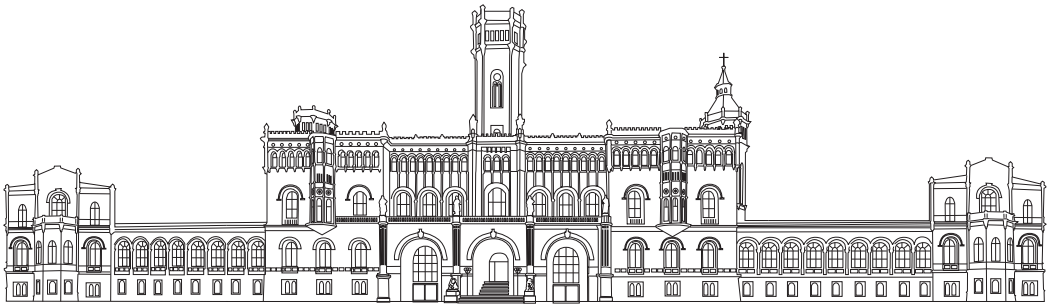


LEIBNIZ UNIVERSITÄT HANNOVER

Faculty of Electrical Engineering and Computer Science
Human-Computer Interaction Group

CASUAL INTERACTION WITH A BRACELET

A Thesis presented for the degree of Master of Science

by

KAROLINE BUSSE

December 2014

First Examiner	:	Prof. Michael Rohs
Second Examiner	:	Prof. Franz-Erich Wolter
Supervisor	:	M.Sc. Henning Pohl

ABSTRACT

Short summary of the contents in English...

ZUSAMMENFASSUNG

Kurze Zusammenfassung des Inhaltes in deutscher Sprache...

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

STL Sterolithography

HCI Human-Computer Interaction

IR	Infrared
LED	Light Emitting Diode
SPP	Serial Port Profile
CAD	Computer-Aided Design
MIT	Massachusetts Institute of Technology
USB	Universal Serial Bus
PWM	Pulse-width Modulation
PCB	Printed Circuit Board
MSE	Mean Square Error
GSS	Golden Section Search

1 INTRODUCTION

2 USAGE SCENARIO

> Alice comes home from grocery shopping. While carrying her stuff to the kitchen, she quickly turns on the light by simply touching her bracelet.
>
> After storing everything, she decides to relax in the living room by reading a book. After turning the kitchen lights off with a touch interaction, she makes herself comfortable on the sofa and picks up the book. To feel more comfortable, Alice dims the living room light a little by touching the bracelet rim and tilting her wrist, just as she was turning a dimmer knob. The living room lights dim accordingly, so she doesn't need to look any closer at the bracelet.
>
> Later that day, Alice prepares the dinner. While the food is finally in the oven, Alice returns to the living room and prepares the table for dining with her friends that will show up later. She uses the bracelet's touch controls to specify exact colors and brightnesses for each light source in the room. Alice saves three lighting setups together with unique activation gestures: A warm setting for their guests' arrival and enjoying the welcoming drinks, a well-matched dining setup with focus on the table and foods, and a darker, colorful mood for the after-dinner party. Changing between these presets with hand gestures allows Alice to focus on her guests and on the meal instead of wasting time and focus by fiddling with wall panels or switches.

This scenario illustrates three different levels of casual interaction with a device: A simple on/off function by covering the bracelet's touch surface, intuitive and eyes-free brightness dimming by touching the bracelet and tilting the wrist, color mood change by precise touch and a fine-tuned hue setup using focused touch interaction. In addition, three hand gestures are available to easily access frequently used setups like bright white work light or slightly dimmed relaxing illumination.

The mentioned interactions induce several requirements for the device. First, the device should stay where it is needed without encumbering the user. Typical remotes or smartphones occupy at least one hand for every interaction they offer. This is not desired, so traditional hand-held devices do not fit the scenario presented above. Instead,

there is the need for a wearable device that is attached to the body without getting in the way during everyday activities.

Second, touch-free interaction (i.e. control by gestures) should be possible with respect to the casual interaction scenarios pictured above. Capacitive touch input offers flexibility and versatility compared to traditional keypads and enables gestured touch input. In addition, the required hardware should be kept on a low cost level and encompass only needed components to keep energy consumption at a minimum, since every recharge procedure is cumbersome to the user.

A bracelet-typed device can fulfill the requirements stated above. It is slim and doesn't encumber you, so it can be worn on the arm all day.

3

RELATED WORK

Casual interaction, which is the core topic of this thesis, has been explored by other researchers.

Pohl and Murray-Smith have researched casual interaction with everyday electronics like smartphones [**?**].

4 THE BRACELET

The interactive bracelet consists mainly of an electronic ink display and several touch sensors for conscious interaction and a motion sensor for gesture recognition. The bracelet's design focuses on wearing comfort, low weight and small error of unintended activation.

4.1 MANUFACTURING TECHNIQUES

In this section, the various methods and tools used for designing and manufacturing the bracelet prototypes will be explained in detail.

