Pinstripe: Integration & Evaluation of a Wearable Linear Input Controller for Everyday Clothing

Bachelor Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Jan Thar

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr.-Ing. Stefan Kowalewski

Registration date: 1/10/2012
Submission date: 1/2/2013

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, January 2013
Jan Thar

Contents

List of Figures

List of Tables

Abstract

This bachelor thesis integrates and evaluates pinstripe, a wearable input device for everyday clothing. With the upcoming of more and more wearable devices—music players, smartphones, display glasses—it is desirable to allow an effortless interaction with these devices.

Pinstripe allows such an interaction: Making a fold in clothing and moving it to control a device could be done on the side and without looking. With only a single micro-controller and the sensor areal woven of conductive thread, it could be seamlessly integrated in clothing, invisible from the outside.

In this thesis, hardware prototypes were made and the evaluation software was reworked, which will be described in 3—"Hardware Prototyping and Software Modification". Furthermore, prototype and software were tested both in a user study to evaluate the system and in daily use to detect weak points. The results will be presented and discussed in chapter 4—"Evaluation". We will conclude the thesis with a summary and an outlook on future work.

Überblick

Diese Bachelorarbeit überarbeitet und bewertet Pinstripe, einen tragbares Eingabegerät, welches in die Kleidung integriert wird. Da mehr und mehr tragbare Geräte verwendet werden—MP3-Spieler, Smartphones usw.—sind nicht weiter ablenkende Interaktionsmöglichkeiten mit diesen Geräten wünschenswert.

Pinstripe ermöglicht eine derartige Eingabe: Eine Falte in der Kleidung bilden und diese bewegen, um ein Gerät zu bedienen kann nebenbei und ohne optische Kontrolle geschehen. Da nur ein einziger Mikrocontroller verwendet wird, und die Sensorfläche aus gewebtem, leitfähigem Garn besteht, kann dieses System gut in die Kleidung integriert werden, ohne sichtbar zu sein.

In dieser Arbeit wurden sowohl Hardwareprototypen erstellt und die Auswertungssoftware überarbeitet. Dies wird in 3—“Hardware Prototyping and Software Modification” beschrieben. Anschliessend wurde Prototypen und Software einmal in einer Benutzerstudie, um das System zu bewerten, und andererseits als Dauertest im täglichen Einsatz verwendet, um (hardwareseitige) Schwachstellen zu finden. Die Ergebnisse werden in Kapitel 4—“Evaluation” präsentiert. Die Arbeit schliesst mit einer Zusammenfassung und einem Ausblick.

Acknowledgements

First of all, I want to thank Prof. Dr. Borchers for supervising my thesis and working on Pinstripe at his chair, and Prof. Dr.-Ing. Kowalewski for being my second examiner.
Secondly, I want to thank Florian Heller for advice, patience (especially all the times when my prototypes stopped working while demonstration) and feedback.
And of course I want to thank the participants of the user study, and everyone else who supported me.

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 British English.

Download links are set off in coloured boxes.

File: myFile[a]

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

Chapter 1

Introduction

Wearable Computing

Technical advantages in miniaturization enable smaller and smaller electronic devices, leading to everyday usage but disappearing computing as described in Weiser [1995]. One example is the wearable music player—compared to the famous Sony Walkman, mp3-players are nowadays far smaller and robust, run longer and have greater storage capacity.

Language control

This eventually leads to small players without buttons at all, but only with a remote only on the headphone cable like the third generation iPod Shuffle. While later versions return to buttons on the player because of better usability, the input on the device itself has its own issues: One has have to take the device and find the controls, and maybe—to prevent unintentional input—first unlock the controls before they can be used.

Cable remote

To offer a more effortless usability, many mp3-players and smartphones additionally offer a remote control on the headphone cable. This allows an eyes-free interaction with the device by getting the cable, feeling for the remote and then pressing, for example, on the according end of the remote. This is easier than taking the device out of the pocket and finding the buttons after unlocking, but allows only few commands. The cable itself may disappear in the future, as in bluetooth headsets, rendering this kind of remote control not useful anymore.

Design rules

This demands new wearable input control devices which should fulfill physical and social constraints to be useful. These constraints are size and movement of the body, interaction with the device, and its size, form and weight. Additionally, aesthetics and how to attach the device on the body are important points. The basic guidelines to meet these constraints are described in Gemperle et al. [1998].

Electronic textiles

Marculescu et al. [2003] augmented clothes with electronic devices . Since they are tailored to meet the individual aesthetics and are made to be worn, the last two points are easily met. By using light, flexible, and low cost components the resulting device will be unobtrusive.

On the other hand, this leads to new constraints: The components and their interconnections have to be washable and ready for daily-use. They use copper wire within yarn as a kind of bus system, either wire-bonded or soldered on flexible foil with copper pads with the single devices.

Since persons do not forget their clothes, the system is ready to be used anywhere and anytime, and a well designed interface should allow anybody to use it without further instructions.

Smart suit

Toney et al. [2003] made an augmented business suit. They use vibration motors and LEDs on cuffs as direct feedback in the suit, connected with capacitive input controls and a wristwatch as processing and output device.

This leaves the electronic system invisible, and input should resemble adjusting the jacket or the cuffs. In best case, the user seems to use no technology either at all or just conventional technology like looking on his wristwatch.

The capacitive buttons used in this study were located inside the hems, near the natural resting position of the hands and should be used eye-less, with perceptible different forms of the buttons. Therefore, the system has to decide if a detected fingertip on a button is just the search for the right button or actual a button-press. Furthermore, the system must suppress capacity changes due to the coupling with the human body itself.

Pinstripe

With pinstripe Karrer et al. [2011] proposes another approach. The directional movement of the fold of a textile—independent of the concrete position—determines the input. Also crumpling the textile is another input possibility. Similar as in the augmented business suit before, using pinstripe within the cuffs or the tie seems to be just an adjustment of the clothing.

The fold is detected through interconnection between conductive stripes on the back of the clothes, leaving the device invisible from the outside. A flexible, woven surface of the sensor array and a low weight and energy consumption are further benefits for using pinstripe as a wearable input device.

In this thesis, both hardware prototype and software is reworked to let pinstripe work as a standalone wearable device with low energy consumption. The resulting prototype is then tested in personal daily-use to detect hardware issues in the long-term run. Additionally a user study validates if that the system is usable without training and which kind of functions may be best controlled with pinstripe.

Chapter 2

Related work

In this part of the thesis we will describe the background for pinstripe. We will present work related to the hardware prototype—especially for the interconnection between the different parts (rigid, flexible, textile)—first. Then the different types of wearable input devices follow: From textile variants of normal buttons to the more advanced, textile-specific and eyes-free types thereafter.

Almost all paper describe both technological background and implementation in demonstrators and conduct a study. The following short descriptions will focus mostly on one aspect relevant for this thesis.

2.1 Interconnections and conductive textiles

One major concern of textile devices is the flexible printed circuit board (PCB): It should feature the same flexibility and durability as normal textile. However, even with the flexible pcbs nowadays used in electronic devices, the durability is limited. Bending and relaxing over time causes eventually a disconnection of the copper layer and therefore the circuit paths.

Therefore, textile interfaces use a combination of both conductive yarn, which could result in a woven conductive textile, flexible PCBs and rigid components, which have to be connected somehow.

Definition: *Conductive yarn and fabric*

CONDUCTIVE YARN AND FABRIC:
Conductive yarn can be produced by either inserting conductive foil or fibres of conductive material, for example stainless steel, within the common not conductive fibres of the yarn. Another variant is coating yarn with a conductive layer. The resulting yarn has—apart from being conductive—similar characteristics as normal yarn, and can therefore be used in a likewise fashion. This includes especially sewing and weaving, resulting in conductive fabric.

A Construction Kit for Electronic Textiles

For example, Buechley [2006] used laser cutted Sn/Cu coated fabric glued together by a heat activated adhesive with a non conductive carrier material. This fabric can be soldered as a traditional PCB or connected with each other with conductive fabric. To test durability, four washing cycles were applied, which all connections survived.

Stretchable Circuit Boards using sinusoidal patterns

Vieroth et al. [2009] laminated a copper foil on polyurethane foil, which is later photolithographicly etched to form stretchable circuit paths on clothing, in this case a dress with blinking LED, to connect discrete components on the clothing. To prevent breaking of the copper path, sinusoidal patterns were used instead of straight paths. The transition between the flexible PCB and the unflexible components was mediated through an enlarged copper area, in addition to encapsulation.

Embroidering Electrical Interconnects with Conductive Yarn for The Integration of Flexible Electronic Modules into Fabric

Linz et al. [2005] used conductive yarn as interconnection between components on flexible PCB, combining the two before-mentioned variants. Using large pads on the flexible substrate and sewing the conductive yarn several times through the holes in these pad improve conductivity. An encapsulation with molding of the interconnection press the yarn on the surface. Furthermore, making the holes in the flexible PCB with a needle while sewing improves reliability compared to prepared holes.

Finally, Post et al. [2000] compared different conductive yarns made (partly) of stainless steel and the different methods to interconnect: soldering, bonding with conductive adhesives, stapling and joining. The last one is the direct connection of a thread with one pin of an electronic component over a bonding gold wire, melt together with each other.

E-broidery: Design and fabrication of textile-based computing

2.2 Types of input devices—Buttons

One approach of a textile input device is the transfer of a press button to the fabric. This may seem simple, but the technological constraints make this transfer complicated. The two common approaches use either a resistive or capacitive technology:

Lee et al. [2010] used a multilayer structure (which is similarly used by Buechley [2006] with textile layers): Structures are printed on a flexible substrate, and the space between two unconnected conductive layers, which is made by a distance layer, can be bridged by pressing the conductive layers together, forming a connection and closing the circuit. This is similar to a standard resistive button.

Arm-Band type Textile-MP3 Player with Multi-layer Planar Fashionable Circuit Board (P-FCB) Techniques

Komor et al. [2009] described another resistive approach: Here the skin resistance bridges the gap between two electrodes. To improve conductivity, an interdigital electrode can be used. The form of the electrodes can be made different to distinguish different buttons through contact, using a different button for either searching or activation, but the buttons remain viewable in contrast to the version before.

Is It Gropable? – Assessing the Impact of Mobility on Textile Interfaces

Finally, Holleis et al. [2008] described capacitive buttons: The finger above such a button changes the electric field and the capacitance, therefore detecting a button-press. These kinds of buttons can be invisible. In a user study, these kinds of buttons are compared in three forms: invisible, with a small ornamental hint, and visible. While the invisible buttons look best, they were less usable.

Evaluating Capacitive Touch Input on Clothes

Conclusion: Location needed

Since the input device should be invisible, the buttons on the other hand need some kind of—at least tactile—hint for their placement.

2.3 Types of input devices—Add-On

Instead of just applying buttons to the clothing, it may be useful to modify special pieces of clothing—like the Keyglove Wearable Input Device[1] by Jeff Rowberg. This device transforms a glove into a replacement for both keyboard and mouse by applying conductive sensors to fingers and palm and accelerometer. This approach is to complex for the use as a mobile interface, which controls only simple devices.

Smart Clothing for the Arctic Environment

An easier interface was described by Rantanen et al. [2000]. The so-called Yo-Yo interface was developed to be used in harsh environment and for survival garment. A winding mechanism with rotation encoder applied on their survival vest is connected with a small display unit. Moving this unit back and forth allows to navigate within a menu on the display, selecting it by pressing the whole unit. This provides a robust, easy to use interface, but with the need of additional (optical) feedback.

Cord Input: An Intuitive, High-Accuracy, Multi-Degree-of-Freedom Input Method for Mobile Devices

Schwarz et al. [2010] used an augmented cord as an input device: A conductive thread on the surface determines the location, a rotary encoder on the end detects twisting and finally a stretch sensor is triggered if the cord is pulled by the user. Each input can be used independently or combined and allows eyes-free interaction.

Conclusion: No seamingless integration

While some clothes have cords applied and therefore allow to seamlessly integrate such a device, most of these cords have their functionality, making the input device non-functional if the cord function itself is needed. Adding an additional cord contradicts the invisibility aspect, and in connection with the usage of it, may irritate other people.

[1]www.keyglove.net

2.4 Types of input devices—Eyes-free

In the optimal case, the input device has to be integrated invisibly in the textile, with a big sensor areal to allow eyes-free interaction without searching for a specific input point and no need for optical hints on the clothing itself. Again, several possibilities for such a device exist.

