The Smart-Glove Maker: An Embroidery-based Pipeline for Fabricating Smart-Gloves

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

by
David García Olivares

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Ulrik Schroeder

Registration date: 16.05.2018
Submission date: 05.11.2018

Eidesstattliche Versicherung

Garcia Olivares, David	362118
Name, Vorname	Matrikelnummer

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

The Smart-Glove Maker: An Embroidery-Based Pipeline for Fabricating Smart-Gloves

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

Aachen, 05.11.2018	
Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

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

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

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

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

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, 05.11.2018	
Ort, Datum	Unterschrift

Contents

List of Figures

List of Tables

Abstract

Motivated by the use of smart-gloves as a tool for the interaction and manipulation of computerized environments. And by looking for methods and techniques to facilitate the creation of these devices. We introduce the ”Smart-Glove Maker”, an end-to-end system that allows people to make their own smart gloves through non-technical steps.

As an initial phase of our research, we explored the current proposals for soft textile sensors, the materials required and the techniques for their fabrication. Then, as a starting point, we developed the first smart-glove prototypes incorporating textile sensors into them. Typically, to build a smart-glove, many skills and many steps are required. From designing the glove pattern, designing the connections, designing the circuit, sewing and isolating the traces, building the glove, until finally, programming the glove to be functional.

Our pipeline reduces the number of these steps into four: the design of the glove, the automatic embroidering of the circuit, the construction of the glove, and its programming. Based on a catalog, people select the button arrangement according to their needs. Next, our system implements a routing algorithm for the automatic creation of the glove patterns. These glove patterns describe the spacing and stitch density between each trace of the circuit. Using computerized embroidery techniques patterns are converted into embroidered circuits. Then, the system interface instructs step by step through visual guides for the construction of the glove. Ultimately, people can program the functionality of the gloves through gestures without the need for additional programming. At the end of our pipeline, people have a smart-glove for gesture detection. Our smart-glove design is made-to-measure, lightweight, flexible and resistant.

Acknowledgements

Foremost, I would like to thank my thesis advisor M.Sc. Nur Al-huda Hamdan for her constant support, and for steering me in the right the direction. I much appreciated your constructive feedback and the time spent.

Second, I would like to thank the persons who participated in the study for this research, your valuable feedback was an invaluable help to improve this work.

Special thanks to Prof. Dr. Jan Borchers for providing me the opportunity to be part of the i10 chair.

Finally, I must express my very profound gratitude to my parents and my girlfriend for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Bless you.

Chapter 1

Introduction

Gloves have been garments used since ancient times. The use of these gloves addresses various purposes; from protecting the hands from the weather, hits or harmful substances, to fashion purposes. Nowadays, gloves can also serve as tools. Our hands are perfect for solving many types of tasks, and we instinctively use our hands to control and interact with other everyday objects. Thanks to the advance of wearable technology, it is possible to propose new interactions based on smart-clothing.

As a topic of interest, in this thesis work, we focused on the fabrication of smart-gloves. Smart gloves are garments that include electronic components. These devices have the potential of enabling new channels for interacting with computerized environments and objects, by extending our biological capabilities. For example, there are proposals to deal with simple activities like answering or ending a phone call by Huber et al. [2013]. Even more complex activities such as the manipulation of drones through the movement of the hand by Sandru et al. [2016], to enable communication between disabled and non-disabled people by converting gestures into text Gollner et al. [2012].

Dedicated smart-gloves help us solving everyday tasks.

Currently, there are broad varieties of smart-gloves on the market focused on solving various tasks. However, their costs are not always accessible. There is also the possibility of fabricating them on your own. Huber et al. [2013]

Current limitations for creating smart-gloves

showed in their evaluation for their smart-glove prototypes, that there is a keen interest in the users to enable the possibility of creating customized smart-gloves, in such a way, they are able to choose the position, shape, and aesthetics of the sensors. Thanks to the boom of personal fabrication and the DIY community, it is possible to find online guides, tutorials, and open source code to assemble your own smart-gloves. Nonetheless, this alternative is not always as simple as we would like to.

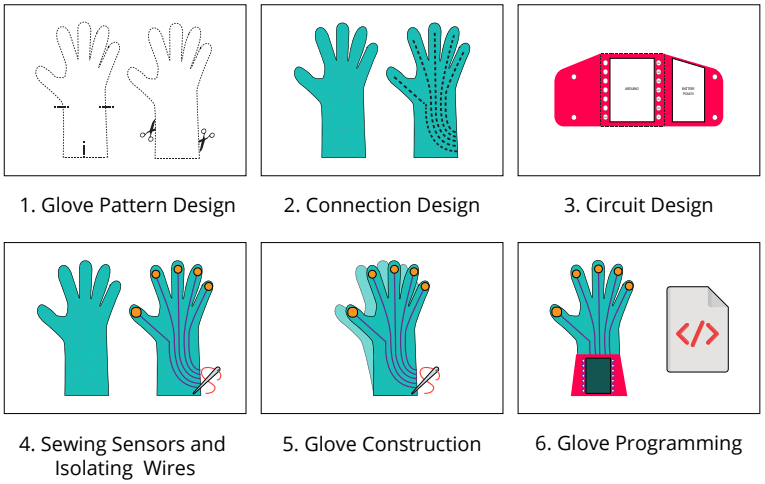


Figure 1.1: Typical steps for fabricating a smart-glove

Fabricating a glove involves several steps that require technical knowledge and skills.

Creating a smart-gloves involves many steps, see figure 1.1: 1. design the glove pattern with the appropriate size and proportions to the user's hand, 2. design the connections for the sensors according to the glove pattern design, 3. design the electrical circuit, 4. sew the sensor connections and isolate them, 5. build the glove and place all the pieces together, and finally, 6. Program the glove to sense inputs. This process requires series of technical knowledge and many skills. In addition, there is a large error margin if users do not possess previous experience.

Proposal of a web-system for helping the user to design, build, and program embroidered smart-gloves

To solve these limitations and optimize the fabrication process, we present "The Smart-Glove Maker"; an end-to-end web system to design, build, and program smart-gloves. Our system converts the smart-glove pattern into embroidery patterns that can be executed by any embroidery machine. We propose a design of smart-glove for gesture detection. Our goal is that users can fabricate their own smart-

gloves, in the shortest possible time, with the least effort and using accessible and low-cost materials and tools.

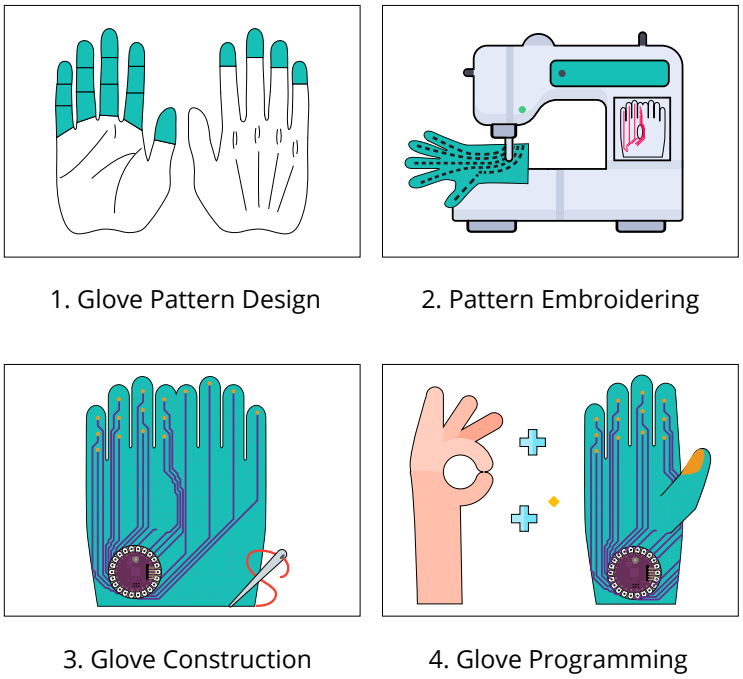


Figure 1.2: Steps in our pipeline for fabricating a smart-glove

The interface of our system divides the task into four non-technical skills, see figure 1.2:

Pattern design: Our UI allows users to arrange basic controls such as touch buttons in the design of the smart glove. To achieve this, we develop an algorithm for the auto-routing in non-rectangular areas. To produce a design, users select according to their preferences the set of buttons to include in the pattern. Based on their selection, the connections between the buttons and the sensing hardware are automatically calculated. Our algorithm generates the necessary files to be processed later by embroidery machines.

We development an algorithm for the auto-routing in non-rectangular areas.