4.1.1 *Computer Aided Design*

In the beginning of a design iteration, a virtual model of the desired object is built with a 3D Computer-Aided Design (CAD) program. For all designs in the context of this thesis, the open source CAD modeler FreeCAD[5] was used. This program classifies itself as a general purpose 3D parametric modeler. A typical work-flow when designing a bracelet is as follows:

The model's basis is a circumference sketch for the bracelet. As the human wrist does not follow a circular shape, all designs made for this thesis are based on an oval circumference. In FreeCAD, this is realized by a composition of arcs and straight lines. It is important to constraint the sketch with radii, lengths and angles as well as symmetries or perpendicular constraints. The sketch should be fully constrained (i.e. leaving no degrees of freedom) before moving on to the next step, also with respect to the printing of the finished design.

In the next step one or more profile cuts are added to the circumference. They define the thickness and shape of the bracelet at different points along the oval circumference. For increased comfort, the bracelet should be as slim as possible, especially on the "backside" below the palm. In more complicated designs, an alternative to complex sketches for the profiles is the creation of multiple, overlaid sketches as a preparation for a subtraction operation in the later process. All those profiles need to be fully constrained as well.

After the circumference as well as one or more profiles are added to the design, a first solid is created by sweeping the profile(s) around the circumference curve. This operation creates a basic shape that can be refined further on, typically by chamfering or filleting the edges with respect to increased wearing comfort. If a complex shape is desired, multiple sweeps can be generated and used in boolean oper-

ations such as union or subtraction. This is the usual approach for most bracelet designs.

The last step in the design process is usually the finishing of the CAD model. In the bracelet case, this translates to edge smoothing with chamfer or fillet tools. Smoothed edges increase the overall wearing comfort of a bracelet, so they are very desired on edges that contact with the skin.

The finished designs are then exported as mesh files (usually in Sterolithography (STL) format) for printing.

4.1.2 *3D Printing*

The Human-Computer Interaction (HCI) group's workshop includes a powder bed and inkjet head 3D printer, a ProJet 360 by 3DSystems, Inc[2]. This manufacturing technique was developed in 1989 by the Massachusetts Institute of Technology (MIT) [13] and then licensed to the Z Corporation in 1995. On January 3, 2012, Z Corporation was acquired by 3DSystems, Inc.

When printing an object, the print bed is filled first with a base layer of powder. The powder used by the printer is called VisiJet PXL, a plaster-like substance. After filling the bed, a standard inkjet printhead is cleaned and prepared for the printing process. Immediately after this setup is complete, the additive manufacturing process starts.

The model is created by printing the binder fluid onto the plaster. After each printed layer, a new thin layer of powder is added to the print bed. The ProJet360 allows for a layer thickness of 0.1 mm with a vertical build speed of 2 cm per hour[1]. This allows even delicate structures without any additional supports, since the printed object is surrounded and therefore supported by plaster powder during the production process. The only drawback of this printing process is that it doesn't support closed, hollow objects since there is no possibility of removing the enclosed excess powder after the process has finished. When designing models for 3D printing, this constraint has to be kept in mind.

The finished object is then carefully removed from the build bed and any excess powder is gently brushed or blown off. The printer offers a cleaning chamber with a pressurized air pistol and a vacuuming system to assist in that task. Without further hardening, the objects are very fragile and easy to break, even with the pressurized air pistol included in the printer. In order to drastically increase the strength of the prints, they are infiltrated with a fluid after they were thoroughly cleaned. Prints produced by the ProJet 360 can be infiltrated with one of three different substances with varying characteristics: The ColorBond "instant-cure infiltrant", the two-part StrengthMax infiltrant "ideal for functional models", and the Salt Water Cure

TYPE	CONFIGURATION	DIAMETER	SPRING FORCE
A	35°	1.4 mm	8.06 N
D	30°	1.4 mm	44.46 N
OO	1.2 mm in spherical radius	2.4 mm	1.11 N

Table 1: Test setups for Shore types A, D, and OO[**?**]

"eco-friendly and hazard-free infiltrant"[1]. All prints produced for this thesis were infiltrated with ColorBond.

The infiltration step adds strength and hardens the material, resulting in a sturdy printed object. However, the objects created with this technique are very rigid and any bending load might break them easily. Wall strengths of 1.5 mm and up have been proven sturdy enough for a bracelet shape, although this also depends on the object geometry.

4.1.3 *Silicone Casting*

Another manufacturing process for bracelet prototypes used in this thesis is liquid silicone casting. Two different types of silicone were used for making various bracelet prototypes, both from manufacturer Smooth-On: Sorta-Clear 37 and Mold Star 15 Slow.

The most important characteristic for silicone in prototype production is the hardness, measured in Shore (after Albert F. Shore) or Durometer. It measures the indentation of a material with a special device which is also called Shore Durometer. It consists of a hardened steel rod with a finer tip and is available in two versions, since there are two different scales for Shore hardness (cf. table 1). The Shore A scale is designed for softer materials and the Shore D scale for harder ones, but they do overlap, so a material classified in Shore D hardness is not necessarily harder than another material classified in Shore A hardness. Each scale ranges from values 0 to 100, higher numbers indicate higher material resistance. The Shore hardness is specified in EN ISO 868.

When dealing with silicone, the Shore A scale is sometimes too "hard" for soft rubbers. Another standard (ISO 7619????) therefore specifies twelve different Durometer types, where the OO scale is commonly used for soft silicones. Figure 1 shows the three Shore hardness scales A, D, and OO, as well as some examples for everyday objects and their corresponding Shore values.