Wearable Sensor Badge & Sensor Jacket for Context Awareness

Farringdon et al. [1999] integrated several stretch sensors on strategic places on a jacket. These stretch sensors—which are knitted strips—measure the resistance change proportional to the change of the length of the sensor field. This allows the system to track movements of the user. A stretchable base material is necessary, otherwise an applied stretch sensor would not work. This reduces the application range of this kind of sensor.

On Body Capacitive Sensing for a Simple Touchless User Interface

Another possibility is the usage of arrays of sensors like the buttons before to achieve a greater sensor area. Cheng et al. [2008] proposed a linear field of capacitive sensors on a doctor's coat as eyes-free and touch less input device, which should enable the doctor to control devices eyes-free and even without physical contact, reducing the possibility of contaminations.

GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices

Another capacitive input device by Rekimoto [2001] expands this idea further to a matrix area, which allows the system to detect gestures by a layer of electrodes below clothing. Two additional layers reduce unintentional coupling of the body with the sensor signal, stabilizing the signal.

Nevertheless, capacitive measurement remains prone to errors: The distance between back electrode and sensor electrode is small compared to the distance of (input) finger and sensor electrode. Therefore, the signal is weak, and movements of the flexible form may have greater influence on the capacity than the intentional input. Furthermore, energy consumption for advanced filtering, computation and the active sensor field reduces usability of these concepts.

TextiPad: Implementation and Evaluation of a Wearable Textile Touchpad

Another pad to detect gestures on a flexible area was made by Ivanov [2012]. It uses a piezo-resistive foil between electrodes to detect the position of a fingertip on the surface. The basic design of such a piezo-resistive foil is shown by Hannah Perner-Wilson in instructables[2] .

Since a point pressure is detected, disturbances through body or textile movement like in the capacitive variants are not possible. On the other hand this kind of usage is still little more obtrusive than just swiping above a surface.

Pinstripe: Eyes-free Continuous Input on Interactive Clothing

Pinstripe (Karrer et al. [2011]) offers a different approach: A textile with conductive stripes is seamed below the clothing. These stripes are used to detect folds in the textile and their movement. Since the absolute position of the fold is not evaluated, only size and movement, and a big sensor areal is used, the user did not need to fold an exactly located position on the cloth, allowing the effortless usage without looking. Furthermore, pinching the garment and rolling the fold are natural gestures. This concept is further reworked in this thesis.

[2]http://www.instructables.com/id/EJKTF3WGV490JGK

Chapter 3

Hardware Prototyping and Software Modification

In this chapter we first describe the general concept used to program and rework the hardware prototype. In the second part different hardware layouts are shown, either different prototypes with their individual advantages and disadvantages itself or additional hardware used for programming or user studies. Finally, the software for the measurement and evaluation of the resulting data is described.

3.1 System Design

The pinstripe prototype could work either with an Atmega or MSP430 micro-controller. While first prototypes before this thesis use the Atmega for rapid prototyping with the Arduino modules, the latest prototype before uses an MSP430.

MSP430 or Atmega are both usable

Both micro-controller families have their benefits: While the TSSOP28 package of the MSP430G2553 allows a smaller layout than the TQFP32 package of an Atmega168/328, the need of an external pull-up resistor for the reset input and only two pin-change interrupt capable I/O channels complicate this layout. On the other hand, the benefit of a lower energy consumption of the MSP430 is partly compensated by the need of changing four registers instead of three (Atmega) for the measurement of each stripe. In major parts both of them are quite similar: They work well with their internal oscillator and have a similar number of outputs.

Since energy consumption is not a major concern in the final prototype—most of the time the micro-controller will sleep, and a similar one-sided layout is made for both micro-controller, the decision, which micro-controller to use is up to the user.

Programming environment: Arduino/Energia IDE and Processing

We chose the Arduino IDE[1] as programming environment for the Atmega. In combination with an Arduino Duemilanove, this allows a simple testing of the functionality. The standalone Atmega based pinstripe versions where later programmed via Arduino as an in-system programmer. For this purpose, the hardware MISO/MOSI/SCK and RESET pins have to be connectable from the outside—which could be done with crocodile clamps on the stripes, if needed. The serial connection in combination with the according Arduino boot loader would need fewer connections, but since the boot loader has to be programmed at least once, it was easier to make an ISP connection and use it for both programming steps. To allow this change, the main function in the Arduino IDE is overloaded, which may be deprecated in future versions.

A clone of this, the Energia IDE[2] , was used with a TI Launchpad with a MSP430G2553[3] as testing device, which could also be used as a programmer for Spy-by-Wire capable MSP430. Here, only the RESET and TEST pin have to be connectable for programming purposes. A boot loader is not necessary.

Additionally, Processing[4] was used to program an interface on a computer for a user study to test the device.

[1]http://www.arduino.cc/
[2]https://github.com/energia/Energia
[3]http://www.ti.com/ww/en/launchpad/msp430_head.html
[4]http://processing.org/

General Program design

To allow the program to work with different stripe layouts and micro-controllers, the micro-controller in use and the mapping of its I/O pins to the stripes is stored in one define block for each layout and has to be chosen manually. In the program, defines are used most of the time instead of function calls, to prevent unnecessary jumps during execution and to leave out unnecessary functions.

All hardware specific commands are summarized in an according header file, which could be easily changed according to the used micro-controller. This includes timer or pin-change interrupts for energy saving behavior, changing the register content of the I/O-pins, sleep and wakeup functionality and initialization of the micro-controller (like disabling non-used modules or enabling the reset pull-up resistor for an Atmega). The needed header file is then included within the above mentioned stripe layout.

After a hardware-dependent initialization the pinstripe micro-controller will sleep until a short circuit between stripes is detected, then measure all stripes, filter the data and evaluate them. This will be discussed later in 3.3—"Software".

Software vs. Hardware Serial

If a command is determined, it is sent to the connected device through the serial port. For these purpose a software serial communication is used instead of the hardware version. This has a number of benefits:

- Each pin could be used for the serial port, the layout of pinstripe is simplified and the serial port could use pins without pin-change interrupt capability.
- An additional channel for debugging is provided at a micro-controller with only one hardware serial channel.
- Only a sending pin is used instead of both sending and receiving, therefore the number of needed pins is reduced, and the (more complex) receiving functionality is disabled. This is only a benefit for the Atmega version, the MSP430 allows the use of the hardware sending pin alone.

The disadvantage of higher energy consumption is neglectable. Compared to the overall program cycle times, serial communication is rare and therefore the impact on the general energy consumption is small. Sent commands are furthermore only few bytes long—and even then, a timer enabled sleep over few clock-cycles could be done with a timer triggered sleep function.

3.2 Hardware

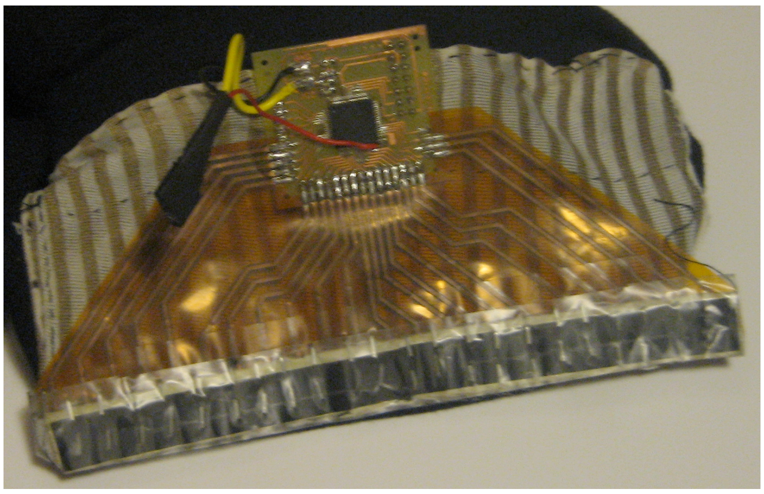
Figure 3.1: The original MSP430 version of pinstripe. On the top the rigid PCB with the micro-controller, below the flexible PCB as interconnection and in the background the stripes, made of conductive garment, which are connected with the black t-shirt textile.

Previous MSP430 Prototype

The existing hardware prototypes use either conductive thread or glue (as the prototype in figure 3.1) to connect the micro-controller via a flexible PCB with the pinstripe textile and the conductive stripes.

Disadvantage: Broken circuit paths, clumsy

Conductive thread seems to loose conductivity over time and the conductive glue allows only a bad conductivity. Furthermore the layout of the MSP version did not allow to use pin-change interrupts for sleeping between measurements and uses the bulky standard JTAG interface instead of the two wire Spy-By-Wire interface for programming (although this JTAG could be implemented efficiently by using the stripes as connectors via crocodile clamps, as mentioned before, making an additional JTAG header obsolete). For the Atmega, an integrated micro-controller version did not exist at all.

3.2.1 Shields for testing

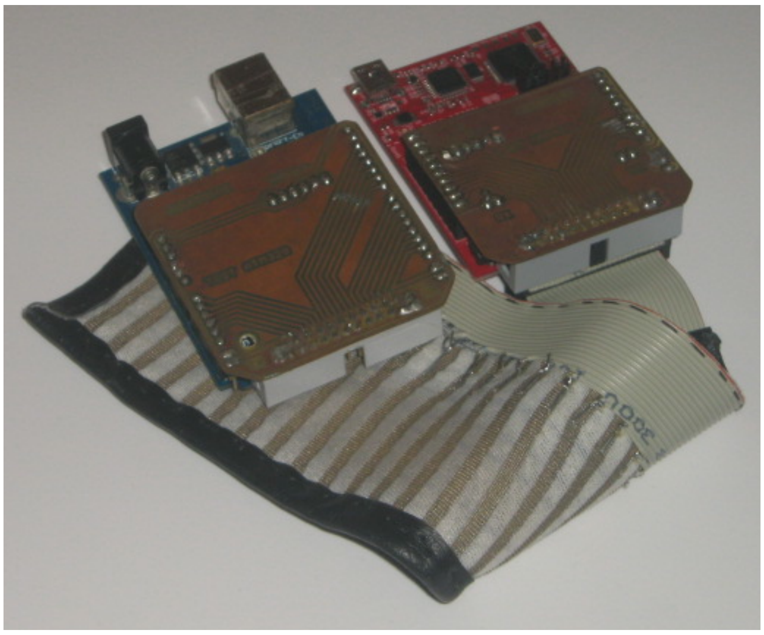

Figure 3.2: Shields for the Launchpad (right) and Duemilanove (left). Connected with the pinstripe garment (below) with staples, soldered on a ribbon cable

Design

To test the software, especially the different micro-controllers, a shield was designed both for an Arduino Duemilanove and the TI Launchpad. Only a reduced number of stripes could be used with the Launchpad (the two channel PDIP version of the MSP430G2553) and the Duemilanove (the TOSCx pins used for an external crystal are not usable). Additionally, the standard micro-controller on the Duemilanove is not configured to use the internal oscillator, and the 5V voltage level did not match with the 3.3V signal level of an iPod, therefore the communication with the iPod could not be tested with this prototype.

Both shields, as shown in figure 3.2, simply route nearly every available general I/O pin—apart from two pins used for debugging and communication—to a connector, which allows the connection with the pinstripe textile through a ribbon cable, shown at the front of the picture. The conductive stripes of the pinstripe textile are then connected with a ribbon cable via staples: Each metal staple makes a mechanical pressure connection with the garment, and is soldered together with one wire. The pinstripe textile is sewed together with a carrier textile to improve mechanical stability of the garment.

Don't use glue with the pinstripe textile

A previous version, where the wires and conductive textile are wrapped did not work properly. In this attempt, glueing the pinstripe textile together with a carrier textile partly isolated the stripes.

3.2.2 Version 1: Conductive thread

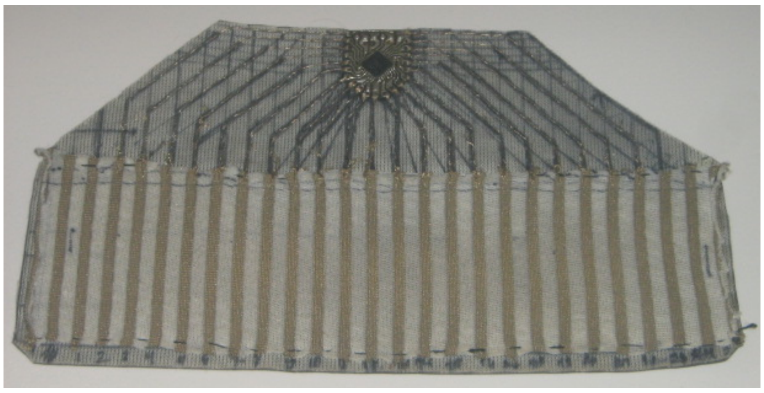

Figure 3.3: The first version—A PCB with an Atmega is connected with the pinstripe textile with conductive thread.

