame contains the 16-bit-long bit array encoding the cell in-

flation states as described in the ValveArray class. The data frame gets decoded as described in section 3.4.4. When receiving the first byte of the frame the most significant 8 bit of the inflation state bit array gets updated. The second byte of the frame updates the least significant 8 bit. In both cases the `applyState(newState)` method is called with the newly received states. Now the end of the main loop is reached and a new iteration starts immediately.

We had to use a different sensor for one cell

During the construction of the prototype, one of the pressure sensors failed. As we could not order a replacement in time we were forced to use another type of sensor we still had lying around from earlier tests. It is a 5 V analog pressure sensor. This is why in the current version of the code the Arduino also reads this extra pressure sensor, decodes the data frames from the atmega8, and inserts the pressure value from the replacement sensor at the right place before relaying the pressure data to the desktop application.

B.3 The desktop application

Similar to the microcontroller software the desktop application also has a main loop drawing the GUI on the screen, a setup section executed at the start, and classes encapsulating different functionality. We will only elaborate on the parts of the software that are responsible for controlling the display. Apart from that, a large part of the code is responsible for rendering the tasks used in the user study. However, this code mostly consists of drawing primitive shapes and handling simple logic for the tasks.

The Sensor class

When the data from the Arduino comes in it first encounters the Sensor class. There is one instance of this class for every one of the ten sensors. In each main loop iteration, every sensor gets updated with the last received raw sensor value. Depending on this raw value a sensor gets put into one of these four states: OUTOFRANGE, DEFLATED, INFLATED, or PRESSED. To decide between the first three states the raw value is just compared to fixed threshold values.

The pressure values are used to distinguish between the states out-of-range, deflated, inflated, or pressed

The detection of the pressed state is a little more sophisticated. When a cell gets pressed the pressure offsets by some amount from the previous inflated pressure level. This means e.g. when a cell is inflated with less pressure, to begin with also the pressure level when pressed is smaller. Thus the pressure level for the PRESSED state has to be dependent on the pressure level of the INFLATED state. Additionally, while a cell is inflated the pressure level can drift slightly either due to changes in temperature or due to small leakages in the system. This is why the threshold for the PRESSED state can not be fixed but has to be constantly adapted. To accomplish this a baseline pressure value gets constantly calculated while in the INFLATED state by using a moving average over the last few raw values. When the pressure moves above this baseline by a certain amount the sensor switches to the PRESSED state until the pressure falls back down to the baseline level. While the PRESSED state is active the baseline remains fixed. To avoid the baseline of increasing when a cell is pressed instead of switching to the PRESSED state and thus missing the press, the baseline value gets updated with a small delay.

The Bubblepad class

The Bubblepad class controls the real-world bubblepad and maps the sensors to the cells. Cells also have their own class mainly storing their current state determined by the state of the associated sensor. The Bubblepad and Cell class include an instance of a special BubbpladRenderer and CellRenderer class, responsible for rendering a graphical representation of the bubblepad and the current cell states on the screen.

Bibliography

Alexandra Delazio, Ken Nakagaki, Roberta L. Klatzky, Scott E. Hudson, Jill Fain Lehman, and Alanson P. Sample. Force jacket. In Regan Mandryk, Mark Hancock, Mark Perry, and Anna Cox, editors, *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, New York, NY, USA, 2018. ACM. ISBN 9781450356206. doi: 10.1145/3173574.3173894.

Sean Follmer, Daniel Leithinger, Alex Olwal, Akimitsu Hogge, and Hiroshi Ishii. inform. In Shahram Izadi, Aaron Quigley, Ivan Poupyrev, and Takeo Igarashi, editors, *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 417–426, New York, NY, USA, 2013. ACM. ISBN 9781450322683. doi: 10.1145/2501988.2502032.

Kristian Gohlke, Eva Hornecker, and Wolfgang Sattler. Pneumatibles: Exploring soft robotic actuators for the design of user interfaces with pneumotactile feedback. In Saskia Bakker, Caroline Hummels, Brygg Ullmer, Luc Geurts, Bart Hengeveld, Daniel Saakes, and Mendel Broekhuijsen, editors, *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 308–315, New York, NY, USA, 2016. ACM. ISBN 9781450335829. doi: 10.1145/2839462.2839489.

Chris Harrison and Scott E. Hudson. Providing dynamically changeable physical buttons on a visual display. In Dan R. Olsen, Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott Hudson, and Saul Greenberg, editors, *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, page 299,

New York, New York, USA, 2009. ACM Press. ISBN 9781605582467. doi: 10.1145/1518701.1518749.