The silicone rubbers used for casting bracelet prototypes are both located on the Shore A scale. The softer Mold Star 15 Slow has a Shore A hardness of 15 that could be roughly compared to that of a rubber

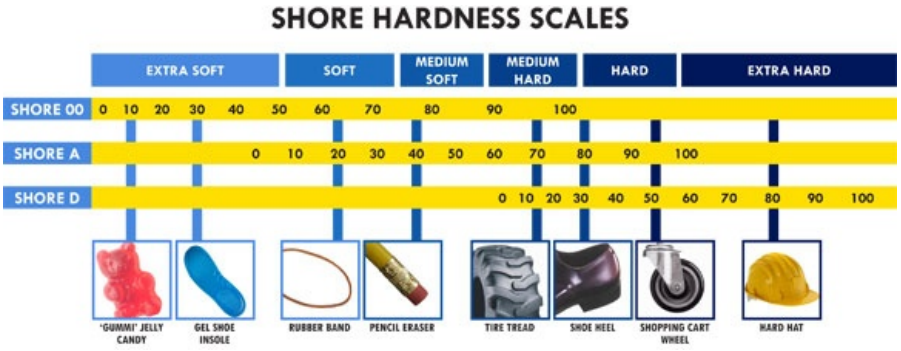


Figure 1: Shore types A, D, and OO in comparison with everyday examples of various hardnesses.[**?**]

band according to figure 1, while the slightly harder Sorta-Clear 37 has a Shore A rating of 37, similar to that of a pencil eraser.

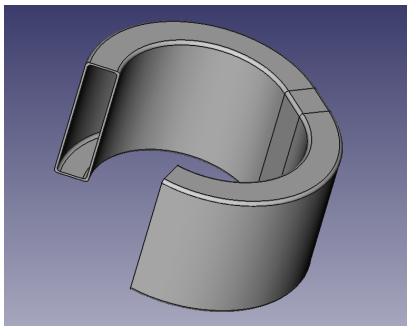
Both silicone products are two compound mixes that have to be added up and stirred before casting. While the Mold Star silicone has a rather low viscosity, casting the Sorta-Clear requires careful mold design, since it does not distribute well and is rather viscous. A vibrating table can help in filling the mold completely, but nonetheless were casts with the Sorta-Clear silicone much less fruitful, especially for the detailed molds of the one-piece designs (cf. section **??**).

4.2 DESIGN PROCESS AND PROTOTYPE MANUFACTURING

The design process for the interactive bracelet presented in this thesis went through different stages. At first, a 3D-printed casing was favored, but later on a cast silicone bracelet turned out to be more comfortable for the user. The different prototypes are explained in detail in the following sections.

4.2.1 *Rigid Designs*

The first approach that comes to mind when thinking about bracelet design is a cuff-like, rigid shape. From the CAD point of view, the first bracelet prototype consisted of a single rectangular profile rotated around a oval curve which was derived from a measured wrist. The bracelet's inside was hollow, that would be the storage space for all electronic components. The cuff's gap was just large enough for the wrist to fit through, although in reality, this lead to light scratches on the skin in combination with the rough texture of the printed material. In addition, the uniform thickness made this first prototype uncomfortable to wear, especially at the open ends. So this fist prototype had some clear downsides that were eliminated in the next iteration.



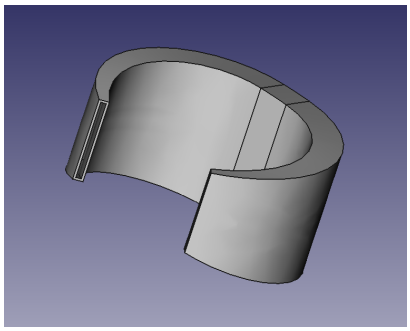
(a) CAD view

(b) Printed bracelet

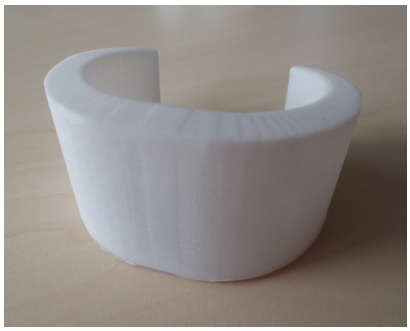
Figure 2: First rigid design with uniform thickness

Due to a settings error in the PC that controls the 3D printer, the printout of this first design was aborted after one third of the process.

In the following iteration, a bracelet of varying thickness was designed to make wearing the prototype more comfortable. Decreasing the thickness from XX cm to YY cm made the look more appealing and wearing a little less clumsy. At the same time, the wall strength was decreased from X.Y mm to 0.7 mm, which made it very fragile in fabrication and usage, both prints broke during post-processing. Wearing the cuff while working on a PC feels only slightly uncomfortable, but twisting the hand is encumbered by the tight-fitting bracelet. A design goal for future prototypes derived from this prototype was adding more space between the arm and the bracelet to ensure better comfort while wearing it.