Conductive thread

A small PCB with an Atmega (shown on top of figure 3.3) is connected with the pinstripe textile below with conductive thread, by sewing a conductive path to the PCB, wrapping the thread through a hole at the according circuit path several times before returning to the pinstripe textile, where the start and end of the thread is interwoven. A header on the PCB allows connection with the hardware serial channel, the voltage pins, and the reset pin. Therefore, it is possible to program the Atmega with the boot loader function over the serial line. The boot loader itself can be programmed via crocodile clamps on stripes which represent the remaining ISP-pins on the textile below, or by connecting the according circuit path on the PCB with an ISP connector and program the boot loader before sewing the system together.

Disadvantage: Thread looses conductivity, complex

The disadvantage of this version is that the sewing is relatively complex: each stripe has to be connected independently with the according circuit path on the PCB. Furthermore the thread loses conductivity over time, the interconnection between some stripe and the PCB was lost faster than expected, while loose ends of the thread may cause unwanted interconnections.

3.2.3 Version 2: Flexible PCB

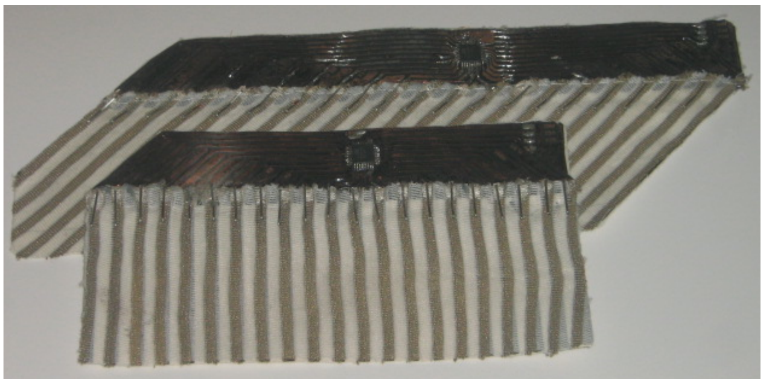

Figure 3.4: The second version—An Atmega put directly on flexible PCB, stapled together with the pinstripe textile. Orthogonally aligned in the foreground and rotated 45° in the background

Flexible PCB

Using the toner transfer method, e.g., a flexible PCB was etched to directly solder a micro-controller on it. The interconnection with the pinstripe textile was made mechanically with pads, pressed together with the conductive stripes of the textile by staples.

Definition: *Example toner transfer method*

EXAMPLE TONER TRANSFER METHOD:
A mirrored layout, printed with a laser printer on transparencies, special transfer paper or even plain paper, is applied on a PCB: As far as we have tested, using transparencies[5] worked well: Apply the printed side of the transparency to the copper of a fine sanded and cleaned—and optional shortly etched—PCB. After this, let an iron stand on top of it for 3 or 4 minutes and full power. Thereafter, rub in the layout for the same time by moving one edge of the iron in small circles and with soft pressure (high pressure may move the molten toner and prevent a sharp layout). After this, directly insert the PCB (the transparency should then stick on top of it due to the molten toner) into cold water, which removes the transparency due to different temperature extension coefficients. Or—if a paper is used—peel off the paper. After this step, control the layout and correct minor parts without toner with a permanent marker or—in case of bigger faults—remove the whole layout from the PCB by sanding or acetone and start over. If the transfer was successful, burn in the layout for additional 3 or 4 minutes on top of the flipped iron. The PCB is now ready for etching.

A second variant of this version uses stripes rotated by 45°. In this case, the folding of the textile will result in a 90° crossing. This reduces the possibility of missing interconnections and therefore produces more stable signals, reducing the error made by unconnected stripes. This attempt was not pursued further, since this benefit was not necessary for the final version and the changed distances between stripes due to the rotation did not allow a direct comparison with the unrotated version: The rotation virtually enlarges the width of both stripe and distance between stripes, therefore enlarges the reaction time of the system and reduces sensibility.

While pads on the PCB near the micro-controller in the middle of the dark PCBs on figure 3.4 remains for the programming (here: ISP pins instead of the serial pins as in the version before) and debugging interface, the iPod connection (serial and voltage) is moved to the side of the prototype (the bright solder pads of the header are seen on the right side of each version): A connection in the middle as in the prototype before causes the connection cable to interfere with the folding, and reduces the usability.

Instead of an Atmega 328 as in the Duemilanove, now a cheaper Atmega 168 is used: According to program size even an Atmega 88 is possible, while the flash memory of an Atmega 48 is too small. The use of an Atmega 8 is not recommended due to the lack of pin-change interrupts. This would prevent interrupt-triggered sleep functions of the micro-controller between measurements.

The pinstripe textile is ironed together with a carrier textile—simulating the t-shirt—with a transfer sheet for t-shirt printing. This meltable plastic glues both of them together, while not isolating the conductive garment like the liquid glue before. With the same method, the pinstripe garment could later be ironed on the target textile.

Disadvantage: Broken circuit paths

While this works well, another design issue has to be removed: The small dimensions of the circuit path—especially near the micro-controller itself—causes broken connections due to bending, rendering this version unusable after a short time.

3.2.4 Version 3: Rigid and flexible PCB

Combine rigid and flexible PCB

In the final version, bigger dimensioned circuit paths on the flexible PCB (seen from the back in figure 3.5—the copper circuit paths are visible through the PCB) and the connection of a flexible and a rigid PCB as micro-controller carrier (in the middle of the flexible PCB) reduces the chance of broken circuit path on each PCB itself.

The overlapping pads between garment and PCB were made bigger and are also pressed together by staples, which increases reliability. Additionally, this allows a disassembly of the pinstripe, if the garment has to be replaced. In a textile version, the staples would be replaced by non-conductive stitching, which will press conductive garment and copper foil together in a similar way. The iPod connector remains on the side of the prototype as in the

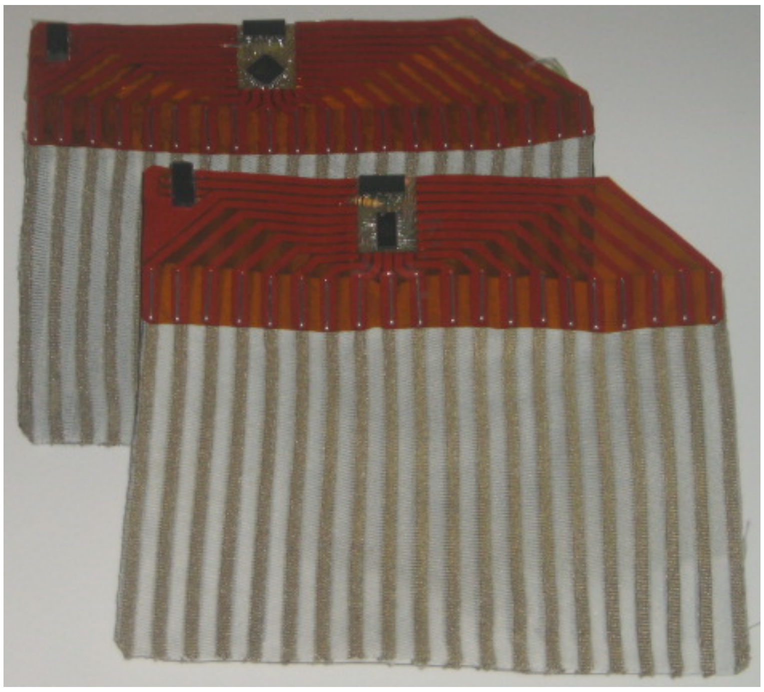

Figure 3.5: The third version—Using both rigid and flexible PCB. In the background is a version with the Atmega168, and in the foreground the MSP430 version

version before. A similar connector is added to the rigid PCB with the remaining programming and debugging pins.

The pinstripe garments were then ironed together with different carrier garments as described before. Five prototypes with the Atmega168 with different textiles—versions without a carrier garment, and with other different surfaces (silk, polyester, cotton, and fly-screen)—were produced for the user study. These final prototypes were then attached to snap-bands to be worn on the arm to simply exchange different textiles.

Another version was build with the MSP430G2553 by exchanging just the rigid PCB part with a different layout. A pull-up resistor for the RESET pin had to be added as an additional element because of the missing programmable pull-up resistor.

Disadvantage: Interconnections

Here, the interconnection between both PCBs is the weak point, where the copper laminate could break due to strong bending, especially if solder is applied. Taping this interconnection for a smoother transition between rigid and flexible may further reduce this—but a broken path only occurs after extreme bendings (crumpling the whole pinstripe prototype) during continuous testing.

3.2.5 iPod connection

iPod connector

Broken cable connections due to pulling was the main cause for failure during continuous testing. Stretching the cable between iPod and pinstripe prototype causes the connector cable to unplug at the prototype either or breaks connection within the connector itself. To improve repairability and to remove this weak point, another small PCB was designed to be soldered in the connector itself, improving the stability of the interconnection. Fixing the cable with hot-glue within the connector additionally reduces failures.

3.2.6 Computer connection

Cable Connection

The Atmega version of the final prototype has three usable I/O pins at the programming header on the rigid PCB (MISO and MOSI, together with RESET) and on the flexible PCB (SCK, together with VCC and GND). To use this pinout, a shield for the Duemilanove was designed, which could be either used as an ISP programmer, for debugging (using a direct connection with a removed jumper between the MOSI pin and TX line of the Duemilanove) or finally as an interface between prototype and computer for the user study.

In this case, the SCK pin is used to send the debug information to the computer, while the two other programmable pins are used to switch between different evaluation functions on the prototype. For this purpose, both pins are used as input pins on the prototype.

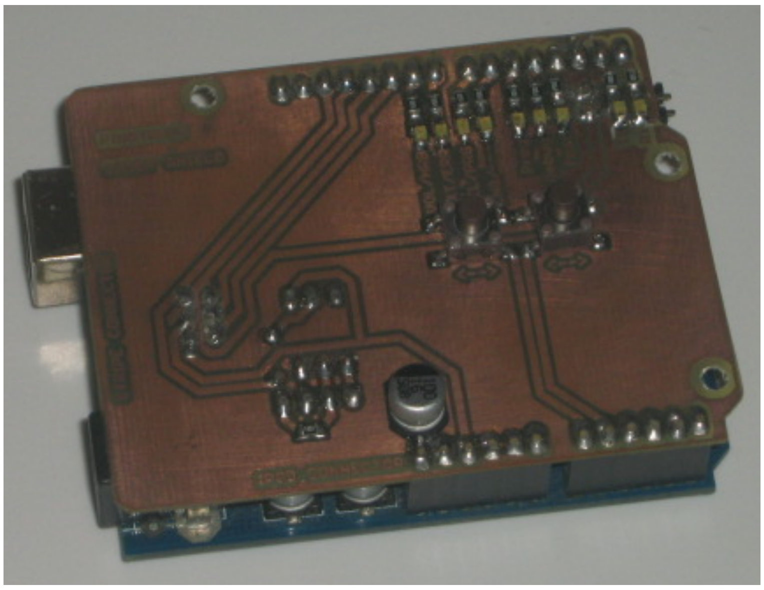

Figure 3.6: Duemilanove shield for the user study and programming

This allows to switch between four different modes according to applied voltage level on these pins.

Two buttons (shown in the middle of figure 3.6) allow to switch between the different functions, which could also be controlled from the computer. The selected function is indicated with LEDs above the buttons. Between power jack and usb connector of the Duemilanove a six-pin connection header is mounted for connections with the pinstripe prototype, while another four-pin connector next to the reset pin could be used for the connection with an iPod—using an opto-coupler for the different voltage levels. To prevent auto resets of the Duemilanove while using it as an ISP programmer, a capacitor is mounted between the reset pin and ground.

For the user study, a modified version with several connectors was made to directly connect all pinstripe prototypes used in the study.