Embroidery:Alternating between our system and the embroidery machine, the users proceed to embroider the patterns in an automated way. For the fabrication of the patterns, we use computerized embroidery. First, the machine stitches the connections with conductive thread, then per-

We use computerized embroidery as an automatic fabrication technique.

forms the isolation of the underlying circuit by an embroidery technique.

As part of the UI, we developed a visual guide for building the embroidery patterns.

Glove construction: To facilitate users in this task, we developed a visual guide as part of our UI. The guide explains step by step how to build the smart glove from the already embroidered patterns, and also the materials and tools needed.

Our System allows programming and testing of the smart-gloves.

Glove programming: Once the glove is completed, we provide source code for gesture detection. The provided code can be extended or even more robust code can be loaded depending on the needs of the user. To test the functionality of our smart-gloves, Our UI allows users to program gestures for the manipulation of actions through the gloves without the need for additional coding. We incorporate a use case in our system. We propose the manipulation of a music player interface. Users can map the gestures that the glove must recognize— e.g., play/stop, backward through a song and forward through a song. After that, users can trigger the actions in the music player by performing the corresponding gestures.

In summary, the contributions of this thesis are:

1. End-to-end web system to design, build and program smart-gloves using computerized embroidery as a technique for the automatic creation of wiring between the textile buttons and sensing hardware.
2. Auto-routing algorithm in non-rectangular areas for creating embroidery patterns. It applies the necessary constraints of smart-textiles and embroidery in general for the generation of the embroidered circuits.
3. The glove programming by gestures to control actions without the need for extra coding.

The thesis is organized as follows:

- **Chapter 2**: In this chapter, we analyze previous works of several researchers focused on the development of smart-gloves and their fabrication techniques.

- **Chapter 3**: We analyze a series of capacitive and resistive textile sensors with greater potential to be included in the design of the smart-gloves. In addition, we describe our initial prototypes of smart-gloves and their respective software to test their functionality.
- **Chapter 4**: We present our auto-routing algorithm for non-rectangular areas. We describe how to create the input pattern image, the functioning of the algorithm and its output files. Furthermore, we present a performance evaluation of the algorithm and its limitations.
- **Chapter 5**: We introduce our "The Smart-Glove Maker" system. In this chapter, we describe in detail the user interface and each step of our pipeline for the fabrication of smart gloves.
- **Chapter 6**: We present the evaluation of our UI and pipeline, along with the results and observations of the study.
- **Chapter 7**: We conclude with the summary of the contributions of this thesis and ideas about future work.

Chapter 2

Related work

Over the last decades, the use of smart-gloves for manipulating and interacting with objects and computerized environments has remained a topic of interest in the field of HCI.

So far there is much research related to smart-gloves but none offers a pipeline for designing, building, and programming of smart-gloves.

There are a considerable number of proposals and implementations of smart-gloves to address various tasks. Currently, hand motion tracking and gesture recognition comprise some of the most established implementations. Based on this form of interaction, authors like Gollner et al. [2012], Huber et al. [2013], and Plant et al. [2016] explored the possibility of using smart-gloves based on textiles to solve real-life tasks. However, until now there have been no proposals that allow automating the fabrication process of smart-gloves. In such a way that users can easily design, build and program functional smart-gloves according to their needs, without neglecting the aesthetics and ergonomics of these garments. Dipietro et al. [2008]

As an initial reference and to have a more straightforward idea of the technical requirements, we analyzed previous work related to materials, and construction techniques. Steimle [2015] developed lightweight and flexible capacitive sensors based on printed electronics. These kind of sensors can be adapted to different types of surfaces. On the other hand, Perner-Wilson and Buechley [2010]. developed a guide for creating sensing elements, circuits, and in-

terfaces based on textiles.their contribution focuses on the use of low-cost materials and conventional tools ease to access. huda Hamdan et al. [2018] introduced an interactive system for developing embroidered sensors. Their implementation stitches the underlying conductive thread that is later it is insulated with computerized embroidery. For our implementation, we use the same technique of stitching and embroidering.

we also analyzed previous work related to smart-glove that incorporate conductive textiles in their designs:

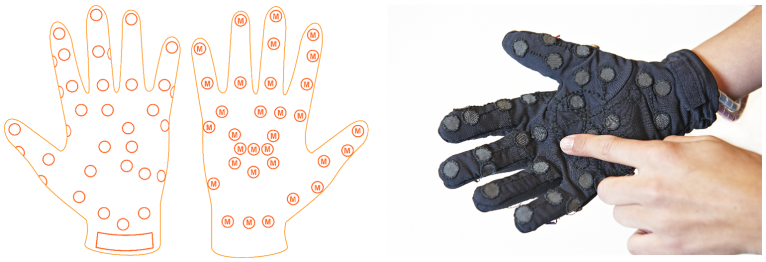


Figure 2.1: Gollner et al. [2012], Mobile Lorm Glove: Left side illustrates the distribution and positioning of the sensors vibrating motors (M), Right side shows the input unit on the palm of the glove.

Mobile Lorm Glove: Gollner et al. [2012] developed a smart-glove that acts as an interpreter of the Lorm alphabet. Lorm is based on tactile characters assigned to specific area of the palm. The bearer of the glove must trace with the finger the characters in the palm of the glove, the device interprets the strokes and converts them into text. Communication via Lorm Glove is bidirectional, the system can also interpret text and convert it into Lorm entries. In this way, communication barriers are overcome.

Mobile Lorm Glove contributes by proposing a glove design with remote communication that incorporates a broader number of sensors and reduces the hardware interface. Technically speaking, the input unit on the palm of the glove is equipped with a matrix of 35 textile pressure sensors for the recognition of strokes. The electrodes are constructed with conductive fabric while conductive thread is used for the connections to the control module. For the output unit located at the back of the glove, the glove design

incorporates a matrix of 32 shaftless coin vibrating motors, which serve for automatic feedback.

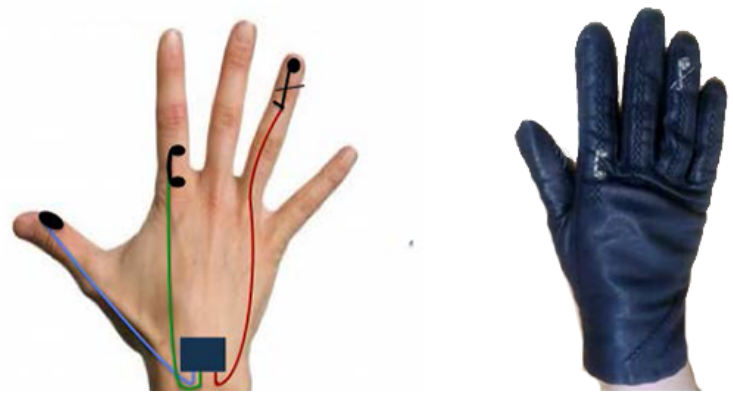


Figure 2.2: Huber et al. [2013], The TeleGlove: Left side shows the schematics and position of sensors. On the right the final prototype of the TeleGlove

TeleGlove: Huber et al. [2013] suggest the use of smart-gloves to complement the interaction with smart-phones or similar systems on a vaster scale. They developed rapid prototypes of smart-gloves that trigger common actions such as answering a call, terminating the call and muting the mobile. To execute these actions, it is enough to perform a predetermined gesture with the glove, in that case by pressing the combination of two fingers in the glove. Regarding the technical aspects, the gloves communicate with mobile devices through the ANT + protocol and incorporate textile handmade touch sensors. Their contribution focuses mainly on proposing the use of smart gloves in real environments as a device for habitual use.

Smart E-Textile Gloves: Plant et al. [2016] developed a smart-glove focused on patients that suffer from Parkinson's disease. The objective of their research was to create a smart-glove that could be used as a medical device. The the device can collect relevant information that determine the effectiveness of the medication in patients with this disease. For the researchers, it was important to develop gloves that were comfortable and aesthetic at the same time. Since the smart glove was intended to be used as a medical device, they wanted to avoid the use of standard electrical wiring that could cause any kind of stress on the patient. There-

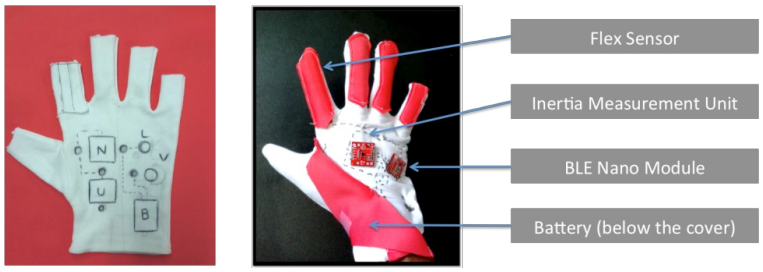


Figure 2.3: Plant et al. [2016], Smart E-Textile Gloves: The left side show the preliminary textile planning. The Right side displays the final smart-glove prototype that includes a flex Sensor, inertia measurement unit, BLE nano module and 3V battery.

fore, the use of e-textiles was a good choice because they found it to be discreet, flexible and easy to integrate with sensing hardware.