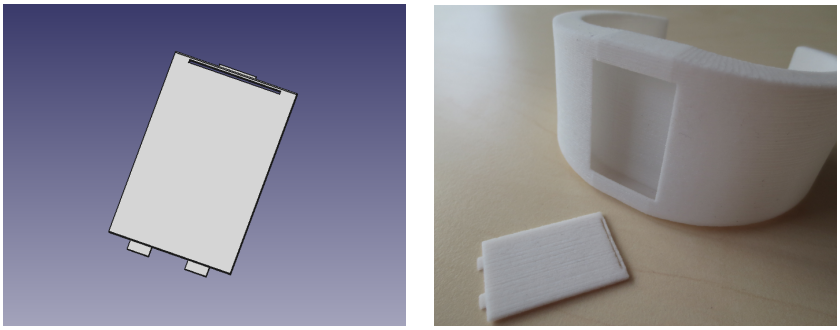
(a) CAD view



(b) Printed bracelet

Figure 3: Second design featuring a tapered shape

A modified design of the aforementioned prototype featured a removable lid since the tapered shape made it hard to access the inside space of the bracelet. The lid design was inspired by battery case covers commonly found in remote controls or small electronic devices. It features two nubsis on the one side that support the lid in its place and another, smaller nubsi with a cavity in the material right next to it, so the nubsi can snap just inside the lid gap when it closes.



(a) Lid in CAD view (b) Printed bracelet

Figure 4: Third design with removable lid

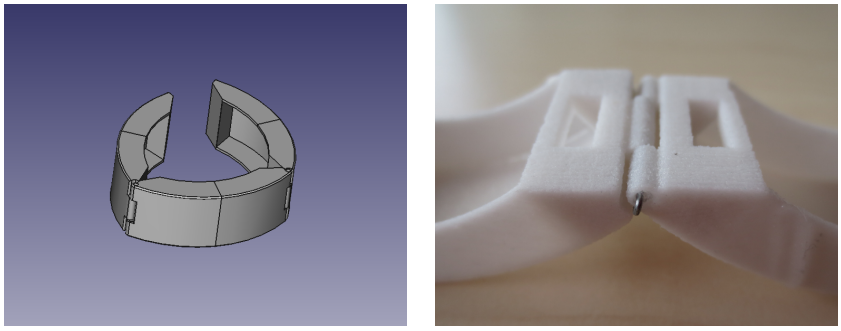
It turned out that the lid features were designed too fine, especially the flexible part was too thin to work as intended. The lid had to be opened and closed very cautiously and overall, the construction seemed not reliable for daily use. In addition, the rigid shape still lead to clumsiness in wearing the bracelet, so the whole concept of rigid bracelets was left behind.

4.2.2 *Segmented Designs*

The search for a more flexible printed bracelet shape lead to an entry for an activity bracelet design contest by Daniel Muschke on a 3D printing template exchange site called *GrabCAD* [11]. Muschke created a CAD file for a bracelet consisting of three segments that are connected by slow hinges. This design considered the electronic components like a micro-controller and a micro Universal Serial Bus (USB) port which made a good start for further modification, but unfortunately the file format made it impossible to alter the design in detail. A print was possible since STL files were included in the upload. However, the design turned out to be too small to be actually wearable, but it demonstrated that printed hinges work well with a pivot made from wire. The style felt more comfortable to wear than the precious prototypes and felt leaner on the wrist than the rigid prototypes. Overall, the printed design looked promising so the idea of a tri-part bracelet was investigated further.

Recreating Muschke's design was not as easy as planned, since the parametric modeling approach was very different in comparison with previous designs. The first prototypes were based on a wrist-like circumference curve, while the multi-part design originated from a partitioned circle, since all parts should have the same dimensions in length and curvature. Adjusting the part size to the designated wrist turned out to be difficult, and prints of the design were either too small or too big.

The bracelet segments are hollow and open on the side that lays to the wrist, this leaves enough space for storing electronics. Since the parts are only connected by hinges, routing the necessary wires between the segments needs to be considered, e.g. by leaving small holes right next to the hinges.



(a) CAD view (b) Hinge Detail

Figure 5: Multi-part design

Since the manufacturing techniques offered at the HCI group don't allow for slow hinges (as the inspiration by Muschke suggests), a different solution for keeping the bracelet closed while worn on the wrist needed to be considered. A magnet clasp with small neodymium magnets was tested, but attaching them turned out to be more difficult than expected. When attached to the inside of the segment tips, the magnetic force was too weak for reliably keeping the bracelet together. Mounting the magnets on the outside of the segments resulted in mounting difficulties, since the magnetic force of neodymium magnets is very strong and frequently resulted in torn glue layers.

When an electrophoretic display was considered to be part of the bracelet, the segmented approach became undesired, since in addition to aforementioned difficulties, the usable surface space was relatively small when it came to hosting a single big component like a display. This issue led to further research into various other manufacturing techniques and potential materials for bracelet prototypes, and eventually led to cast silicone.

4.2.3 *Silicone Bracelet*

After some consideration on a flexible e-ink display, a prototype bracelet made of silicone was considered. The molds used in the casting process were designed and printed just as the bracelets presented in the previous sections. However, designing a mold was significantly more difficult.

As with 3d printed prototypes, the bracelet positive was designed first. The flexibility of silicone allows for some features that weren't practical when implemented in rigid material, for example cavities for

electronic components. This resulted in overall more complex bracelet concepts. In addition, closing mechanisms like magnets had to be considered in this design stage.

When the CAD process for the positive is finished, the mold is successively designed by adding surrounding geometries to the model and applying boolean difference operations. If reuseability is desired for individual mold components or the whole mold, the geometry and constellation of the mold parts needs to be considered and the characteristics of the printed material have to be taken into consideration. For example... Another important aspect of mold design is planning the casting process. Some types of silicone are more viscous than others, and the mold design needs to make sure that the liquid rubber reaches all corners and delicate parts well. The distribution inside the mold can be supported by applying gentle vibration during the cure process, but this assumes that the liquid silicone is already distributed into most regions of the mold.