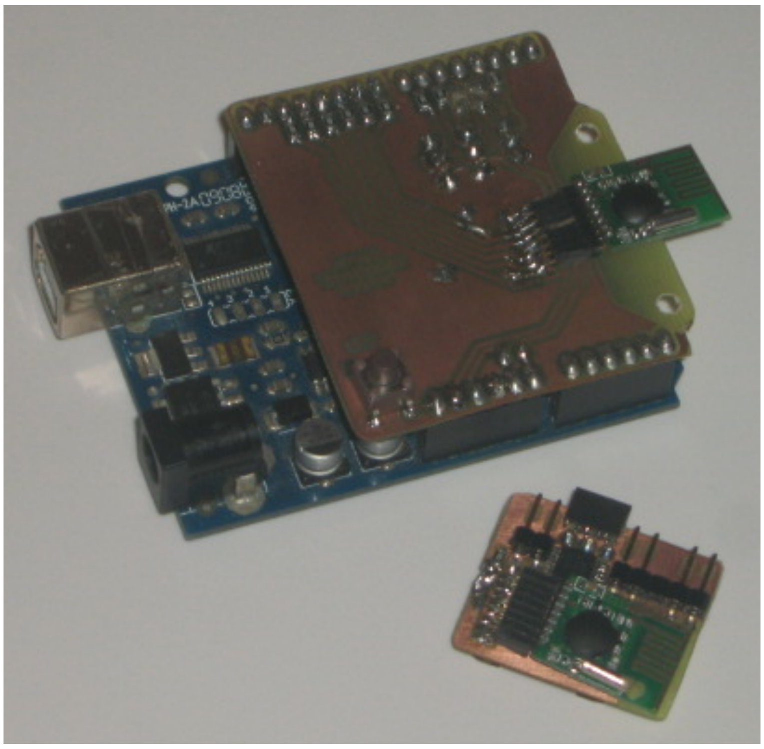

Figure 3.7: Duomilanove with attached RFM70 module (above) and transmitter module (RFM70 and MSP430G2452, below) for connection with pinstripe

Wireless Connection

To control a device without cable, a transmitter PCB, as shown in figure 3.7, was designed. It features a MSPG2452, a RFM70 module, and a battery and can be directly connected with a pinstripe prototype. The MSP in its 14 pin version uses a modification of the rfm70 library[6] by Wouter van Ooijen to control the transceiver module with 6 of the I/O pins, while the other 4—hardware serial and I2C channel—are independently usable, allowing the transmitter module to work also as a standalone device. This MSP is again programmed with the Launchpad as programmer over the two wire interface.

A Duemilanove with another RFM70 transceiver works as receiver station. 5V compatible pins allow a connection with the Duemilanove, while the transceiver itself need a supply voltage of 3.3V.

[6]http://www.voti.nl/rfm70/rfm70-arduino.zip

The RFM70 transceiver was chosen instead of a common XBee module, because of its smaller size and the smaller and simpler header (a single row of eight pins in a 1.27mm grid instead of the two distant rows of 10 pins in a 2mm grid). This allows a very compact design, even a direct soldering of the module on the PCB without header. Furthermore and -most, the RFM70 module only costs part of an XBee module.

Definition: *Transceiver modules*

TRANSCEIVER MODULES:
A transceiver module is used to transmit data with radio frequencies - in this case with the 2.4 GHz band. A common module is the XBee module, which could directly transmit and receive data of a RX/TX connection. This allows a use of the transmitter module instead of a wired serial connection. In combination with the sleep control pin, a three pin layout is possible, while the RFM70 transceiver module needs 6 pins for a four pin SPI connection (MISO, MOSI, SCK, and CSN), one pin for sending or receiving and an interrupt output pin. Different power level (sending power) and channel (carrier frequencies) could be chosen for both modules, and receiving up to six addresses simultaneous allows a radio network (1 to 6 star network) for the RFM70, while the XBee allows direct transmission of analog data, and has further network abilities like broadcasting.

3.3 Software

Previous: Measurement matrix

Each stripe of the sensor field of the pinstripe hardware is connected with an I/O pin of the micro-controller. To determine the position of a fold with the original evaluation software, one stripe at a time is connected to ground (grey in figure 3.8, no measured value for this stripe), while all other stripes are pulled up through the internal pull-up resistors. Each of the other stripes is then measured, and if a ground level is detected, a connection to the grounded stripe exists (blue colored). This procedure is repeated for the next stripe until all stripes are once grounded.

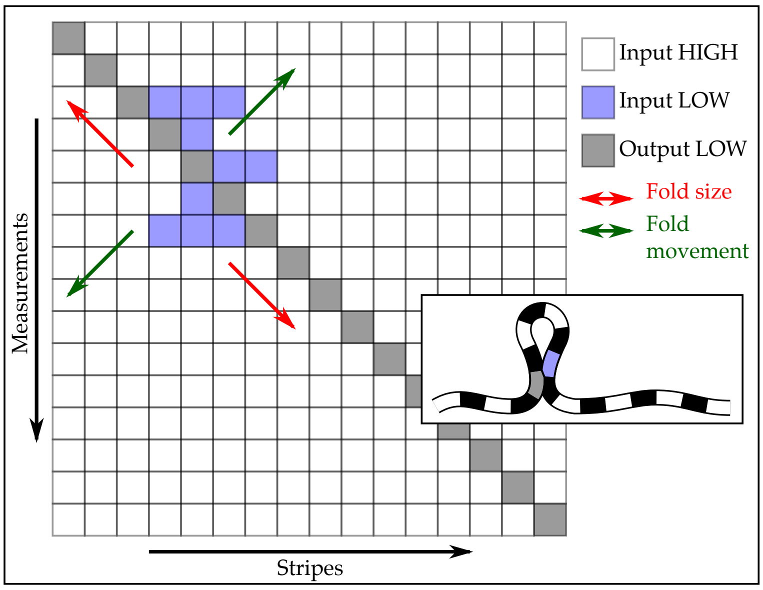

Figure 3.8: Matrix evaluation of a pinstripe signal. Each row of the array is one measurement of one stripe against each other, while the vertical columns represent the different stripes. Blue fields indicate a connection between this stripe and the measuring stripe, indicated grey in each row. The unsymmetric matrix indicates either noise or changes of the fold between a slow-going measurement.

The resulting matrix can then be used to detect the size of the fold on one diagonal by weighting each connection on one side from the grey diagonal by its position. The average of these weighted positions is the fold position. The minimal distance between resulting position and the diagonal represents half the size of the fold. Comparing the according positions at the different measurement determines the fold movement along the diagonale.

Detect distance by Weight and Add

The weighting of the single connections of a non-symmetric array, which indicates some errors due to measurements, allows some kind of correction: Each side of the diagonal may be evaluated independently, and the result could be compared. On the other hand, the relative high number of measurements per array in combination with the determination of the position already smoothes the data.

Smoothing by mean

Low-pass filter: Adding old values

To prevent errors due to lost connections (e.g., mechanical disconnections of a stripe or loose connections while moving the fold) or non-intentionally shortened stripes (e.g., movement of the garment), these values are additionally smoothed over time by adding weighted old and new values, forming a low-pass filter. If a movement is detected, an according command is send to the receiving iPod, and is repeated according to the size of the fold.

Safe, but slow and expensive

The low-pass filter prevents most unintentional commands, but increases the time between an intentional directional change by the user and a resulting corresponding command. The use of these weighted values needs floating point calculation, which is less suited for a micro-controller. Since an exact value is not necessary, the value could be changed into an integer by multiplying an according factor and rounding. By choosing this factor, an efficient weighting may be implemented with bit-shifting or using the integrated hardware multiplier of the MSP430.

To speed up the reaction time of pinstripe and nevertheless getting only intentional commands, the filter algorithm has to be adapted. For an efficient design, the measurement of the stripe signal was overworked, too. The resulting patterns in the measured data then lead to the overworked filter algorithm, described in the next part of this thesis.

3.3.1 Measurement method and pattern

New: diagonal row instead of full array

Instead of setting one stripe on ground and measure all other stripes against this one stripe, in this version all stripes are grounded and one stripe after another is pulled-up and measured against all others. This represents the diagonal (grey fields) of the pinstripe matrix from the original algorithm - while for the rest of the matrix no data exists anymore. Both movement of the fold parallel to the main diagonal and the fold size orthogonally to this diagonal are therefore projected on this line: The distance between the outer connected stripes determines fold size and a fold movement changes these connected stripes.

Fast and efficient, but less stable

The main advantage is a faster measurement: Only one measurement has to be done for each pin, and the measured value can be directly used as result. This needs only register manipulations for a single pin, while all other remain unchanged. A faster measurement itself is not useful, the fold movement determines an upper bound for a repeated of measurement: Otherwise no change, apart from noise could be detected. Instead, it allows an implementation with lower clock frequency and therefore better energy consumption. If the pin is pulled-down through an interconnection with another stripe, this pin is an outer limit of the fold, if no stripes before or after are shortened. Furthermore, counting all shortened stripes could be directly compared with a limit value to detect crumbling of the textile. The size of the fold could be computed as the difference between the two limiting stripes. The fold movement can be obtained by comparing the difference of the limiting stripes of two measurements after another.

As an additional benefit, debugging is simplified, since each row of measured data could be compared with the predecessor, getting a two dimensional array over time. The use of fewer measurements to determinate size and position of the fold in comparison to the original two dimensional array increases the risk of measuring unintentional disconnections. This should be taken into account for the filtering. For this purpose some of the most common patterns over time are discussed next.

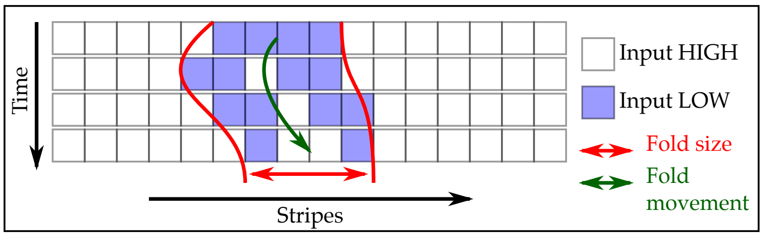

Figure 3.9: Pinstripe patterns—Fold movement

Fold movement

Each row in figure 3.9 represents one measurement of all (horizontal arranged) stripes. Again a blue field indicates, that the stripe on this position has a connection to another stripe. In the vertical direction from top to bottom we see changes over time: First, the fold moves and increases to the left side, then goes to the right side and becomes smaller (distance between the outermost blue fields). Since a minimal change of position and fold size could be easily caused, e.g., by loose connection or by change of the grip, a bigger position change than one over time should be necessary to trigger a command.

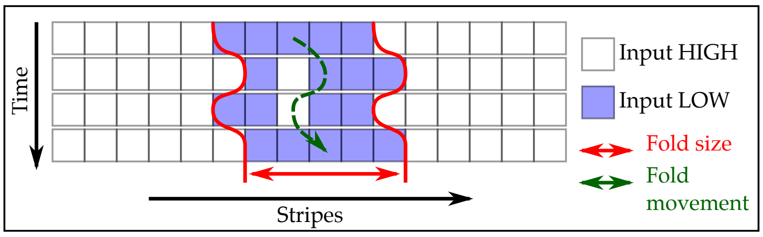

Figure 3.10: Pinstripe patterns—Unstable signal

Unstable signal

Loose connections and changes in finger pressure or unintentional moves of the fold may cause changes in the signal as shown in 3.10. Therefore, the signal has to be stable for a certain time until a command is generated: The comparison of a signal with the predecessor should indicate the same movement as the comparison of the predecessor with the pre-predecessor or at least no movement.

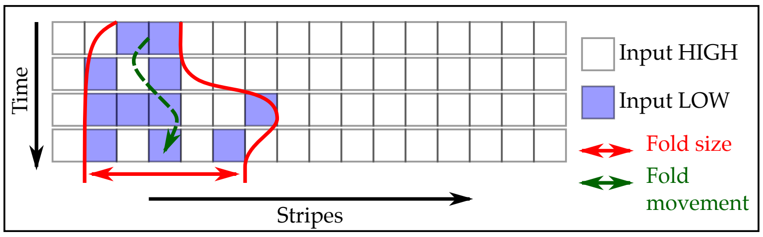

Figure 3.11: Pinstripe patterns—Changes in the fold size

Changes in the fold size

A big change in fold size as shown in figure 3.11 between the second and third row from the top may be caused either by loose connections, unintentional folds, or recatching. To prevent these commands due to this massive movement of the fold, a signal is generated only if the fold size changes within certain bounds between two measurements. A minimal change has to be allowed, since moving the fold by rubbing the textile between thumb and index finger results in a steady change in fold size. This kind of making a fold causes also a difference in usability between the two directions of moving the fold.