In summary, The glove is equipped with flex sensors, one on each finger. Thanks to the use of these sensors, they were able to analyze the progress of the disease, by measuring the joint of the hand. To make the connections between the flex sensors and the micro-controller, they used conductive thread and were insulated with cotton and neoprene. To control the sensors, they used a BLE nano module in order to be able to include wireless capabilities in the glove. (see figure 2.3)

Chapter 3

Prototyping and system requirements

In this chapter, we explore textiles sensors based on both resistive and resistive sensing including the materials and fabrication techniques. In addition, we established the distribution of the sensors in the glove to maximized on-hand inputs in the pattern design of our smart-gloves. Ultimately, a series of smart-glove prototypes were made to establish in this way the requirements of our system.

3.1 First prototype iteration

3.1.1 Soft sensor types

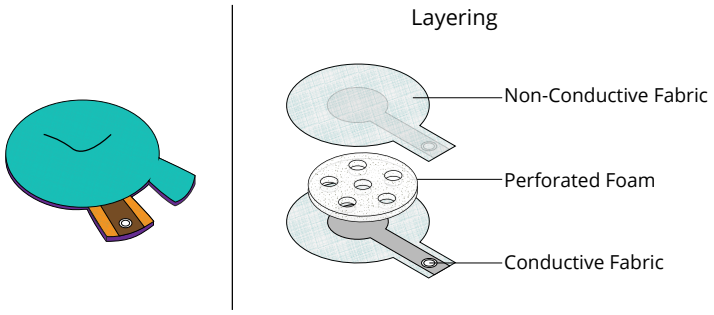


Figure 3.1: Layering of a resistive touch button

Resistive Touch Button: It is primarily made up of two layers of conductive fabric, which are separated by a layer of perforated sponge to avoid direct contact. Finally, each layer of conductive fabric is covered by non-conductive fabric at its respective end in order to isolate the button. By touching the sensor, the conductive elements come into contact which generates the voltage output. The voltage can be mapped to 1 when the button is pressed and 0 when it is released.

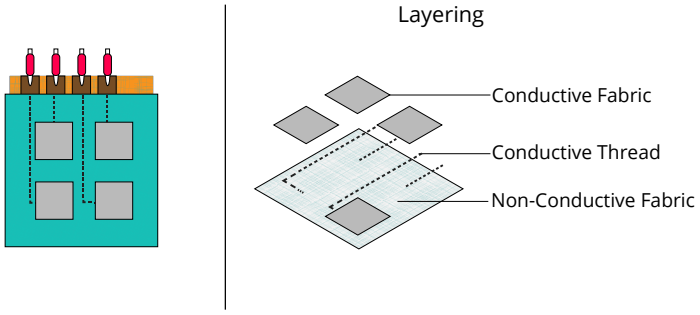


Figure 3.2: Structure of grid of capacitive touch buttons

Capacitive Touch Button: The structure of a capacitive button is even more basic. For its construction it is enough to have a conductive fabric electrode that can be attached to

the fabric by conductive thread. Even, using only conductive thread, the capacitance can be measured. Capacitive button sensors measure the difference of the signal in function to the surface that is touched and the distance between the electrode and the body. However, to do the sensing, it needs additional software.-e.g, Arduino's CapacitiveSensor library. The applications of these buttons can be expanded to create 2D trackpad and sliders. Figure 3.2 illustrates a 2x2 grid of capacitive buttons.

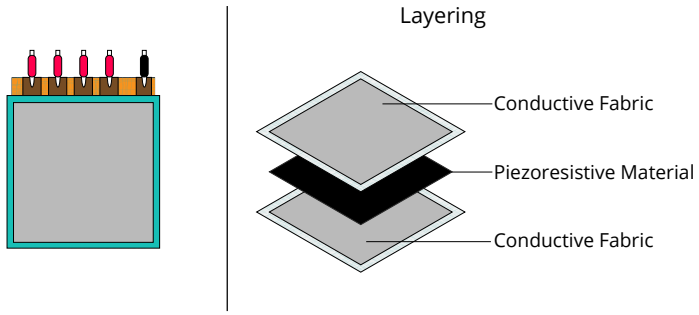


Figure 3.3: Layering of a resistive pressure sensor using piezoresistive

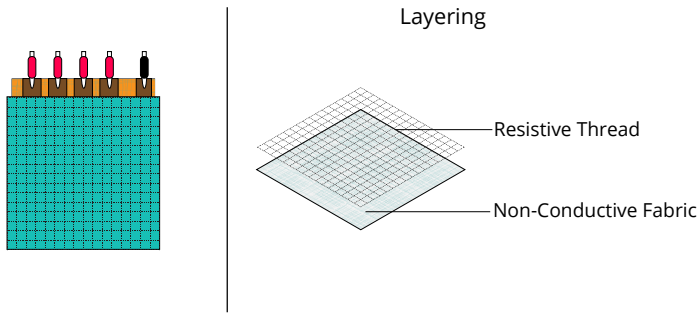


Figure 3.4: Layering of a resistive pressure sensor using resistive thread

Resistive Pressure Sensor: There a several techniques to develop a resistive pressure sensor. One can be made by layering piezoresistive material between two electrodes made of conductive fabric. See figure 3.3. The piezoresistive is a conductive material that is sensitive to pressure and changes the electrical resistivity of the electrodes across distance or when stretched or compressed. With this type of

structure, it is possible to measure the position and pressure. Another alternative proposed by Parzer et al. [2018] uses conductive thread as a base. Conductive thread are added to the fabric in the form of a grid, see figure 3.4. In such a way that when the intersections between the overlapped wires are pressed, the change in resistance can be measured by applying voltage to one of the wires and measuring the voltage drop across the other one.

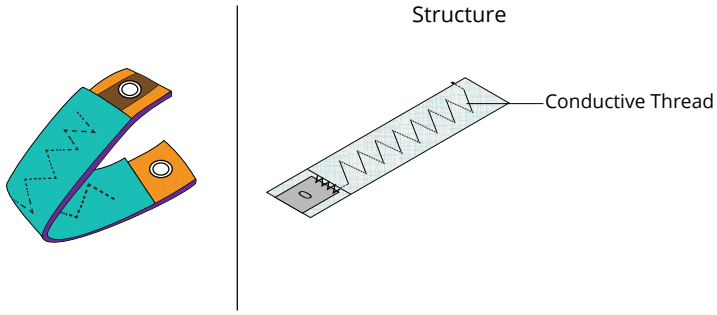


Figure 3.5: Structure of a resistive bend sensor

Resistive Bend Sensor: Vogl et al. [2017] designed a stretchable bend sensor that is constructed of a stretch fabric strip. In the strip, a zig-zag stitching is embroidered with conductive thread. This type of sensors are able to detect simultaneous modalities such as bending and stretching. The sensing is based on the changes in the resistance which is proportional to the length of the wire, and inversely proportional to the intersection of the area.

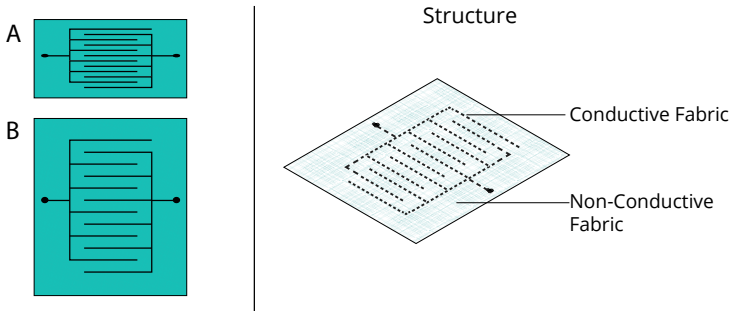


Figure 3.6: Structure of a capacitive bend sensor: A) Sensor when compressed, B) Sensor not deformed

Capacitive Bend Sensor: Keller et al. [2015] They invented strain sensors that incorporate a resistive strain gauge embroidered on a cloth layer. At the moment of deformation of this sensor, the electrical capacitance of the sensor is inversely proportional to the space at the distance between the adjacent seams. Figure 3.6.A exemplifies the bend sensor when it is compressed and 3.6.B shows its normal state.

3.1.2 Flex sensor

Flex Sensor: this sensor measures the amount of deflection. Because of its shape, this type of sensor can bend and flex physically with motion devices and has been extensively used in other smart-gloves implementations. Roy et al. [2015] and Plant et al. [2016] included flex sensor in their glove prototype for gathering information on finger position.