The first silicone design was a simple strap with an open pocket for the long electrophoretic display and some cavities for a magnet clasp.

Mold design turned out a little tricky but finally succeeded. A two-piece mold was printed which only needed a little post-processing to fit properly together. The mold was coated with black spray paint to make the inside a little smoother, but this didn't work as intended so the painting step could be omitted in future mold making processes. The filling holes for the silicone were too small, and the mixture was more viscous than expected.

The first cast with a closed mold and filling through the holes was unsuccessful; it produced two small end pieces and nothing in between. For following casts, one part of the mold was filled with silicone and closed afterwards; this turned out significantly better. The orientation in which the mold is placed during the dry period is also relevant, as air bubbles will float towards the "top" of the mold, leading to instabilities when oriented inappropriately. Getting the silicone part out of the mold was no problem.

This first rubber bracelet design involved a magnet clasp, but it was infeasible to reliably attach magnets to silicone with anything but silicone itself and they would likely jump out of place and snap together if placed too close to each other in the design.

Two casts were made, one with each of the available silicone mixtures. The softer one was slightly too soft and had a disappealing color. In addition, the cavity rims were too short, so the display would jump out of place almost instantly when bending the bracelet.

4.2.4 *One-Piece Silicone Bracelet*

The next design was a ring-like silicone bracelet in one piece, so the issue with the clasp was no concern. The mold for this prototype con-

sisted of three pieces, two rings and a bottom plate. Due to the very rigid characteristic of the printed material, the molds could not be recovered after the cast and had to be destroyed. In order to prevent unnecessary waste of material, the wall strength was reduced to 2.5 mm which turned out to be strong enough to survive the cast. The advantage of using one-time molds was the possibility to add tunnels to the design. This was used especially for the display area.

This mold was much harder to fill with silicone than the previous one. Especially the Sortaclear silicone was too viscous to fill the complete mold, resulting in broken casts. The green MoldStar silicone turned out to work quite well. A simple vibrating plate made with a servo monitor with some scrap materials as an imbalance glued onto a plastic tray helped filling all parts of the mold and reducing air bubbles in the cast. The design also features wider cavity rims to hold the display in place and first wearing tests resulted in success.

The design was iterated several times as the hardware components were layouted on a Printed Circuit Board (PCB) shield and the display was omitted for initial interaction tests.

4.3 TEENSY DEVELOPMENT BOARD

The core part of the bracelet's electronics is the Teensy USB development board, which is built around a MK20DX256 32 bit ARM Cortex-M4 Processor running on 72 MHz clock speed [14]. The Teensy can be programmed using the popular Arduino IDE and thus can profit from the great amount of existing libraries and code examples for the Arduino family.

4.4 TOUCH SLIDER

The major input interface of the bracelet is a touch surface which consists of seven cut copper foil segments that are placed in a zigzag pattern. This capacitive strip can either be used as an array of seven individual buttons or as a single large surface that can detect swiping gestures or a primitive variant of multi-touch interaction.

4.5 3D ACCELEROMETER

4.6 VISUAL FEEDBACK

To provide visual feedback to the user, an RGB Light Emitting Diode (LED) is incorporated in the bracelet. Since most changes are reflected by the light source with no perceptible delay, the LED serves mainly as an indicator for the recognition of specific interactions like the double tap detection.

5 INTERACTIVE LIGHT SOURCE

The previously mentioned light source for the scenario is built around a 12 V powered RGB LED strip, which can be purchased for home lighting or car decoration. It is approximately XX cm long and features XX LEDs on each XX cm long segment. The original controller box which included an Infrared (IR) receiver for a remote was removed and replaced by an Arduino Uno [3] with a custom shield.

The shield features a Roving Networks RN42 Bluetooth module [12] and some transistors for upscaling the voltage from 3.5 V to 12 V. The whole setup is powered by a standard 12 V power supply and encased in a spherical lamp shade made from translucent glass in order to diffuse the spotted impression of the LED strip (cf. figure **??**).

Figure 6: The interactive light source, encased.

Communication with the bracelet takes place via Bluetooth. The RN42 module implements the Bluetooth Serial Port Profile (SPP) and can be accessed easily with the `SoftwareSerial` module from the Arduino standard library. The lamp is configured as a Bluetooth slave, the bracelet acts as the master device. For command transmission, a serial connection is established over the Bluetooth link between the devices. Once it is set up, the master transmits a nine-digit string composed from three values from 000 to 255. Note that all values must have three digits, leading zeros must not be omitted. These values represent the intensities of the red, green, and blue channel respectively. The algorithm on the Arduino feeds them into the platform's analog

outputs which feature Pulse-width Modulation (PWM). These output signals are scaled as mentioned before and so the light color changes. A simple fading algorithm prevents disruptive flashing while switching colors.

It turned out that the RGB LEDs use a lot of current, a test with an adjustable power supply yielded that the lights become much brighter with increased current. The power supply was limited to XX Ampere and there was no saturation indicated at this level. However, the standard power supply mentioned above serves only XX A which results in less brightness. A way to improve this would be to shorten the LED strip by cutting off several segments.

6 INTERACTION

This chapter describes the various interaction levels with the bracelet in detail and illustrates the respective algorithms.