Filip Ilievski, Aaron D. Mazzeo, Robert F. Shepherd, Xin Chen, and George M. Whitesides. Soft robotics for chemists. *Angewandte Chemie (International ed. in English)*, 50(8):1890–1895, 2011. doi: 10.1002/anie.201006464.

Hiroshi Ishii, Daniel Leithinger, Sean Follmer, Amit Zoran, Philipp Schoessler, and Jared Counts. Transform. In Bo Begole, Jinwoo Kim, Kori Inkpen, and Woontack Woo, editors, *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 687–694, New York, NY, USA, 2015. ACM. ISBN 9781450331463. doi: 10.1145/2702613.2702969.

Seoktae Kim, Hyunjung Kim, Boram Lee, Tek-Jin Nam, and Woohun Lee. Inflatable mouse: Volume-adjustable mouse with air-pressure-sensitive input and haptic feedback. In Mary Czerwinski, Arnie Lund, and Desney Tan, editors, *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 211, New York, New York, USA, 2008. ACM Press. ISBN 9781605580111. doi: 10.1145/1357054.1357090.

Daniel Leithinger, David Lakatos, Anthony DeVincenzi, Matthew Blackshaw, and Hiroshi Ishii. Direct and gestural interaction with relief. In Jeff Pierce, Maneesh Agrawala, and Scott Klemmer, editors, *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, page 541, New York, New York, USA, 2011. ACM Press. ISBN 9781450307161. doi: 10.1145/2047196.2047268.

Microchip Technology Inc. Atmega48a, atmega48pa, atmega88a, atmega88pa, atmega168a, atmega1688pa, atmega328, atmega328p datasheet: Ds40002061b, 2018. URL `https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf`.

Yong-Lae Park, Jobim Santos, Kevin G. Galloway, Eugene C. Goldfield, and Robert J. Wood. A soft wearable

robotic device for active knee motions using flat pneumatic artificial muscles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4805–4810. IEEE, 2014. ISBN 978-1-4799-3685-4. doi: 10.1109/ICRA.2014.6907562.

Ivan Poupyrev, Tatsushi Nashida, and Makoto Okabe. Actuation and tangible user interfaces. In Brygg Ullmer and Albrecht Schmidt, editors, *Proceedings of the 1st international conference on Tangible and embedded interaction - TEI '07*, page 205, New York, New York, USA, 2007. ACM Press. ISBN 9781595936196. doi: 10.1145/1226969.1227012.

Simon Robinson, Céline Coutrix, Jennifer Pearson, Juan Rosso, Matheus Fernandes Torquato, Laurence Nigay, and Matt Jones. Emergeables. In Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade, editors, *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3793–3805, New York, NY, USA, 2016. ACM. ISBN 9781450333627. doi: 10.1145/2858036.2858097.

Alexa F. Siu, Eric J. Gonzalez, Shenli Yuan, Jason B. Ginsberg, and Sean Follmer. shapeshift. In Regan Mandryk, Mark Hancock, Mark Perry, and Anna Cox, editors, *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, New York, NY, USA, 2018. ACM. ISBN 9781450356206. doi: 10.1145/3173574.3173865.

Ivan Sutherland. The ultimate display. *Proceedings of the IFIPS Congress 65(2):506-508. New York: IFIP*, 2, 1965.

Shan-Yuan Teng, Tzu-Sheng Kuo, Chi Wang, Chi-huan Chiang, Da-Yuan Huang, Liwei Chan, and Bing-Yu Chen. Pupop. In Patrick Baudisch, Albrecht Schmidt, and Andy Wilson, editors, *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 5–17, New York, NY, USA, 2018. ACM. ISBN 9781450359481. doi: 10.1145/3242587.3242628.

Marynel Vázquez, Eric Brockmeyer, Ruta Desai, Chris Harrison, and Scott E. Hudson. 3d printing pneumatic device controls with variable activation force capabilities. In

Bo Begole, Jinwoo Kim, Kori Inkpen, and Woontack Woo, editors, *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1295–1304, New York, NY, USA, 2015. ACM. ISBN 9781450331456. doi: 10.1145/2702123.2702569.

Mark Weiser. The computer for the 21 st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999. ISSN 1559-1662. doi: 10.1145/329124.329126.

Lining Yao, Ryuma Niiyama, Jifei Ou, Sean Follmer, Clark Della Silva, and Hiroshi Ishii. Pneui: Pneumatically actuated soft composite materials for shape changing interfaces. In Shahram Izadi, Aaron Quigley, Ivan Poupyrev, and Takeo Igarashi, editors, *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 13–22, New York, NY, USA, 2013. ACM. ISBN 9781450322683. doi: 10.1145/2501988.2502037.

Index

Typeset November 25, 2021