3.1.3 Sensor locations

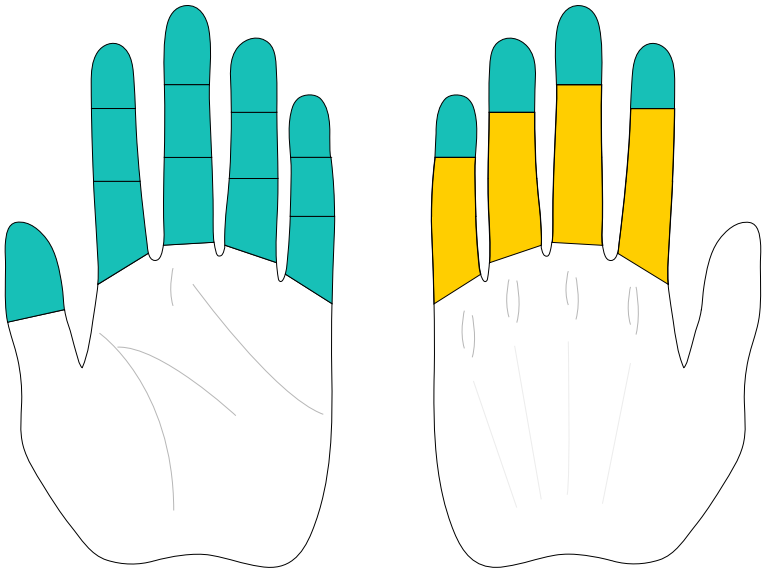


Figure 3.7: Distribution of sensors: In our prototypes we placed touch or pressure sensor in the green areas. In the yellow areas, bend sensors.

Oh and Findlater [2015] They investigated on-hand input modalities for users with visual impairments. They suggest a set of locations in the hand with potential to maximize on-hand inputs. They divided the hand according to the landmarks between the phalanges, thumb, and palm of the hand. Plant et al. [2016] used the back of the fingers to incorporate flex sensors.

Based on the contribution of Oh and Findlater [2015] and Plant et al. [2016], we included some of the proposed locations. Since our gloves use the thumb to activate the buttons, we chose the phalanges on the front of the hand and the nails to place sensors. In the case of including bend sensors, they would be incorporated over the joints of the fingers. See figure 3.7.

3.1.4 Fabrication techniques

Fabrication Technique	Machinery Costs	Material Costs	Process Complexity	Resistance to Wear
Embroidery	High	Low	High	High
Sewing	Low	Low	Low	High
Weaving	Low	High	High	High
Non-woven	Low	Low	Low	Low
Knitting	Low	High	High	Low
Spinning	Low	Low	Low	Low

Table 3.1: Qualitative comparison of e-textiles fabrication attributes - Table extracted from Gonçalves et al. [2018]

3.2 Implementation

The DIY community is predominantly use manual stitching with conductive thread to create soft intelligent gloves. We built the first prototypes completely from scratch; from the design of the pattern to the assembly of the sensors. In this way, we could detect which phases can be automated.

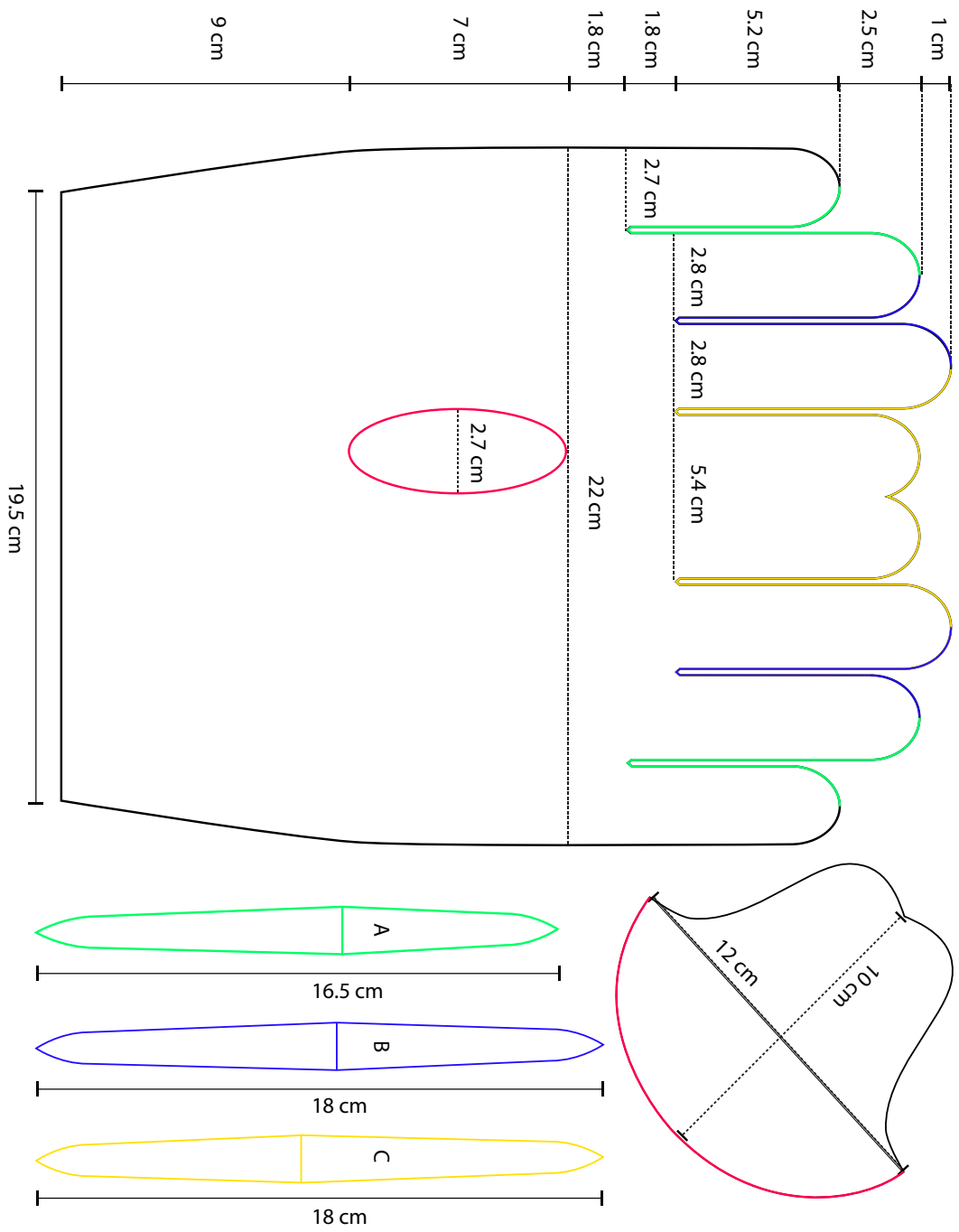


Figure 3.8: Glove pattern used in our final prototype

3.2.1 Hardware setup

Figure 3.9: Our first prototype includes pressure sensors and a bend sensor

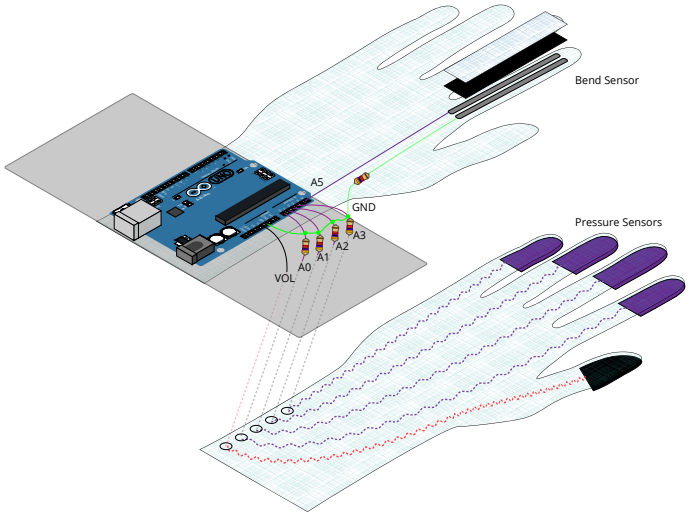


Figure 3.10: Diagram of the first prototype

In our initial prototype we used Arduino UNO as the micro-controller board. The smart-glove has four pressure sensors, one on each finger with the exception of the thumb. In addition, we included a bend sensor on the index finger. For the connections between the sensors and the Arduino, we sewed conductive thread from the fingertips to the wrist, then we completed the connections with common wiring to the Arduino. Figure 3.9 shows the final result of our first prototype and figure 3.10 shows its circuit diagram.