3.3.2 Filter algorithm

Frequency filter

The pinstripe behavior could be modeled as a line of buttons, which are pressed and released by the user while moving the fold. The time between each release or press depends on how fast the fold is moved. Additionally, there are only few measurements before a signal should be generated: The movement of the fold until a command is generated should be as small as possible for a fast feedback of the system. In this case a frequency filter can not be used to detect and filter the intentional signal, because of the lack of frequency—the ideal user input is a linear movement of the stripes to one side.

On the other hand, the user may also correct unintentional moves by changing the direction, and—like buttons—each activity may be connected with a bouncing behavior. Furthermore some buttons may be inactive (defect stripes), or unintentional released by the user, or connected by movement of the garment itself.

Design fundamentals

This results in the following thoughts:

1. The fold has to be moved over a minimal number of stripes to be count as valid movement, smaller movements may result because of bouncing behavior. Changes of very few stripes have to trigger a reaction for a short reaction time of the system. Therefore, no longtime analysis to determinate the speed of the user input for filtering analysis is possible. With a longer reaction time, more stripes will be crossed by the fold and therefore a safer signal may be generated.
2. To prevent wrong signals because of unintentional connections or disconnections of stripes, the size of the fold has to be stable. Since the size may change because of rolling the fold a minimal change in the size between measurements has to be allowed.

3. If a movement in one direction is detected, a corresponding command is only triggered if the next measurement determines a movement in the same direction or no movement at all. Comparing more measurements will result in longer reaction time, but again in safer signal detection.

4. To prevent wrong commands due to the unknown behavior while pinching the fold, a delay between the first measurements and the evaluation may be useful.

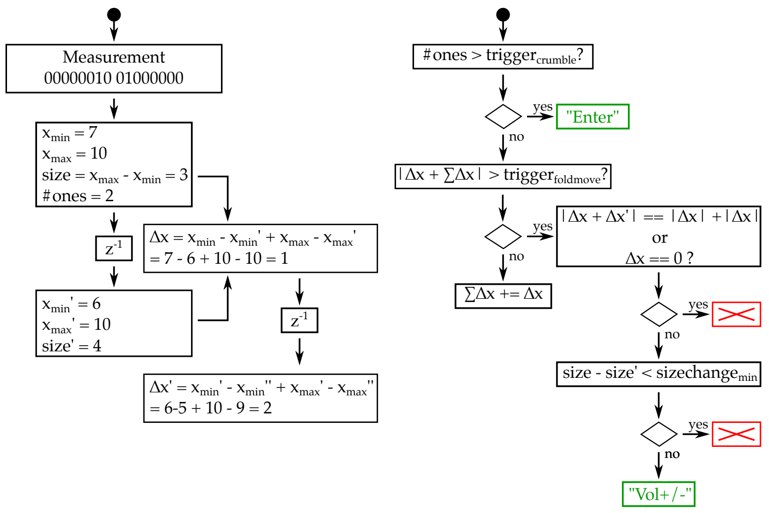
Measurement
00000010 01000000
$x_{min} = 7$
$x_{max} = 10$
$size = x_{max} - x_{min} = 3$
#ones = 2
z^{-1}
$\Delta x = x_{min} - x_{min}' + x_{max} - x_{max}'$
$= 7 - 6 + 10 - 10 = 1$
$x_{min}' = 6$
$x_{max}' = 10$
$size' = 4$
z^{-1}
$\Delta x' = x_{min}' - x_{min}'' + x_{max}' - x_{max}''$
$= 6\text{-}5 + 10 - 9 = 2$
$\#ones > trigger_{crumble}?$
yes
"Enter"
no
$|\Delta x + \sum\Delta x| > trigger_{foldmove}?$
yes
no
$\sum\Delta x \mathrel{+}= \Delta x$
$|\Delta x + \Delta x'| == |\Delta x| + |\Delta x|$
or
$\Delta x == 0$?
yes
no
$size - size' < sizechange_{min}$
yes
no
"Vol+/-"

Figure 3.12: System design: Arithmetic operations over time on the left and decision tree on the right side

This could be efficiently implemented with a microcontroller (figure 3.12, left side): Only comparisons between different integer values are needed. Both actual and previous values are stored in an array, where a boolean value, which is toggled in each measurement, determines the actual position. The result of each comparison at a time can be used as a reference for the next analogue comparison, preventing a doubled computation. In particular the leftmost and rightmost stripe, which are connected with other stripes are stored as actual position, and the also the total number of connected stripes. The difference between the outermost stripe positions gives the fold size, and changes of the position between measurements give the fold movement.

If connections between stripes are not detected, the microcontroller can sleep, since no fold is present. This also resets all values.

The right side of figure 3.12 represents the decision tree: If more than a certain number of stripes are connected, a crumbling of the textile is detected and therefore a corresponding command (here: "Enter") could be send. Otherwise, the actual movement of the fold, represented here by the Δx, together with the sum over all previous fold movements is compared with a trigger value. The trigger represents the minimal distance which should be covered by the fold movement before a command is triggered. If this trigger is reached, the system checks if the size of the fold stays within the allowed bounds and if the signal remains stable: The actual Δx should be at least zero (no movement) or should have the same prefix as the $\Delta x'$ before (movement in the same direction). If both requirements are fulfilled, the corresponding command, depending on the prefix, could be sent.

Control options

Once a command is generated, it could be used in different ways to accomplish various goals:

1. The number of commands per grab can be limited through a trigger value, which is only resetted if the fold is released. This is especially useful for crumpling the textile to generate a toggle command.
2. This command could be repeated according to the fold size: Here it makes sense to use a nonlinear function. Very small fold sizes could not be reasonably distinguished, and therefore should all trigger one single command, whereas after a certain size, the number of repeated commands can be increased. A simple and efficient function is shifting the fold size two bits to the right and adding one for each cycle.
3. The sum over the fold movements may be reseted, if only one command per certain movement of the fold is favored, like a skip forward command for an audio-player. Otherwise the command is repeated until either the fold moves back to the original position,

resetting the sum below the trigger value, or the fold is released. This allows to achieve high numbers of repeated commands, while releasing the fold allows a fast feedback and therefore a relative high precision is still possible.

3.3.3 Command generation

If the evaluation of the measurement determines a stable movement and size of a fold or a high number of shortened stripes, which indicates a crumpling of the garment, the corresponding command is send either to the receiving device and/or a computer for debugging.

For this purpose a serial sender is implemented, based upon the Software serial library by David A. Mellis, as found within the Arduino IDE. Basically—without receiving and only sending single bytes—it is just a timed change of the output level of a pin. These timed delay could be implemented by either counting a number of clock cycles or—less energy consuming—by a short timer-controlled sleeping function of the micro-controller. After such a delay, the next bit of a byte is sent by changing the pin level.

The bytes to send to control an iPod are derived from the iPodLibrary[7] from David Findlay, with additional information like according debug bytes and using a chain of commands instead one single command.

3.3.4 Energy saving

Clock frequency

A low clock frequency results in a low energy consumption. Therefore a slow frequency of only 1 MHz is used for the normal function, and is even lowered between measurements. Since the Atmega is used with an internal clock frequency of 8MHz or an external 16MHz crystal on the Arduino board, different clock frequency dividers are

[7] https://github.com/finsprings/arduinaap

used. For this purpose the internal base clock frequency is read out and the according clock divider is selected automatically.

Timer interrupt

Between measurements on one hand and as short time delays for serial communication on the other, the micro-controller should sleep, disabling most of its functionality apart from the internal oscillator. Each tick of the oscillator is counted until a certain number is reached, where a timer event is triggered, waking up the micro-controller.

The timing interval can be controlled by changing the corresponding trigger value, enabling the usage of the same timer for both types of time delays.

Pin-change interrupt

Whenever no fold is present, which results in no interconnection between stripes, no further measurement is necessary and the micro-controller can be suspended until the next interconnection happens. For this purpose, each second stripe is used as an input and pulled up, while the other remain grounded. If one stripe of each group is then connected with one of the other, a pin-change interrupt is triggered and wakes up the micro-controller.

In rare cases, a fold could connect in rare cases in connecting only members of each group with each other, letting the micro-controller sleep. Small movements of the fold resolve this case.

Chapter 4

Evaluation

At the end the reworked prototype and software has to be tested: Both in daily use of the prototype to detect mechanical design flaws and a user study to determine the usability of hardware prototype and software, especially the constraints for textiles and the software response on user input.

4.1 Daily Use

Mechanical failures

One major cause for hardware breakdown of the final prototype were unintentional pulling on the cable between pinstripe and iPod. The PCB within the iPod connector improves repairability, and the connector between cable and flexible PCB prevents defects on this side. The connection between rigid and flexible PCB was prone to defects due to extreme bending: Solder on the flexible copper laminate makes the circuit path breakable on this spot. Applying solder only between both PCBs and making a continuous transition between rigid and flexible PCB may further reduce the probability of a circuit break.

Loosing conductivity

Additionally, the system behaves less sensitive because the conductive surface of the pinstripe textile wears down while using.

4.2 User Study

4.2.1 Design

Structure and design

Each part of the study uses the same configuration on the screen, which is is shown in figure 4.1: On the top a button, a continuous slider, and an indexed slider is shown (and greyed out if not active, like the indexed slider in the picture). Below is a questionnaire for the according part of the study, which could be filled out at any time before continuing with the next part. Pinstripe did not work on top of other clothes, since these textile may be included in the fold, isolating the stripes from each other. Therefore an additional snap-band was used in the study to prevent this behavior, delivering a non-foldable background below the pinstripe prototype and above the clothes.

Compare textiles

The user study first lets the participant compare the different textiles (the pinstripe garment with only a transfer film applied, and with polyester, silk, cotton and fly screen on top of this film). For this purpose, each textile could be attached on the arm of the participant via a snap-band. To distinguish the different textiles for evaluation, apart from color-coding the textile itself, the used textile is indicated on screen.

Grade different functionality

After choosing the best suitable textile (and grading all of them), the participant continues the study with this textile. Gender, handedness, age and experience with wearable computing are queried next. Furthermore, the participant chooses how the direction of a fold movement should be translated in a movement of the slider. After this part, they evaluate this input mode—how fast the system reacts, if it delivers the right commands or freezes, if the possibility of error correction works well, and if this mode may be generally useful. The same questions are then asked again for the following, slightly different modes:

1. “Volume”: In the first part crumbling the textile switches the button on the top, and moving a fold to the left or right results in a continuous movement of the slider in the middle, until the textile is either

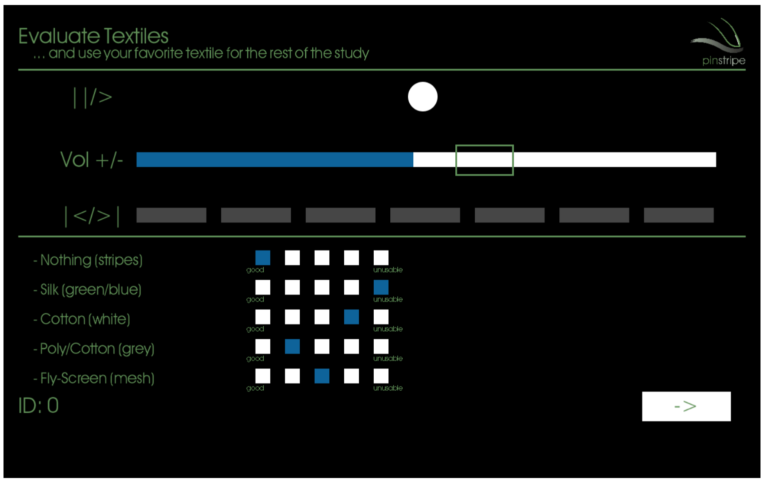

Figure 4.1: Screenshot of the user study GUI: In the upper half the active continuous slider and button and the inactive indexed slider. In the lower half the questionnaire.

released or the fold is moved back to the start position, which is like a continuous "Volume" control of a device.

2. "Skip": The indexed slider is used—"skipping" back and forth between each part—and the button. The button is used as before, but a movement of the fold in one direction triggers only one command per covered distance by fold movement, not repeating the command until released as before.

3. "Switch": Both slider are in use, "switching" between each of them via fold size. For example a big fold will change the volume, while small folds skip forward or backward.

4. "Select": This mode uses the indexed slider again, but with the button function (crumpling the textile) to confirm the selected input, as "selecting" an entry from a list.