6.1 PAIRING THE BRACELET WITH A LIGHT SOURCE

The bracelet's Bluetooth device automatically searches for nearby devices and connects to a range of matching IDs without any interaction or confirmation by the user. This can pose a security risk to spoofing a lamp device, but since no sensitive data is handled by either participant, the impact would rather be an annoying disturbance than a serious threat to privacy.

6.2 SWITCHING THE LIGHT SOURCE ON AND OFF

Since turning the lights in a specific room on or off is a frequently used interaction, it should require few complexity in terms of cognitive as well as physical workload, i.e. a simple, easily memorable way of interaction is much desired. For those reasons, simply covering the touch surface with the whole hand and holding for a few seconds will result in switching the lights on or off, dependent on the current state.

6.3 ADJUSTING THE LIGHT SOURCE'S BRIGHTNESS

Apart from switching the lights on or off, a change in brightness is the second most desired interaction in the smart lighting scenario. For example, the incentive of watching TV in the living room benefits from a dimmed light setting. However, if the user fails to locate the remote control for the television, a quick dim interaction towards brighter light facilitates the search for the missing remote control. After the item is found, the brightness level of the room's lighting can easily be dimmed back to the desired setting.

The dimming interaction is different from the other use cases, since there exists a physical solution for this task in form of dim knobs for wall outlets. Usually, those wall dimmers are rotary knobs connected to a potentiometer which dims the light source by increasing the electric resistance. Hence, the interaction of turning a knob for dimming the light level is an association for many people.

The intention was to preserve this association to make the interaction with the bracelet more intuitively. The dimming interaction is a

combination of twisting the wrist while covering the bracelet's touch surface, imitating the interaction with the wall dimmer. A counter-clockwise movement reduces the brightness, while a clockwise twist increases the brightness.

6.4 CHANGING THE LIGHTING MOOD

6.5 PRECISE TOUCH INPUT FOR COLOR CHANGE

6.6 TEMPLATE PICKING BY GESTURE RECOGNITION

The most casual form of interaction with the bracelet is by drawing gestures in the air to trigger basic operations, e.g. switching a light source on or off. These gestures are recorded by the bracelet's accelerometer and processed using the "3$ Gesture Recognizer" [9], an extension of the popular "1$ Recognizer" by Wobbrock et. al [15]. The algorithm is explained in detail in the following paragraphs.

After a gesture is recorded, it is resampled to a fixed number of points. If the gesture was drawn quickly, it would have less samples and thus less points compared to a slowly drawn gesture. In order to be able to compare these two gestures, a resampling is performed before further processing. The length of the gesture path M is calculated and an increment size I derived by dividing M by $(N/1)$, where N is the desired number of samples. The gesture path is stepped trough and after each distance I, a new point is inserted using linear interpolation (cf. listing 1).

Listing 1: Resampling of a points path into N evenly spaced points

```
def resample(points, N):
        I = path_length(points) / (N - 1)
        D = 0
        newpoints = points[:1]
        for i in range(1, len(points)):
                dist = distance(points[i-1], points[i])
                if(D + dist) > I:
                        q = points[i-1] + ((I - D)/dist) * (
                            points[i] - points[i-1])
                        newpoints.append(q)
                        D = distance(q, points[i])
                else:
                        D = D + dist
        if(D + dist) > increment: #append last point manually
                q = newpoints[-1] + ((increment - D) / dist) * (
                    points[i] - newpoints[-1])
                newpoints.append(q)
        return newpoints
```

In the next step, the gesture is prepared for matching against the templates by rotating it to a specific position. The so called *indicative*

angle θ between the gesture's centroid C and its first point is calculated using the normalized scalar product of the centroid and the first point's position vectors. Afterwards the gesture is rotated so that θ is at 0°. Wobbrock et. al. do this by using the inverse tangens function, however this is not possible in 3D space. Hence, *Rodrigues' rotation formula* (named after French mathematician Olinde Rodrigues) was implemented instead. This efficient algorithm for rotating a vector in $\mathbb{R}^3$ takes the rotation angle θ as well as the axis unit vector k as input, and calculates the following formula:

$$v_{rot} = v\cos\theta + (k \times v)\sin\theta + k(k \cdot v)(1 - \cos\theta). \text{ [7]}$$

After applying this function on the gesture, a good starting point is created for the actual recognizing part of the algorithm. Listing 2 illustrates the rotation procedure.

Listing 2: Rotation of points so that their indicative angle is at 0°

```
def rotate_to_zero(points):
        c = centroid(points)
        theta = acos(points[0] * c / (|points[0]| * |c|)
        newpoints = rotate_by(points, -theta)
        return newpoints
```

In order to harmonize gestures of different sizes, the points are scaled to a reference cube with edge length of 100 units and translated so that the respective centroid C is on the origin (cf. listing 3).

Listing 3: Scaling to reference cube and translation to origin

```
def scale_and_translate(points, size): # size=100
        B = Bounding_Box(points)
        newpoints = []
        for p in points:
                q = Point()
                q.x = (p.x * (size / B.width)) - c.x
                q.y = (p.y * (size / B.height)) - c.y
                q.z = (p.z * (size / B.depth)) - c.z
                newpoints.append(q)
        return newpoints
```

After these harmonization and preparation steps, the gestures are matched against the prerecorded templates (cf. listing 4). The aforementioned steps are applied to templates as well as to newly recorded gestures, the following algorithms only apply to unrecognized gestures.