Figure 3.11: Smart-Glove prototype with a flex sensor

In the second prototype we integrated a flex sensor. In the previous prototype, pressure sensors and tactile sensors gave good results, and the performance was stable enough when testing with our software. However, the bend sensors did not work as expected. Although the output of these sensors had a considerable range in analog value, the performance was unstable and the output in many cases unpredictable.

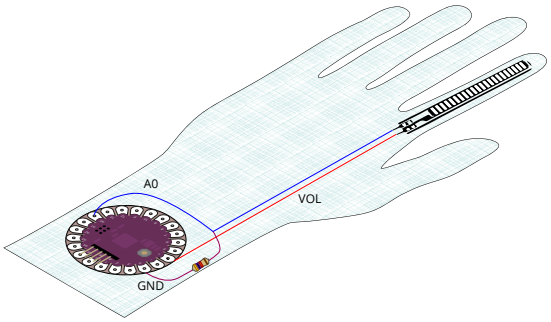


Figure 3.12: Diagram of the second prototype

We opted to make a second prototype which include a flex sensor. Our intention was to use all possible alternatives to determine which type of bending sensor suits the best in our design. In addition to this implementation, we replaced the Arduino UNO for an Arduino LilyPad.

3.2.2 Software setup

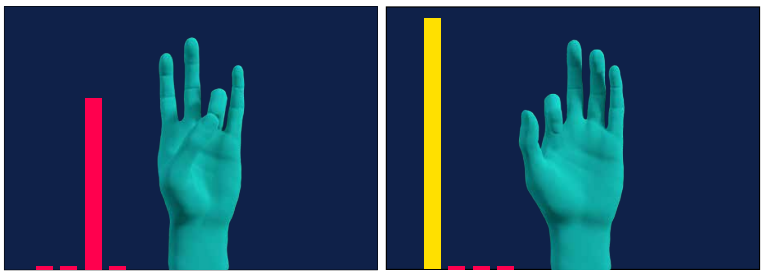


Figure 3.13: Software application for testing the first smart-glove prototype

To test the functionality of the first smart-glove with textile Sensors (Figure 3.9), we develop a basic application using Processing. Figure 3.13 shows two different screens of the application. On the left side a pressure sensor is been tightened. On the right side the bending sensor is triggered. The bars on the left side of each screen map the voltage for each sensing element. The red bars correspond to the pressure sensors, while the yellow bar represents the bending sensor. Each time a sensor is activated, its output value is reflected by its corresponding bar. Where the greater the voltage output, the larger the size of the bar.

Figure 3.14: Music player that changes audio pitch based on the smart-glove's input

To test the performance of our second smart-glove prototype, we developed a basic music player with JavaScript. The music player applies a low-pass filter to the audio. The audio changes its frequency according to the analogous values thrown by the flex sensor. The output range remained constant between ≈700 to ≈900 in analog value, hence its performance was relatively stable.

3.3 Smart-glove maker requirements

After the first prototype iteration, we defined the following requirements to be implemented in our system:

1. *Portability*: If we want to make smart-gloves to be more portable. Then we have to follow the path of textile sensors. Since hardware-based sensors are heavier, more robust, and more intrusive in comparison to textile sensors. In contrast, textile sensors are light, discrete and can be customized in size, shape, and color.

2. *Compactness*: Use LilyPad Arduino 328 instead of the Arduino UNO. The Arduino lilyPad is a controller

from the Arduino family. Its circular design is reduced to the maximum and is especially thought out to be sewn in e-Textiles. Hence, circuit connections can be consistently achieved with the exclusive use of conductive thread. In addition, this board provides 12 digital pins and 6 analog pins which gives us a considerable margin for the implementation of textile sensors in the same smart-glove.

3. *Design*: Create an easy glove pattern to build without compromising aesthetics and ergonomics. In addition, analyze what sizes of gloves and sensors are adequate to ensure the responsiveness of our smart-gloves.

4. *Routing*: Generate an algorithm capable of performing automatic routing of connections. Having in mind the essential requirements and constrains for e-textile- e.g, spacing between strokes, size and density of stitches.

5. *Automation*: Use of computerized embroidery as digital fabrication tool.

6. *Programming*: Implement a UI that coherently unifies all the steps that involve the fabrication process.

Chapter 4

Routing algorithm for non-rectangular areas

After the first prototypes of the smart-gloves we made, we realized that designing and constructing the textile circuits is one of the steps that consumes more time and effort. In addition, it is one of the crucial steps we found in the fabrication process. If mistakes are made in the design of the embroidered circuits and they are not detected in time, these mistakes could be dragged to other steps of the process. Therefore, this fact could compromise the progress made.

For the design of our pipeline, we wanted to incorporate tools that would allow us to automate the process and reduces the probability of error. Eichinger et al. [2007] automated the creation of embroidered circuits using PCB design tools and custom software. Its contribution lies in the conversion of the layout of a printed circuit board (PCB) to an embroidery format for sewing machines. As part of their work, they relied on Eagle, a tool for creating PCB designs. Eagle provides a set of rules for manual and automatic circuit routing. These design rules were useful in determining the size, spacing, and density of the stitch. Following the footsteps of Eichinger et al. [2007], we tried Eagle to create circuits automatically (Figure 4.1). However, we noticed some disadvantages:

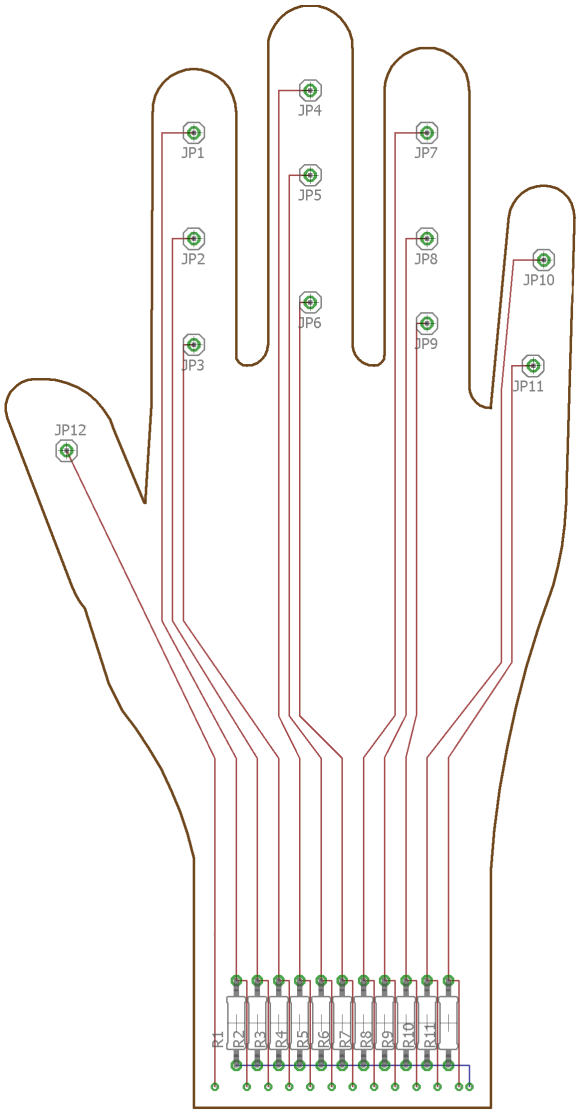


Figure 4.1: PCB layout designed with Eagle Software

Software limitations: Although Eagle offers auto-routing mechanisms, these are limited to rectangular areas and in standard versions, it restricts the dimensions of the PCB to 100mm x 160mm. Doing the manual routing in non-rectangular areas can consume too much time and effort. Especially because tools provided by Eagle to sculpt the layout of the board are not powerful enough as the tools provided by design graphic software such as Illustrator or Inkscape.

User constraints: Using Eagle without previous experience in PCB circuit design present a formidable challenge. To create a PCB design, it is necessary to demonstrate considerable expertise in electronics. Both to generate the schematic design and the routing layout of the PCB.

Based on our observations and the failed attempts to easily generate the circuit of our smart-glove using Eagle, we decided to develop our own algorithm. In this chapter, we present our auto-routing algorithm for non-rectangular areas. The characterization of our algorithm is based on the following:

1. Auto-routing of traces regardless of the shape and size of the area.
2. Ease to define routing points, i.e., position between textile buttons and micro-controller pins.
3. Guarantee of minimum spacing between traces, e.g., 2mm apart from each other.
4. Generation of pattern files to be processed later by computerized embroidery machines.
5. Reduction of human effort for designing embroidered circuits.