All the time a green rectangle indicates a target. If it is reached, it will disappear and a new one will appear on another position. This is evaluated in the first three modes by releasing the textile, and for the last mode with the

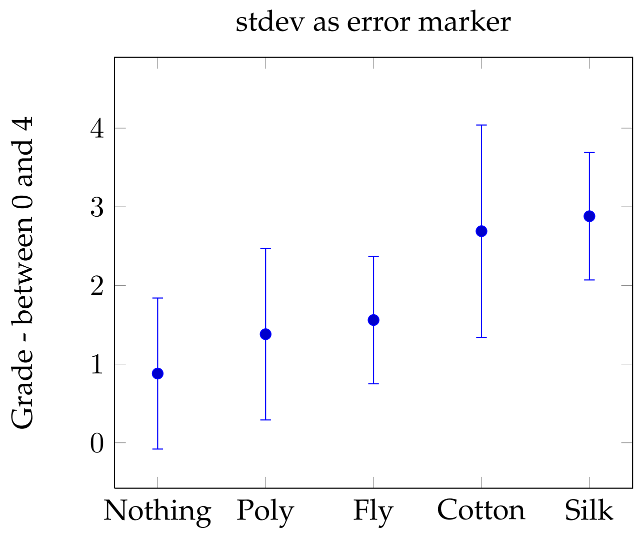

Figure 4.2: Evaluation of carrier textiles

crumpling command. Time until the target is reached, target and size, the raw data of each stripe connection as well as the processed commands are stored.

4.2.2 Results

Determing a good textile

As shown in figure 4.2 the best textile to use with pinstripe is no textile at all: The pinstripe textile with only an applied transfer film (Nothing) for t-shirt print could be well handed and is highly flexible. On the other hand, the polyester/cotton (Poly) textile is worse to grab, but with a lower flexibility a more precise handling may be possible, reducing the possibility of unwanted folds. Both disadvantages are mentioned in the study from some participants. At least this may explain why one part of the users prefer "Nothing", while other prefer polyester/cotton. The fly-screen is between these two most of the time: with its better grip and firmer garment it is also in this top flight—although it has a nearly significant worse grading to the version with only the transfer film.

Cotton and silk remain far behind—either too firm and/or with a worse grip, although one person rated the cotton version as best textile and used it in the rest of the study.

Evaluating the raw data

These results could be partly confirmed by the number of wrong commands per total number of commands until a target is reached. To determine this, the data from the user study was evaluated automatically: The data files of each user where scanned for the "…expected" sentence, indicating the begin of a new targeting cycle. Each command on the right side of the numbers—indicating the connection("1") or non-connection ("0") of stripes—is then counted, until either a change in the mode occurs or, as it happens in the example in table 4.1 a "Time" is printed out. Then time, the expected target (here: crumbling the textile—or toggle "TP"), and used textile (here: cotton, "C") and the number of the different commands is stored, as well as the actual part of the user study. The user input in the study is evaluated after a sleep command ("SL"), indicating that no fold is present—while this is not necessary for the crumbling, the sleep indicates a release of the textile and therefore the comparison with the target value for the slider movements. For example, in 4.1 a time for successful reaching a target follows the "SL", which indicates that this evaluation was made.

…	Toggle expected: true Actual value: falseC 0000000011100111000 0000000111111111100TP 0000000111111111100 … 0000000111111111100 0000000101100100100 0000000000000000000?? 0000000000000000000?? 0000000000000000000SL Time: 3864.0c	…

Table 4.1: Raw user data

The resulting data can then be used to either compare different parts of the user study or, for this part of the evaluation, the different textiles within the first part. These results are shown in figure 4.3: Only the position of cotton and fly-screen is flipped, but the differences between all textiles are not significant and both error rate and standard

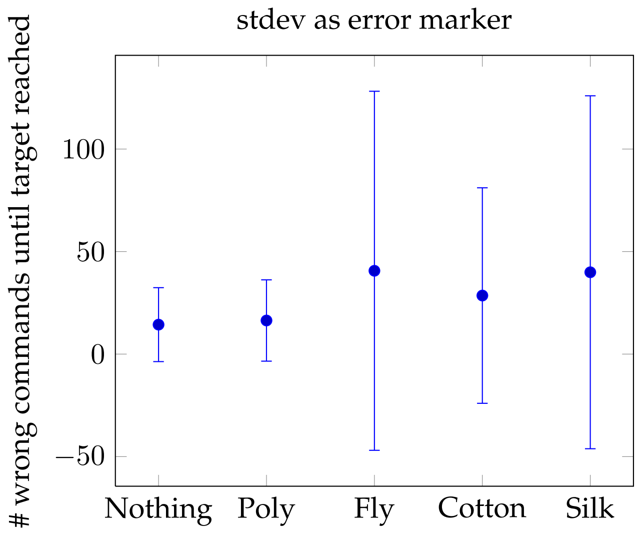

Figure 4.3: Data evaluation of carrier textiles

deviation are high. This high number of wrong inputs—contradicting the later described overall good grading of the usability—has several causes: First, there was no training phase, every user input from the beginning on is evaluated. This produces a high number of wrong commands, especially since some user were not able to handle pinstripe to their satisfaction. Second, the user may play around with the system and did not want to reach the target directly, especially here at the beginning. And a unusable textiles for one user causes extreme high error rates.

System response

The response time of the system is graded as good for all modes in the questionnaire. The overall time according to the raw user data between waking the micro-controller up the and first response command is about 486 ms (counting the number of lines of data, equal the number of measurements, after waking up until the first command is sent, times the time for one measurement).

The users graded the response time around 1 for all three modes—and similarly, the system gave rarely no reaction at all. But of course in this case also a wrong response counts as a response.

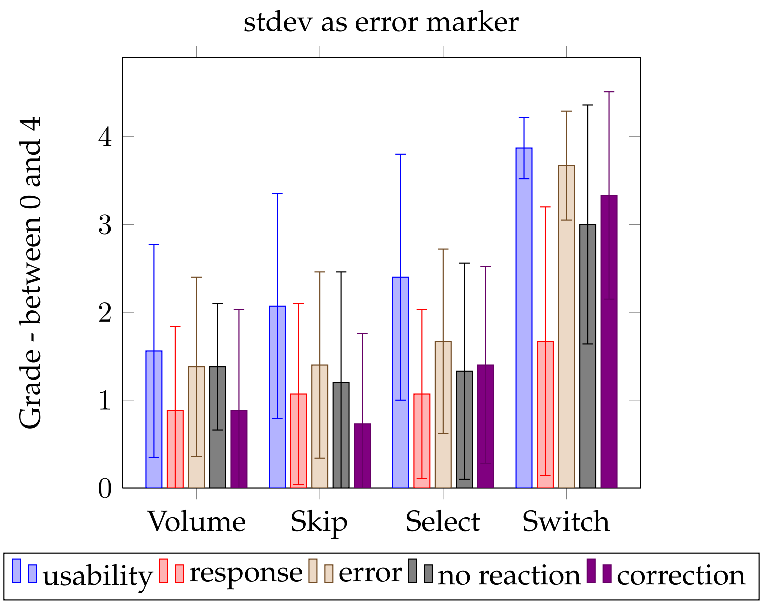

Figure 4.4: Evaluation of different modes

Evaluate Functionality

Most of the people did not have experience with wearable computing devices before, but apart from the mode where the participant has to distinguishing between the two slider with the fold size (“Switch”), most users could use it: confirming the slider position with crumpling the textile (“Select”) was slightly worse, since an additional grip per command was needed. Crumpling the textile may also result in short movement, which moves the position away from the target, reducing usability. Especially error correction was far worse than the error correction for the indexed slider (“Skip”).

The usability of the different modes is similar: “Volume” and “Skip” got the best grades—which version is better suited for the mobile version has to be determined in the future, since the difference between these two is not significant. Additionally, a mobile application with audio feedback has different constraints than this user study with visual feedback: A short test within the daily use indicates that the “Volume” function may change the volume too fast.

The “Select” function, maybe usable for list selection, has a slightly lower grade. Here, the major disadvantage is the

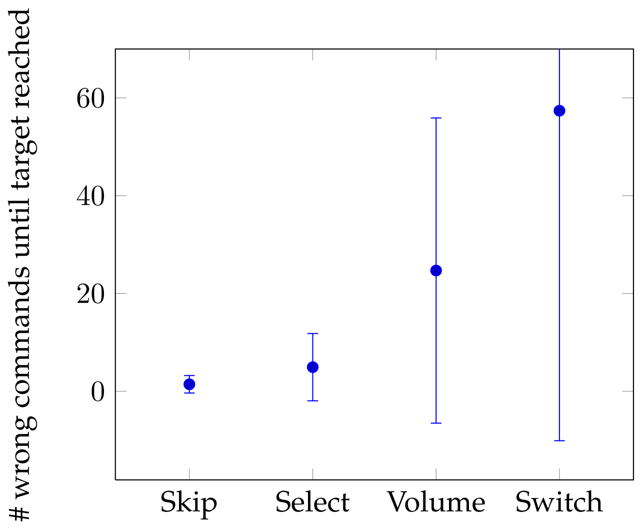

Figure 4.5: Data evaluation of different modes (stdev as error marker)

additional grip. For a mobile application a feedback for list selection has to be given either optically or acoustically, therefore reducing the prime feature of the eyes-free usability of pinstripe. Without surprise, the "Switch" via fold size is graded as useless.

According to this data, as shown in figure 4.4, a preference for "Volume" and "Skip", followed by "Select" seems to exist. But all these differences are not significant, as the relative large standard deviation also indicates. Only distinguishing different modes with the fold size ("Switch") is significantly worse than the other, making it unusable.

Evaluating the raw data, a wrong command happens often, especially for the "Switch" mode where only some users reach a target at all. The high numbers for the "Volume" were discussed above, and here in figure 4.5 the data from the best textile grading mode is used again for the "Volume" evaluation. "Skip" and "Select" achieve much lower values, indicating only small error rates in real-world usage. Nevertheless, apart from the mentioned issues with the "Volume" data, the ranking is the same as the grading of the users and therefore supporting it.

Other observations

The mapping of fold movement to slider movement should be user-adaptable, since no mapping is consistently preferred by all user. Most participants wrap the pinstripe prototype around the lower forearm, which was the easiest way to control it while sitting in front of the computer.

Chapter 5

Summary and future work

5.1 Summary and contributions

Wearable devices like smartphones and mp3-players are used today effortless while doing other things. Therefore, input devices, which do not need much attention, are necessary. A good example is the remote control on the headphone cable: It mirrors simply controls like volume changes of the device on the other end of the cable, but it is still more convenient to grab the cable, find the remote and press their buttons than using the same buttons on the device itself.

But locating the remote is still not effortless. A eyes-free and therefore more effortless interaction is the goal of the pinstripe system, which allows the users to control the device just by making a fold in their clothes. This causes short circuits between an array of conductive stripes on the back of the cloth, resulting in a corresponding pattern by the fold movement.

While wearable prototypes of this system already exist, they could be further improved by using smaller micro-controller (either MSP430G2553 or Atmega 168) and less components. Furthermore, the filter algorithm had to be adapted for the system, since the prototype before did not work in an optimal fashion—the system reacts relatively slow on user input. On the other hand, a low energy consumption is necessary for a mobile device.

System redesign

The evaluation software, including the measurement of the stripes, was reworked: An inverted measurement of the stripes—measure one stripe as an input against all other outputs instead of the other way around has the benefit of a more efficient implementation and additionally allows an easier debugging (a single line of zeros and ones, representing each stripe and their connections, instead of the two dimensional array before). It also eliminates the need of floating point calculation and uses time shifted comparisons instead of a frequency filter algorithm.

This results in faster reaction times and needs only a slow clock frequency.

With a slower reaction time, the error rate of misinterpreted patterns could be further reduced, but since errors can be corrected easily and quickly, the faster reaction time may be better. For lower power consumption, pin-change interrupts are used to detect folds. The micro-controller can sleep otherwise. Timer interrupts are used as delays with an otherwise disabled controller.

This prototype was tested both in a user study and in daily use.

Daily Use: Abrasion of the pinstripe fabric

The main remaining issue is the decrease of conductivity of the stripes due to wear and tear. Other mechanical failure causes like getting caught with the cable, causing disconnections within the iPod connector, could be reduced until the connector becomes the break point, which can be plugged in again. Another failure cause is the solder connection between rigid and flexible PCB. This failure only happens after extreme bending, and could be reduced by making a continuous mechanical transition between the two materials..

The software on the other hand works stable, although the mapping between (vertical) volume control and horizontal fold movement is not optimal.

A user study with different textiles indicates that the clothes, which could be used with pinstripe, should have an easy tangible surface: The plastic surface of a pinstripe textile coated with a transfer film for t-shirt prints (or a silicon surface, not tested, but proposed by a participant) is well graded by the users, while a rough surfaces like the fly-screen is lower graded. On the other hand, the textile should not be too flexible, allowing a precise control without unintentional folds: A polyester/cotton textile also gets good grades, although it has a worse tangible surface.