Listing 4: Matching candidate gesture against every template

```
def recognize(points, templates, rescale_size):
        theta_min = -180
        theta_max = 180
```

```
        theta_delta = 2

        best = float("inf")
        for t in templates:
                dist = distance_at_best_angle(points, t,
                    theta_min, theta_max, theta_delta)
                if dist < best:
                        best = dist
                        t_best = t
        return t_best
```

The candidate is compared to all stored templates using the average Mean Square Error (MSE) as a scoring metric. The optimal position between the candidate gesture and a stored template is possibly offset by a certain rotation, so the optimal combination of angles between the two gestures needs to be determined. Since rotations are costly in terms od computation time, the candidate gesture should be aligned to the template in as few tries as possible. Hence, as proposed by [9], a Golden Section Search (GSS) is used to find the optimal angles α, β, and γ for rotation around the three axis of the coordinate system.

The GSS algorithm was invented by US-American statistician Jack Kiefer in 1953 and is conceptualized to find the minimum value x^* of a unimodal function $f(x)$ in a given interval $[a, b]$ [6]. Two function points $f(x_1)$ and $f(x_2)$ with

$$x_1 = a + (1 - \phi)(b - a) \qquad x_2 = a + \phi(b - a)$$

are calculated and compared, where $\phi = 2/(1 + \sqrt{5})$ denotes the Golden Section constant. If $f(x_1) < f(x_2)$, the interval for the next iteration becomes $[a, x_1]$ and the new test points are calculated as described above with respect to the new interval borders. Note that the value $f(x_1)$ can be reused [4]. If $f(x_1) > f(x_2)$, the interval for the next iteration step changes to $[x_2, b]$ respectively. If the function points differ not more than a given x_Δ, the current minimum will be accepted as the algorithm's result.

In the context of the implemented gesture recognizer, the function to be minimized is the distance between a gesture candidate and a certain template at the best angle. Since the recognizer works with three-dimensional data, there is not a single angle for rotation, but one fir each coordinate axis. Hence, the GSS needs to be adapted for three dimensions.

As listing 5 illustrates, the three-dimensional approach is similar to the one-dimensional algorithm. Each of the variables α, β, γ has its own search interval and a pair of calculated values. This leads to eight function points to be evaluated in each iteration, denoted in the code by variables `f1` to `f8`. The minimum of these values is calculated and the intervals adjusted accordingly. Note that in every case one function value from the previous step can be carried over,

for example when `f1` is the minimum, its value is used for the new `f8`.

Listing 5: Three-dimensional Golden Section Search for finding the best angle between a candidate gesture and a template

```
def distance_at_best_angle(p, t, theta_min, theta_max,
    theta_delta):
        phi = 0.5 * (-1 + math.sqrt(5))
        a_min = theta_min
        a_max = theta_max
        b_min = theta_min
        b_max = theta_max
        g_min = theta_min
        g_max = theta_max

        x1 = phi * a_min + (1 - phi) * a_max
        x2 = (1 - phi) * a_min + phi * a_max
        y1 = phi * b_min + (1 - phi) * b_max
        y2 = (1 - phi) * b_min + phi * b_max
        z1 = phi * g_min + (1 - phi) * g_max
        z2 = (1 - phi) * g_min + phi * g_max

        f1 = distance_at_angle(p, t, x1, y1, z1)
        f2 = distance_at_angle(p, t, x1, y1, z2)
        f3 = distance_at_angle(p, t, x1, y2, z1)
        f4 = distance_at_angle(p, t, x1, y2, z2)
        f5 = distance_at_angle(p, t, x2, y1, z1)
        f6 = distance_at_angle(p, t, x2, y1, z2)
        f7 = distance_at_angle(p, t, x2, y2, z1)
        f8 = distance_at_angle(p, t, x2, y2, z2)

        while (|a_max - a_min| > theta_delta) or (|b_max - b_min|
             > theta_delta) or (|g_max - g_min| > theta_delta):
                min_f = min(f1, f2, f3, f4, f5, f6, f7, f8)
                if min_f == f1: #x1, y1, z1
                        a_max = x2
                        x2 = x1
                        x1 = phi * a_min + (1 - phi) * a_max
                        b_max = y2
                        y2 = y1
                        y1 = phi * b_min + (1 - phi) * b_max
                        g_max = z2
                        z2 = z1
                        z1 = phi * g_min + (1 - phi) * g_max
                        f8 = f1
                        f1 = distance_at_angle(p, t, x1, y1, z1)
                        ...
                        f7 = distance_at_angle(p, t, x2, y2, z1)
                elif min_f == f2: #x1, y1, z2
                        ...
                else: #x2, y2, z2
                        ...
```

```
        return min(f1, f2, f3, f4, f5, f6, f7, f8)
```

To determine the distance between a gesture and a certain template at given angles α, β, and γ, the gesture is rotated using the Rodrigues rotation formula discussed in the context of the indicative angle above. The rotation formula needs an axis and an angle for executing the rotation, but the GSS works with three angles. Therefore, the rotation matrix formed by the angles needs to be transformed into a Euler axis/angle pair for use in the implemented rotation formula. The authors of [9] state the following rotation matrix in their reference implementation [8]:

$$A = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}$$

The Euler rotation angle θ is composed by the matrix's diagonal elements.