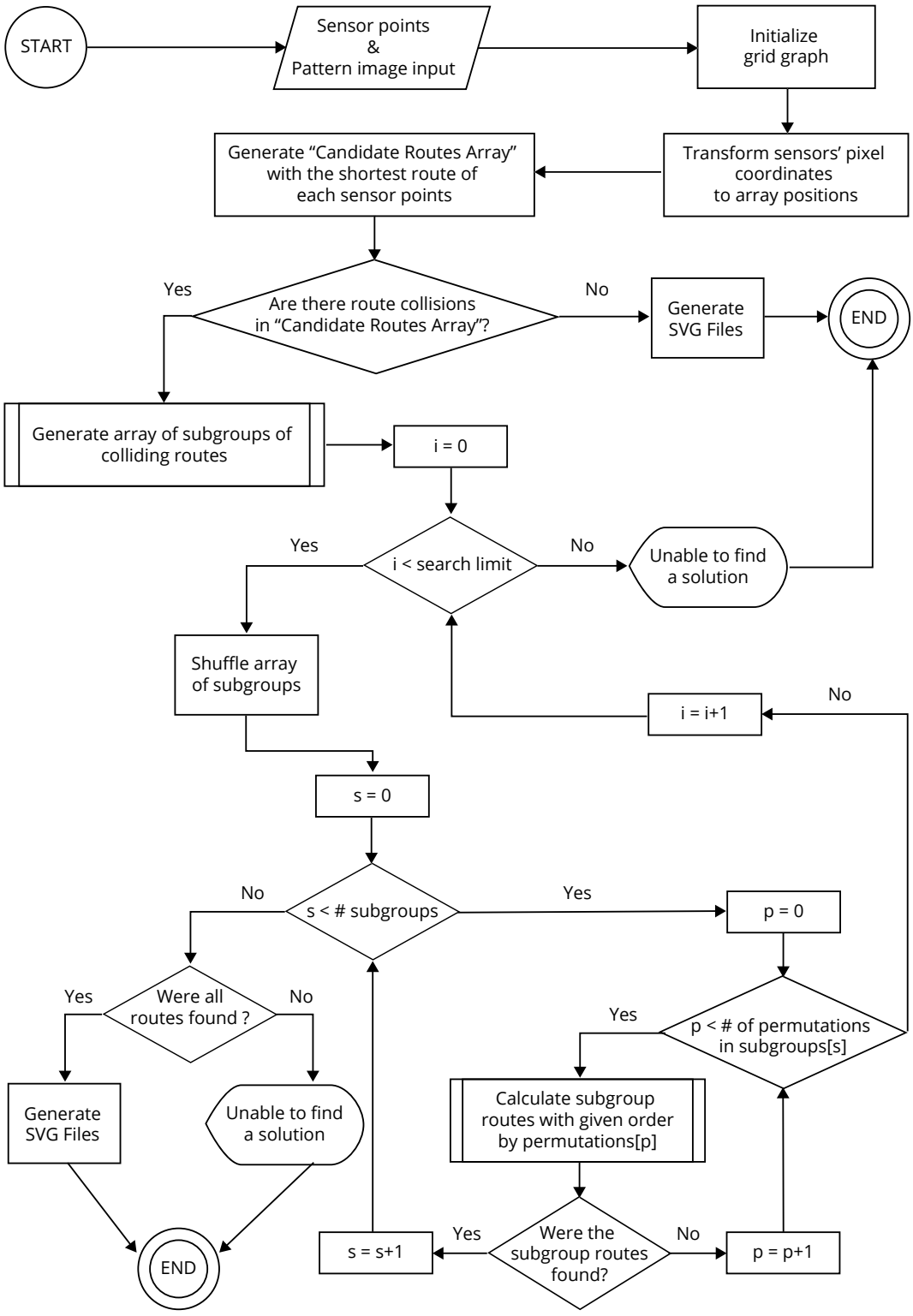


Figure 4.2: Auto-Routing algorithm chart flow

4.1 Algorithm definition

4.1.1 Definition of the input pattern and routing points

Figure 4.3: Input Pattern: The green area is where the auto-routing takes place. The transparent areas are denominated gaps and excluded when performing the auto-routing.

Input Pattern: the algorithm receives a pattern image in SVG format as input. The input pattern is used to determine which areas are enabled for the auto-routing. Figure 4.3 illustrates the different areas of the input pattern. This pattern may include one or more traced objects, regardless of their shape, orientation or size. The design can be generated with the help of any graphic design tool. When designing the input pattern, the following guidelines should be taken into account:

Routing Area: The filling of the routing area can be any solid color without any transparency.

Gaps: On the other hand, instead of filling, full transparency should be applied.

Pattern size: It is important that the image has the desired real dimensions — i.e., escalate it to life-size. In this way, the output files will have the same size as the input pattern.

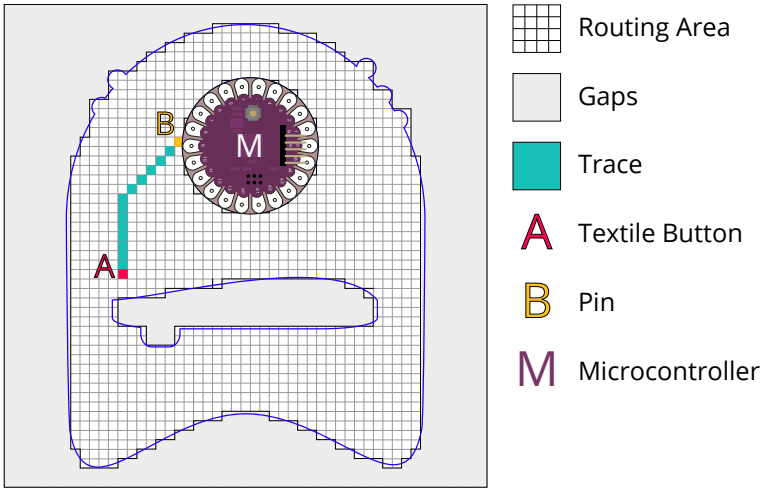


Figure 4.4: Snippet: Definition of routing points

Routing points: For each set of coordinates expressed in pixels between *A* and *B*, our algorithm finds the directest trace possible if exists one. Where A represents the position of the textile button defined by the absolute coordinates with respect to the input pattern. *B* represents the position of a pin of the micro-controller *M*, such that, *B*'s coordinates are relative to the center of M. Lastly, the center position of M is defined by the absolute coordinates with respect to the input pattern. Figure 4.4 exemplifies the routing between points A and B according to the input pattern.

Figure 4.5 provides an example to define the set of coordinates of textile buttons according to our implementation of the current system.

```
let pointsSet =  {
        //Definition of Position of the Button Sensor
        sensors:{
                1:{ startPoint: [265,550], endPin: "GND" },
                2:{ startPoint: [286,101], endPin: "2" },
                3:{ startPoint: [286,164], endPin: "3" },
                4:{ startPoint: [286,226], endPin: "4" }
        },
        //Definition of Pins' position relative to the micro-controller
        controller:{
                "GND": { endPoint:[10,-73] },
                "VOL":  { endPoint:[-10,-73] },
                "5":    { endPoint:[-35,-67] },
                "6":    { endPoint:[-55,-55] },
                "7":    { endPoint:[-61,-40] },
                "8":    { endPoint:[-70,-21] },
                "9":    { endPoint:[-73,0] },
                "10":   { endPoint:[-70,20] },
                "11":   { endPoint:[-61,40] },
                "12":   { endPoint:[-55,55] },
                "13":   { endPoint:[-35,67] },
                "A0":   { endPoint:[-10,73] },
                "A1":   { endPoint:[10,73] },
                "A2":   { endPoint:[35,67] },
                "A3":   { endPoint:[55,55] },
                "A4":   { endPoint:[61,40] },
                "A5":   { endPoint:[75,21] }
        },
        //Definition of Position of the micro-controller
        controllerCenterPoint: [182,700],
        controllerPixelSize: 142
}
```

Figure 4.5: Abstraction of routing points

4.1.2 Grid-graph

The grid-graph is defined by a two-dimensional array $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ with size $\mathrm{m} \times \mathrm{n}$.

Where $\mathrm{m} = (\mathrm{w}/\mathrm{u})$ and $\mathrm{n} = (\mathrm{h}/\mathrm{u})$, such that, u is the unit per cell in pixels , w the length of the image in pixels and h the height of the image in pixels.

The cells of the array represent the vertices of the graph, which correspond to the points of a coordinate plane,

where the $\mathrm{x} - \mathrm{coordinates}$ begin in the range of $0, ..., \mathrm{n}$ and the $\mathrm{y} - \mathrm{coordinates}$ begin in the range of $0, ..., \mathrm{m}$.