User Study: Tangible textiles, no switches between modes

Apart from one function, where different functions were controlled depending of the fold size, the overall usability was good: The response time was short, and an easy error correction and low error rate allow a usage without training.

5.2 Future work

The main issue may be the mentioned instability of the conductivity of the stripes over time. Since bending and a mechanical connection and movement between the stripes is necessary for the function, this issue could not be easily resolved, but may be further reduced by using different conductive yarns for the pinstripe fabric, e.g., made of stainless steel.

More robust pinstripe fabric

With a more robust fabric, the system could be tested in a longer user study with the intended use as an input or a wearable device. Here it may be interesting to select between the two possible mappings of fold movement to signal generation. Furthermore, a selection between sending just a certain number of commands per fold movement, and repeating these commands until either the fold is released or moved back to the start position may be interesting. Both modes where popular in the user survey and a certain mapping, corresponding for example to handedness, could not be determined.

Daily-use user study

For this purpose, a shield with two switches may be added to the prototype to enable to switch between the two modes while using pinstripe.

If the decrease of conductivity can not be reduced, another method of measurement instead the resistive one may be interesting.

Time Domain Reflectometry (TDR)

One—not practicable nowadays due to size and costs—approach for a similar device could be the use of time domain reflectometry: Wimmer and Baudisch [2011] propose a system, where the signal propagation delay is used to detect for example (capacitive) interconnections between two wires. Furthermore, both stretchable and even sketched forms where used. Transferred to the pinstripe idea, this could be used to either simplify the fabric layout (rotated by 90°, and with only two wires instead of the multiple ones—which could also be scaled better) or adding a second dimension—but again, nowadays a corresponding integrated time-to-digital converter for this scale (size, price, and resolution) does not exists.

Capacitive measurement

Another possible change is to switch to capacitive measurements, where only gestures above are detected. In this case, the distance to other parts of the body—like the arm—disturbs the measurement. This may be reduced with a double layer textile—a bottom, grounded layer is used as fixed potential, while the second layer measures the gestures on the other side, which would be similar to the GesturePad design proposed by Rekimoto [2001]. Since the used MSP430G2553 has the capability for capacitive measurements[1] for all 24 GPIO pins, it is easy to adapt the hardware prototype: The only necessary part is adding a second, conductive layer between pinstripe fabric and human body, which is connected to ground. In this case the abrasion will be reduced, but the short distance between the two layer will make a gesture recognition problematic. A reprogramming of the launchpad version shows that this measurement method generally works, but with flexible textiles, the required dimensions and distances,

[1]http://processors.wiki.ti.com/index.php/MSP430_Low_Cost_PinOsc_Capacitive_Touch_Overview

disturbances outmatch the signal. One modification may be a hand-sized grid of nine electrodes on the upper arm. Using the hand instead of just the finger enlarges the electrode and results in bigger changes in capacity. This array may be enough to detect simple patterns like moving the hand up or down above the sensor for volume control.

Snap-band as input device

Finally a snap band instead of the textile as a carrier can be used as an input device or even as a standalone mp3-player. The metal base could be either work as defined back layer for a capacitive measurement of gestures above, or movements of a top layer relative to the bottom layer could be detected with switches, reed contacts, or even phototransistor. The corresponding input devices, micro-controller and battery could be mounted on small, partly flexible PCBs on top of snap-band.

Even without exactly locating these two capacitive devices, the hand will be at some point above the sensor areal by moving along the opposite arm, and triggers a corresponding command. Power consumption and the stability of the measurement results as well as duration of one measurement cycle are the main challenges for this method. A further improvement of the pinstripe fabric with a different conductive yarn might be easier and is more promising.

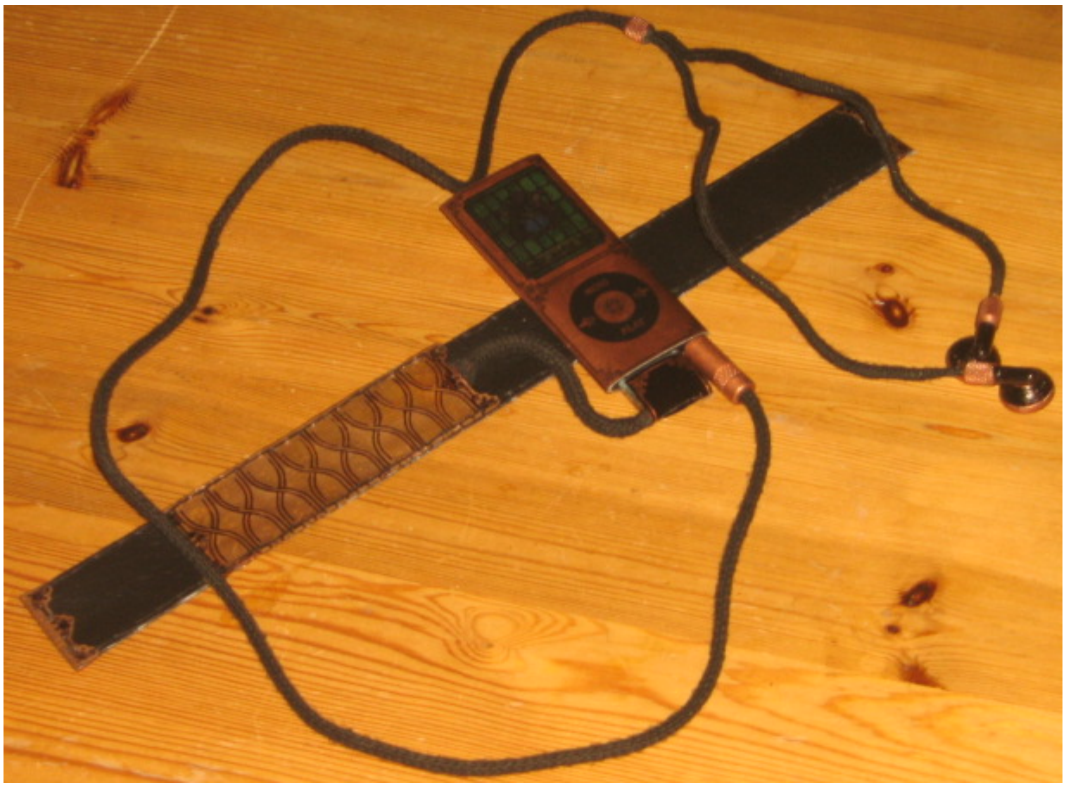

Figure 5.1: Future work: Snapband control

Appendix A

Micro-controller programming

This section summarizes the micro-controller code used in this thesis for both Atmega and MSP430 with short description and code examples.

A.1 Micro-controller selection

First, the overall selection, which micro-controller is used is done by choosing the according header file with the used pinstripe layout. In the header file, the same defines are established for each controller. Similar, both Arduino and Energia IDE uses defines to establish different register for each subtype of micro-controller. To distinguish these types, for example to adapt to different numbers of channels, a simple scan for the corresponding header file can be done: If for example __AVR_ATmega1280__ is defined, the according Arduino board is selected in the Board menu—and if this is not the right controller, an upload will not work and therefore no harm will be done.

Atmega	MSP430
`#if defined...`	`#if defined...`
`...(__AVR_ATmega1280__);`	`...(__MSP430G2553__);`
`...`	`...`
`#endif;`	#endif;

Table A.1: Determine micro-controller

A.2 Pin manipulation

Since we use the Arduino/Energia IDE, pin manipulation could be done with the shortcuts of the IDE like "pinMode" etc. This is inefficient—especially if a whole channel has to be switched, but also due to safety time delays and –requests. A direct register manipulation is better. A list of these bitwise register changes is enlisted in table A.2. Important is the additional possibility of an internal pull-down resistor at the MSP430, which allows greater flexibility—which we did not use—but on the cost of an additional command per pin change during measurements. Additionally, to these bitwise manipulations all bits of a channel could be simultaneously manipulated by setting the whole register, which is useful for initialization.

In the example, "pinNr" stands for a certain output pin of the Arduino or Launchpad board, while "pin" is a certain pin of each channel. These channels are enumerated alphabetical for the Atmega (B,C,D) or numerical for the MSP430(1,2,3), replacing the x in either DDRx, PORTx and PINx for the Atmega or PxDIR, PxOUT, PxREN and PxIN for the MSP430.

To achieve a similar structure for both micro-controller, an identical structure is defined for the pin layout:

```
typedef struct pinDefinition {
   volatile uint8_t port;
   volatile uint8_t ddr;
   const volatile uint8_t pin;
   volatile uint8_t ren;
   uint8_t pinNr;
} pinDefinition;
```

Atmega	MSP430
`pinMode(pinNr, OUTPUT);`	
`DDRx &= ˜(1<<pin);`	`PxDIR &= ˜(1<<pin);`
`digitalWrite(pinNr, LOW);`	
`PORTx &= ˜(1<<pin);`	`PxOUT &= ˜(1<<pin);`
`digitalWrite(pinNr, HIGH);`	
`PORTx |= (1<<pin);`	`PxOUT |= (1<<pin);`
`digitalWrite(pinNr, INPUT);`	
`DDRx |= (1<<pin);`	`PxDIR |= (1<<pin);`
`PORTx &= ˜(1<<pin);`	`PxREN &= ˜(1<<pin);`
`digitalWrite(pinNr, INPUT_PULLUP);`	
`DDRx |= (1<<pin);`	`PxDIR |= (1<<pin);`
`PORTx |= (1<<pin);`	`PxREN |= (1<<pin);`
	`PxREN |= (1<<pin);`
`digitalWrite(pinNr, INPUT_PULLDOWN);*`	
	`PxDIR |= (1<<pin);`
	`PxOUT &= ˜(1<<pin);`
	`PxREN |= (1<<pin);`
`digitalRead(pinNr)`	
`((1<<pin) & (PINx))`	`((1<<pin) & (PxIN))`

Table A.2: Pin manipulation
* do not exist for the Atmega

With this definition e.g. PxDIR in the pin manipulation is than replaced with *pinDefinition.ddr. To simplify the layout definition, a define for each micro-controller reduces the need of definitions to fill the structure to just two variables: Px for the MSP430 or x for the Atmega and the pinNr:

Atmega	MSP430
`#define EXP(channel)`	`#define EXP(channel)`
`&PORT ## channel,`	`&channel ## OUT,`
`&DDR ## channel,`	`&channel ## DIR,`
`&PIN ## channel,`	`&channel ## IN,`
`&PIN ## channel`	`&channel ## REN`

Table A.3: Set pin register

This code is basically the same as used in the original pinstripe prototype, on which this thesis is based upon.

The Atmega has the capability to use the Reset-pin as an general purpose pin: An internal pull-up resistor be programmed with the corresponding channel C, pinNr.6 as input and high. In this case an external pull-up resistor is not necessary.

A.3 Clock frequency

The base clock frequency of the Atmega can be divided by 1 (0x00), 2(0x01), 4(0x02), 8(0x03) and 16(0x04), while the MSP has four calibrated clock frequencies (…1MHz, …4MHz, …12MHz and …16MHz), other frequencies are possible with according DCOCTL and BCSCTL values. These values are written to the corresponding register, with the little difference, that the Atmega has to enable a timer change first with `CLKPR=(1<<CLKPCE);`—the change of the CLKPR register has to happen within few cycles after this command. Additionally, the internal base clock frequency of the Atmega differs from the external clocked Duemilanove—resulting in dissimilar dividers. To use the correct divider the base clock frequency can be determined with `F_CPU == 16000000UL`.

Atmega	MSP430
Enable frequency change:	
`CLKPR=(1<<CLKPCE);`	
Set new clock frequency:	
`CLKPR=0x03;`	`DCOCTL=CALDCO_1MHZ;`
	`BCSCTL1=CALBC1_1MHZ;`

Table A.4: Code example: Select clock frequency

A.4 Sleeping and timer interrupt

The basic `delay(duration)` of the Arduino IDE could be simulated as

```
for (int u = 0; u < delay; u++) {
  __asm__ __volatile__("nop");
}
```

This is not energy efficient (but accurate and stable), since the delay is programmed by "doing nothing" during a given time. A better approach uses timer—with a clock frequency as low as possible again—to count the clock cycles, while the rest of the micro-controller does not do anything—using a timer interrupt (see also AVR and Arduino timer interrupts[1]) in combination with a sleepmode (as described in AVR and Arduino sleep mode basics[2] , which covers the pin-change interrupts discussed next.).