$$\begin{aligned} \theta &= \arccos\left(\frac{1}{2}(A_{11} + A_{22} + A_{33} - 1)\right) \\ &= \arccos\left(\frac{1}{2}(\cos\alpha\cos\beta + \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma + \cos\beta\cos\gamma - 1)\right) \end{aligned}$$

The rotation axis vector e can be derived from A and θ as follows:

$$\begin{aligned} e_1 &= \frac{A_{32} - A_{23}}{2\sin\theta} \\ &= \frac{\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma - \cos\beta\sin\gamma}{2\sin\theta} \\ e_2 &= \frac{A_{13} - A_{31}}{2\sin\theta} \\ &= \frac{-\sin\beta - \cos\alpha\sin\beta\cos\gamma - \sin\alpha\sin\gamma}{2\sin\theta} \\ e_3 &= \frac{A_{21} - A_{12}}{2\sin\theta} \\ &= \frac{\cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma - \sin\alpha\cos\beta}{2\sin\theta} \end{aligned}$$

An so, the angles determined in algorithm 5 lead to an axis/angle rotation which is performed on the candidate gesture.

Once the best template for a candidate gesture is found, the corresponding index in the stored template list is returned and the recognized gesture triggers certain changes in the light source's color or brightness. Note that this gesture recognition algorithm is only used for recognition of the preset gestures formulated in section XX.

6.7 PRESETS AND CONFIGURATION

Gesture recording and recognition is triggered by a gentle double tap on the bracelet. This small but focused activation reduces unwanted

REGISTER NAME	PARAMETER	VALUE
PULSE_THSZ	Tap Detection Threshold	100g
PULSE_TMLT	Interval between Start and End Pulse	6.25ms
PULSE_LTCY	Ignore Interval after Detection	25ms
PULSE_WIND	Maximum Double Tap Interval	500ms

Table 2: Single- and double tap detection configuration for the MMA8652FC digital accelerometer

triggering of the gesture recognition process, e.g. while gesturing heavily, and thus reducing false positives. The tap detection functionality is a built-in feature of the bracelet's accelerometer, configuration parameters for this process are listed in table 2.

In order to be recognized as a tap interaction, the initial impulse needs to be at least XX g in intensity. When calibrated like this, jerky movements like suddenly raising the hand at a high speed are correctly not recognized as a tap. However since the threshold is that high, the activation tap needs to be executed directly on the hardware which is located on the inner wrist.

The PULSE_TMLT register configures the maximum time interval between the impulse exceeding the threshold on the Z axis and falling back under said threshold. If the mentioned interval lasts at most 6.25ms, the interaction is considered as a tap.

After a tap is detected, all impulses in the following 25ms are ignored by the detection mechanism. This prevents bouncing effects and detecting multiple taps in a singe tap movement.

The MMA8652FC accelerometer is able to distinguish between single and double taps. The last configuration register listed in table 2 is a parameter for double tap detection. It specifies the maximum time interval between two double taps and is set to 500ms, the same time interval as the Windows default between two mouse clicks of a double click [10].

7 EVALUATION

7.1 STUDY DESIGN

7.2 RESULTS

7.3 DISCUSSION

8

CONCLUSION AND FUTURE WORK

BIBLIOGRAPHY

[1] 3D Systems, Inc. ProJet x60 Series - Professional 3D Printers, 2014. URL http://www.3dsystems.com/about-us.

[2] 3D Systems, Inc. About 3D Systems, 2014. URL http://www.3dsystems.com/about-us.

[3] Arduino. Arduino Uno, 2014. URL http://arduino.cc/en/Main/ArduinoBoardUno.

[4] Yen-Ching Chang. N-dimension golden section search: Its variants and limitations. In *Biomedical Engineering and Informatics, 2009. BMEI'09. 2nd International Conference on*, pages 1–6. IEEE, 2009.

[5] Freecad. An open source parametric 3D CAD modeler, 2014. URL http://freecadweb.org.

[6] Jack Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953.

[7] Don Koks. *Explorations in Mathematical Physics*. Springer Science+Business Media, 2006.

[8] Sven Kratz. Google code: $3 gesture recognizer, June 2010. URL https://code.google.com/p/three-dollar-gesture-recognizer.

[9] Sven Kratz and Michael Rohs. A $3 gesture recognizer: simple gesture recognition for devices equipped with 3D acceleration sensors. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 341–344. ACM, 2010.

[10] Microsoft Corporation. Windows Dev Center - Desktop: GetDoubleClickTime function, 2014. URL http://msdn.microsoft.com/en-us/library/windows/desktop/ms646258.aspx.

[11] Daniel Muschke. Amico Bracelet, June 2012. URL http://grabcad.com/library/amico-bracelet--9.

[12] Roving Networks. RN-42 Class 2 Bluetooth Module, April 2013. URL http://ww1.microchip.com/downloads/en/DeviceDoc/rn-42-ds-v2.32r.pdf.

[13] E.M. Sachs, J.S. Haggerty, M.J. Cima, and P.A. Williams. Three-dimensional printing techniques, August 23 1994. URL http://www.google.com/patents/US5340656. US Patent 5,340,656.

[14] Paul Stoffregen and Robin Coon. Teensy usb development board, 2014. URL https://www.pjrc.com/teensy/index.html.

[15] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168. ACM, 2007.

EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die wörtlich oder inhaltlich aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Hannover, December 2014

Karoline Busse