Each node is connected to its eight adjacent cells, such that a $\mathrm{V_{mn}}$ node has the edges of

$$E_{mn} = \{(V_m, V_{n+1}), (V_m, V_{n-1}), (V_{m+1}, V_n), (V_{m-1}, V_n), (V_{m+1}, V_{n+1}), (V_{m+1}, V_{n-1}), (V_{m-1}, V_{n-1}), (V_{m-1}, V_{n+1})\}$$

The graph contains the necessary information to perform the routing for each trace according to the specific restrictions of the input image. The nomenclature used to define the states by node are the following:

"Empty": Area available to calculate the trace.

"Obstacle": Segment where there is currently a portion of some previously calculated trace. This attribute allows us to avoid overlapping between traces.

"Gap": Area inaccessible for routing. This area is defined by the transparent parts of the input image.

"Start": Starting point of the trace.

"End": End point of the trace.

"Visited": visited nodes.

At run time, the nodes of the graph are marked with either *"Empty"* or *"Gap"*. To determine weather, a node $\mathrm{V_{nm}}$ has state of "Empty" or "Gap". Our algorithm is restricted to perform the auto-routing in non-rectangular areas, by analyzing the image pattern in the following way:

The image is divided into $\mathrm{m} \times \mathrm{n}$ subsections of size u. Each subsection $\mathrm{S_{nm}}$ is analyzed. If 80% of the pixels of $\mathrm{S_{nm}}$ have the alpha channel at 100% (transparent pixel) then their corresponding vertex $\mathrm{V_{nm}}$ is marked as "gap", otherwise it is marked as "Empty".

4.1.3 Auto-routing

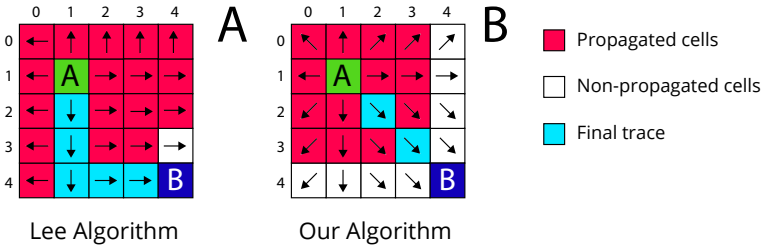


Figure 4.6: Comparison of outputs between Lee Algorithm versus our implementation

Our algorithm is based on the principles of Lee's maze-routing algorithm Lee [1961]. Lee's algorithm performs the search for the shortest path between two nodes of the graph using the "Breadth-First Search" technique. This technique consists in expanding all the neighbors from a root node. In the following propagations, each of the neighbors explores its respective adjacent nodes. And so on until the trace that connects the two nodes is discovered or the graph has been completely traversed.

Unlike the algorithm proposed by Lee, our implementation is able to find paths in eight directions *"South", "East", "North", "West", "Southeast", "Northeast", "Northwest", and "Southwest."* This feature allows us to calculate even directer paths than those calculated by Lee.

In figure 4.6, we show the shortest path calculated by each algorithm in a grid graph of 4x4. As we can see, the path of figure 4.6.A is of length 5, while the path of figure 4.6.B is of length 2. This represents a considerable shortening of the path, consequently, a reduction in computational cost.

The Auto-routing step by step - Calculating the most direct trace for each set of coordinates

Being A the root node and B the target node:

Step 1: We define a queue Q. This queue keeps track of the nodes that are candidates for the following propagations.

Step 2: Add root node *A(n,m)* to *Q*. Therefore, *Q = [(n,m)]*

Step 3: Iteratively extract the first element from *Q* until it is empty or node *B* has been found.

Step 3.1: Start the propagation, that is, add the adjacent neighbors with status "Empty" as soon as they are discovered at *Q[0]*

The order of exploration in our algorithm is predetermined with the following sequence: "South", "East", "North", "West", "Southeast", "Northeast", "Northwest", and "Southwest."

Step 3.2: If *Q[0] = B* then we have found the way, otherwise, mark *Q[0]* as visited

4.1.4 Heuristic

Each time that a new trace *T* is calculated, the obstacles generated in *T* while propagating a point are released in the grid-graph *G*. Then, the points of *R* and its adjacent cells are marked as new obstacles in G. In this way the next trace *T'* is calculated with the appropriate restrictions and overlapping of traces is prevented.

The time and space complexity for this algorithm for an *M* $\mathrm{m} \times \mathrm{n}$ *N* grid is *O (MN)*. To reduce the execution time, we propose the following heuristic (see figure 4.2).

Given a finite set *A* of n elements:

1. The shortest path is calculated for each trace in *A*.
2. If there are collisions between traces, subgroups of overlapping traces are generated.
3. Then, the subgroups are randomly calculated. By doing this the number of permutations reduces.

For example, without heuristics, if there is only one solution for the calculation of 15 traces, then in the worst-case scenario it will be needed $15! = 3628800$ combinations. Now, by applying the heuristic, suppose that we have 3 subgroups, each with 5 traces, then in the worst-case scenario the number of combinations needed is $3! * (5! + 5! + 5!) = 2160$.

4.2 Evaluation

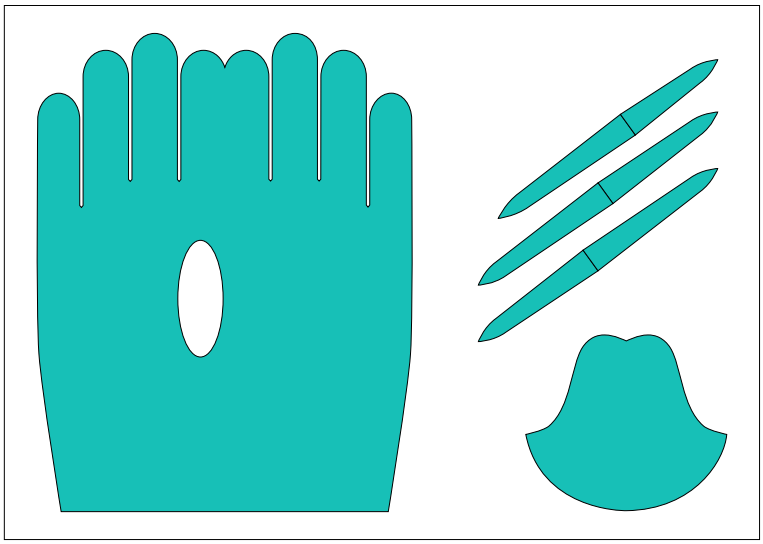
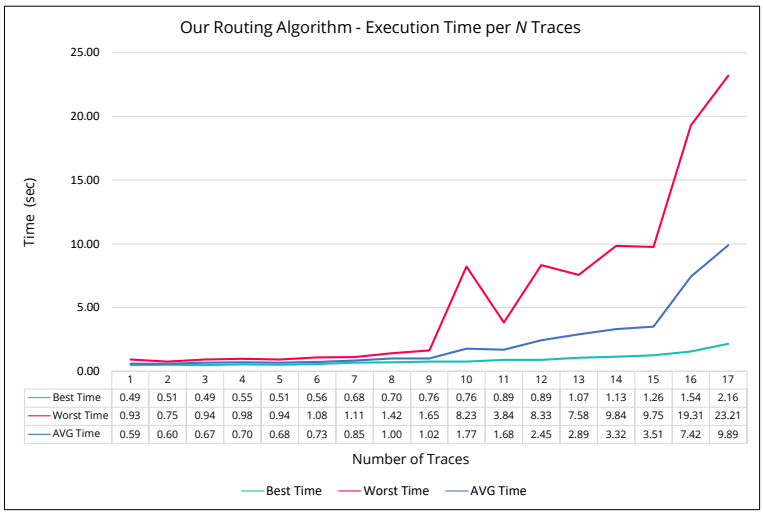


Figure 4.7: Input pattern used to evaluate our algorithm in comparison with Lee's algorithm

To evaluate the performance of our algorithm in relation to Lee's algorithm, we measured the execution time for the auto-routing of traces. We started from 1 trace to the 17 possible traces that our glove design can incorporate. For each measurement, 10 repetitions were executed. The selection of sensors was random and incremental, that is, the measurement started with one trace and adding randomly a trace after each exercise.

The input pattern we used is the current design for the fabrication of our smart-gloves, see figure 4.7. Lastly, the grid generated automatically by our algorithm based on the input pattern has a size of 121x99.