Once again, the basic schema is the same for both controller: First select a timer (here, the Atmega uses the unchanged main clock frequency, while the MSP430 can use an additional low frequency oscillator (LFO)), and the type of counting for each clock cycle. Additionally, the trigger value, which should be reached by counting, has to be defined. After this initialization and enabling interrupts, the timer can start and the controller can go to sleep. The `LPM0;`–command both selects type ("0") of sleep and enters sleep mode in one step. To enable sleep modes for the Atmega, `<avr/sleep.h>` has to be included before.

The different sleep modes allow parts of the controller to remain active. In this case a low sleep mode with slightly worse power savings than the deeper ones is used, since the main clock has to remain active. Other special functions of the micro-controller are not used at all.

[1] http://www.engblaze.com/microcontroller-tutorial-avr-and-arduino-timer-interrupts/

[2] http://www.engblaze.com/hush-little-microprocessor-avr-and-arduino-sleep-mode-basics/

Atmega	MSP430
Setup:	
Use low frequency oscillator:	
	`BCSCTL3|=LFXT1S_2;`
Count up function of timer:	
`TCCR0A=0x02;`	`TACTL=TASSEL_1;`
Set trigger value for counter:	
`OCR0A=kWait;`	`TACCR0=kWait;`
Enable timer:	
`TIMSK0=(1<<OCIE0A);`	`TACCTL0|=CCIE;`
Start counting:	
`TCCR0B=0x01;`	`TACTL|=MC_2;`
Enable sleep mode:	
`set_sleep_mode(...`	
`...SLEEP_MODE_IDLE);`	
`sleep_mode();`	`LPM0;`
Exit sleep mode:	
`sleep_disable();`	
Disable timer:	
`TCCR0B=0x00;`	
Clear counter:	
`TCNT0=0x00;`	
Interrupt service routine:	
`ISR...`	`#pragma vector=...`
`(TIMER0_COMPA_vect) {`	`...TIMER0_A0_VECTOR`
	`__interrupt void...`
	`...Timer_A(void){`
Stop counting:	
	`TACTL&=~MC_2;`
Exit sleep mode:	
	`LPM0_EXIT;`
Disable timer:	
	`TACCTL0&=~CCIE;`
Clear counter:	
	`TACLR;`
`}`	`}`

Table A.5: Code example: Timer interrupt

When the counter reaches the trigger value, an interrupt is released and the interrupt routine is executed and the micro-controller wakes up. The timer can be stopped and disabled next and the counter is cleared for the next round.

Interestingly, this clean-up actions works well on different places: For the Atmega, most of it happens after jumping back to the main program, while the MSP430 works best with nearly everything within the interrupt routine.

For the Atmega, `#include <avr/sleep.h>` for the use of sleep modes at the beginning.

A.5 Sleeping and pin-change interrupt

For phases, where no fold—which means no connection between stripes—is detected, another kind of interrupt is used to wake up the controller: The pin-change interrupt which detects if the input level on a pin is changed causes an corresponding interrupt event. This allows all timer to sleep, further decreasing the power consumption of the controller.

While all pins of the first three channel of the Atmega are capable of pin-change interrupts (see also How to Enable Interrupts on ANY pin[3]), only the first two channel of the MSP430 feature this capability. This could be compensated by a corresponding layout: In this mode, each second stripe is grounded as a reference signal, which does not need to be pin-change interrupt capable. The others are pulled-up inputs and can therefore be grounded by the others if a fold is present, triggering an interrupt.

The structure is again similar between the two types of controller: Choose a type of edge detection (rising, falling, ...) and enable these type of interrupt for certain pins—in this case, all of them (grounded pins will not change, but do not bother either). While enabling and selecting can be done in one step for the MSP430 with the PxIES register for each channel, for the Atmega an additional register has to be changed: A general register, where the type of edge detection for all pins is defined (MCUCR).

[3]http://www.me.ucsb.edu/~me170c/Code/How_to_Enable_Interrupts_on_ANY_pin.pdf

After this initialization, the interrupts can be turned on and the sleep mode can be entered—in this case the deepest one, turning nearly everything besides the pin-change interrupt off.

After one of the pulled-up inputs is grounded by a neighboring stripe while folding the textile, the corresponding interrupt is triggered, waking the controller and setting a corresponding flag. Thereafter, the sleep mode is cancelled, the interrupt disabled and the flag cleared, thus returning to the initial state.

Atmega	MSP430
Setup:	
Set interrupt for all pins of a channel:	
`PCMSKx=0xFF;`	
Detect edge:	

`MCUCR=(1<<ISC01)...`
`...|(1<<ISC01);`
	`PxIES=0xFF;`

Enable pin-change interrupt:	
`PCICR=0x07;`	`P1IE=0xFF;`
Enable sleep mode:	

`set_sleep_mode(...`
`...SLEEP_MODE_PWR_DOWN);`
`sleep_mode();`
	`LPM4;`

Exit sleep mode:	
`sleep_disable();`	
Disable pin-change interrupt:	
`PCICR=0x00;`	`PxIE=0x00;`
Clear interrupt flag:	
`PCIFR=0x00;`	`PxIFG=0x00;`
Interrupt service routine:	

`ISR(PCINTx_vect) {}`	`#pragma vector=...`
	`...PORTx_VECTOR`
	`__interrupt...`
	`...void Port_x(void){`

Exit sleep mode:	
	`LPM4_EXIT;`
Clear interrupt flag:	

	`P2IFG=0x00;`
	`}`

Table A.6: Code example: Pin change interrupt

A.6 Capacitive measurement

The MSP430G2553 has the ability to measure capacitance on all general purpose pins. It uses the integrated comparator to charge and discharge the according pin until a certain voltage level is reached. These charting cycles are counted for a certain number of time, and are inversely proportional to the capacitance. A general overview about measurement methods and design guidelines could be found at Capacitive Touch Sense Technology[4] .

The program code follows the structure of the CapTouch[5] library by Robert Wessels: PIN OSC is enabled for the measuring pin. Each rising and falling edge triggers a interrupt event and this events are counted for a certain time with the timer. The watchdog timer is further reduced by WDTHOLD to measure over a longer time.

[4]http://www.silabs.com/Support Documents/Software/Capacitive Touch Sense_Technology_SPKR.pdf

[5]https://gist.github.com/2941071

MSP430

Initializiation:

Timer & Watchdog:

```
TA0CTL = TASSEL_3 + MC_2;
WDTCTL = WDTPW + WDTHOLD;
WDTCTL = (WDTPW + WDTTMSEL + 0x00 + 0x03);
IE1 |= WDTIE;
```

Measurement:

Enable capacitive measurement:

```
PxSEL2 |= (1 << pin);
PxSEL &= ~(1 << pin);
```

Enable Counting of timer:

```
TA0CCTL1 = CM_3 + CCIS_2 + CAP;
TA0CTL |= TACLR;
```

Sleep until edge detected, count:

```
__bis_status_register(LPM0_bits + GIE);
TA0CCTL1 ^= CCIS0;
```

Disable capacitive measurement:

```
PxSEL |= (1 << pin);
PxSEL2 &= ~(1 << pin);
```

Cycle number

```
capacitive = 1/TA0CCR1;
```

Table A.7: Code example: Capacitive measurement

Appendix B

Code and Data

Schematics[a]

[a]http://hci.rwth-aachen.de/public/Research Projects/Pinstripe/Documentation/Hardware/Schematics/Jan Thar

Pinstripe source code[a]

[a]http://hci.rwth-aachen.de/public/Research Projects/Pinstripe/Documentation/Software/Pinstripe Jan Thar

User study[a]

[a]http://hci.rwth-aachen.de/public/Research Projects/Pinstripe/Data/User study /Jan Thar

Thesis[a]

[a]http://hci.rwth-aachen.de/public/Research Projects/Pinstripe/Publications/Thesis Jan Thar

Bibliography

Leah Buechley. A construction kit for electronic textiles. In *Proceedings of the 10th IEEE International Symposium on Wearable Computers*, ISWC '06, 2006.

Jingyuan Cheng, D. Bannach, and P. Lukowicz. On body capacitive sensing for a simple touchless user interface. In *Medical Devices and Biosensors, 2008. ISSS-MDBS 2008. 5th International Summer School and Symposium on*, pages 113 –116, 2008.

Jonny Farringdon, Andrewj. Moore, Nancy Tilbury, James Church, and Pieter D. Biemond. Wearable sensor badge & sensor jacket for context awareness, 1999.

Francine Gemperle, Chris Kasabach, John Stivoric, Malcolm Bauer, and Richard Martin. Design for wearability. In *Proceedings of the IEEE 2nd International Symposium on Wearable Computers*, ISWC '98, pages 116–122, 1998.

Paul Holleis, Albrecht Schmidt, Susanna Paasovaara, Arto Puikkonen, and Jonna Häkkilä. Evaluating capacitive touch input on clothes. In *Mobile HCI*, ACM International Conference Proceeding Series, pages 81–90, 2008.

Stefan Ivanov. Textipad: Implementation and evaluation of a wearable textile touchpad. mastersthesis, RWTH Aachen University, September 2012.

Thorsten Karrer, Moritz Wittenhagen, Leonhard Lichtschlag, Florian Heller, and Jan Borchers. Pinstripe: eyes-free continuous input on interactive clothing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1313–1322, 2011.

Nicholas Komor, Scott M. Gilliland, James Clawson, Manish Bhardwaj, Mayank Garg, Clint Zeagler, and Thad Starner. Is it gropable? - assessing the impact of mobility on textile interfaces. In *Proceedings of the 13th IEEE International Symposium on Wearable Computers*, ISWC '09, pages 71–74, 2009.

Seulki Lee, Binhee Kim, Taehwan Roh, Sunjoo Hong, and Hoi-Jun Yoo. Arm-band type textile-mp3 player with multi-layer planar fashionable circuit board (p-fcb) techniques. In *Proceedings of the 14th IEEE International Symposium on Wearable Computers*, ISWC '10, pages 1–7, 2010.

Torsten Linz, Christine Kallmayer, Rolf Aschenbrenner, and Herbert Reichl. Embroidering electrical interconnects with conductive yarn for the integration of flexible electronic modules into fabric. In *Proceedings of the 9th IEEE International Symposium on Wearable Computers*, ISWC '05, pages 86–91, 2005.

Diana Marculescu, Radu Marculescu, Nicholas H. Zamora, Phillip Stanley-Marbell, Pradeep K. Khosla, Sungmee Park, Sundaresan Jayaraman, Stefan Jung, Christl Lauterbach, Werner Weber, Tünde Kirstein, Didier Cottet, Janusz Grzyb, Gerhard Tröster, Mark Jones, and Zahi Nakad. Electronic textiles: A platform for pervasive computing. *Proceedings of the IEEE*, 91(12), 2003.

E. R. Post, M. Orth, P. R. Russo, and N. Gershenfeld. E-broidery: design and fabrication of textile-based computing. *IBM Systems Journal*, 39(3-4):840–860, 2000.

J. Rantanen, N. Alfthan, J. Impiö, T. Karinsalo, M. Malmivaara, R. Matala, M. Mäkinen, A. Reho, P. Talvenmaa, M. Tasanen, and J. Vanhala. Smart clothing for the arctic environment. In *Proceedings of the 4th IEEE International Symposium on Wearable Computers*, ISWC '00, pages 15–23, 2000.

Jun Rekimoto. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In *Proceedings of the 5th IEEE International Symposium on Wearable Computers*, ISWC '01, pages 21–27, 2001.

Julia Schwarz, Chris Harrison, Scott Hudson, and Jennifer Mankoff. Cord input: An intuitive, high-accuracy, multi-degree-of-freedom input method for mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1657–1660, 2010.

Aaron Toney, Barrie Mulley, Bruce H. Thomas, and Wayne Piekarski. Social weight: Designing to minimise the social consequences arising from technology use by the mobile professional. *Personal and Ubiquitous Computing (2003)*, pages 309–320, 2003.

Rene Vieroth, Thomas Loher, Manuel Seckel, Christian Dils, Christine Kallmayer, Andreas Ostmann, and Herbert Reichl. Stretchable circuit board technology and application. *2009 International Symposium on Wearable Computers*, 0:33–36, 2009.

M. Weiser. The computer for the 21st century. *Scientific American*, 272(3):78–89, 1995.

Raphael Wimmer and Patrick Baudisch. Modular and deformable touch-sensitive surfaces based on time domain reflectometry. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 517–526, 2011.

Typeset February 11, 2013