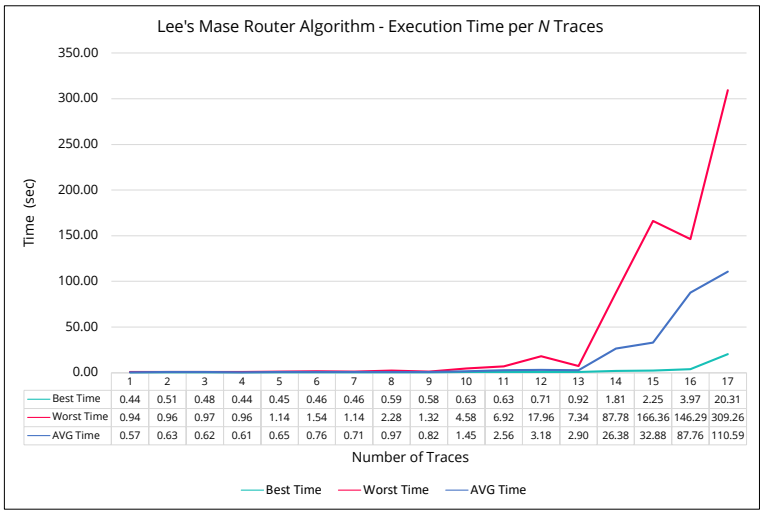
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Best Time	0.49	0.51	0.49	0.55	0.51	0.56	0.68	0.70	0.76	0.76	0.89	0.89	1.07	1.13	1.26	1.54	2.16
Worst Time	0.93	0.75	0.94	0.98	0.94	1.08	1.11	1.42	1.65	8.23	3.84	8.33	7.58	9.84	9.75	19.31	23.21
AVG Time	0.59	0.60	0.67	0.70	0.68	0.73	0.85	1.00	1.02	1.77	1.68	2.45	2.89	3.32	3.51	7.42	9.89

Figure 4.8: Our auto-routing algorithm execution time in N traces

In Figure 4.8, we show the best and the worst execution time, along with its average time of our algorithm. As we can see in the graph, time increases as the number of traces increases. However, the average time remains reasonable. This means that if one wants to generate the patterns of an smart-glove with the 17 possible connections, it will require an average of ≈ 10 seconds.

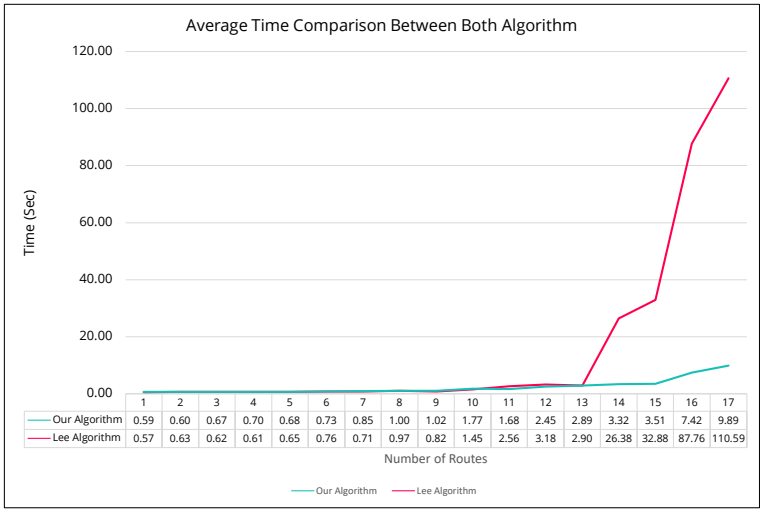
In figure 4.9 we can detect the similar behavior of the growing times as new traces are added. However, the execution times are drastically triggered by the combination of 12 to 17 traces. It means that the average time to calculate a glove pattern with the 17 traces would take ≈ 1.8 minutes. Approximately 11 times slower than our algorithm.

In Figure 4.10 we can observe the performance of both we algorithms.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Best Time	0.44	0.51	0.48	0.44	0.45	0.46	0.46	0.59	0.58	0.63	0.63	0.71	0.92	1.81	2.25	3.97	20.31
Worst Time	0.94	0.96	0.97	0.96	1.14	1.54	1.14	2.28	1.32	4.58	6.92	17.96	7.34	87.78	166.36	146.29	309.26
AVG Time	0.57	0.63	0.62	0.61	0.65	0.76	0.71	0.97	0.82	1.45	2.56	3.18	2.90	26.38	32.88	87.76	110.59

Figure 4.9: Lee's algorithm execution time in N traces



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Our Algorithm	0.59	0.60	0.67	0.70	0.68	0.73	0.85	1.00	1.02	1.77	1.68	2.45	2.89	3.32	3.51	7.42	9.89
Lee Algorithm	0.57	0.63	0.62	0.61	0.65	0.76	0.71	0.97	0.82	1.45	2.56	3.18	2.90	26.38	32.88	87.76	110.59

Figure 4.10: Average time comparison between both algorithms.

4.3 Limitations

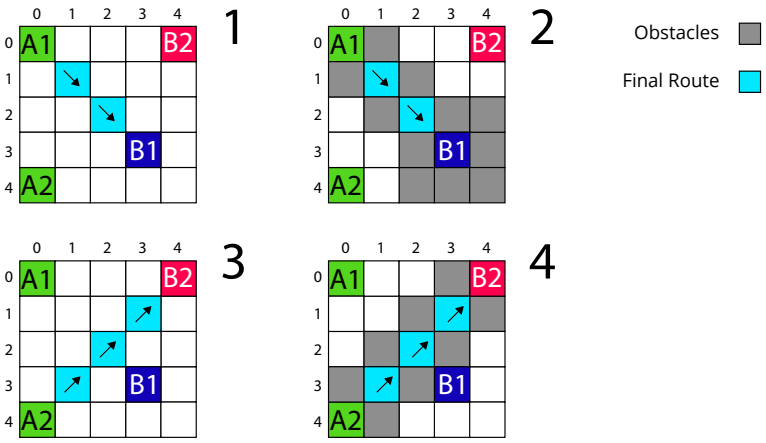


Figure 4.11: Interlock: Although there is a candidate trace that connects each pair of points, due to the added obstacles, only one path can be a resolved.

Interlock: There are some cases where the algorithm suffers from interlock even though there is a candidate path for the trace. This happens when the calculation of a paths blocks the access with obstacles. Figure 4.11 exemplifies a case when interlock occurs.

Performance: In general, the computation cost of the algorithm is very high. To improve the efficiency and the execution time there is the possibility of parallelizing the architecture of our algorithm. Yen et al. [1993] proposes strategies to parallelize Lee's algorithm, therefore these techniques could be adjusted and implemented in our algorithm.

Output: At the moment, our algorithm only generates glove patterns in SVG format. To obtain the final pattern with the format for embroidery machines, it is necessary to process the files manually with the specific software of each embroidery machine.

Chapter 5

Interface and pipeline

Figure 5.1: Home Section

In this chapter, we present our pipeline for the creation of textile smart-gloves, the "Smart-Glove Maker". Our pipeline enables users to design, embroider, build and program their gloves. Our pipeline integrates intelligence for the automatic design of smart-gloves patterns. We use computerized embroidery-machines as a digital fabrication tool along with low-cost materials and other accessible tools. Our user interface helps users step by step during the entire process, offering a visual guide to build the embroidered

gloves patterns. In addition, our pipeline allows programming the smart-gloves. By completing our pipeline users will have a functional self-made smart-glove for gesture recognition.

5.1 Designing the smart-glove

5.1.1 Choose buttons

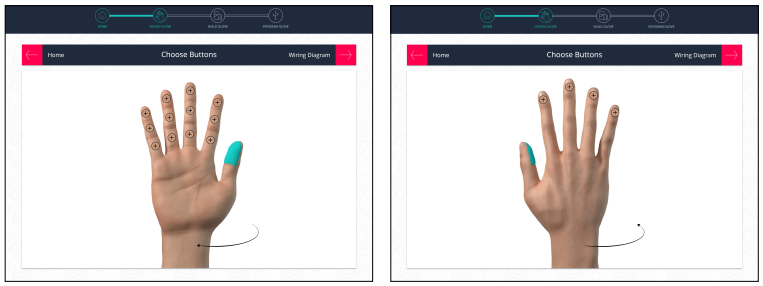


Figure 5.2: Design Section: Choosing buttons from both sides of the hand.

To design a smart-glove we use a catalog of possible gestures that can be recognized. A gesture can be triggered by touching in a burst of time a sequence of buttons. The figure 5.2 shows the mapping of the possible buttons that can be incorporated in the pattern; a maximum of 12 buttons on the palm of the hand and 4 at the back.

5.1.2 Show routings

Once the buttons are selected, the wiring diagram can be generated. In the figure 5.3 a proposal of a viable conflict-free wiring diagram of the auto-routing is shown. However, if there is more than one routing combination, it is possible to recalculate the wiring routing by pressing the button ”Recalculate Wiring.”

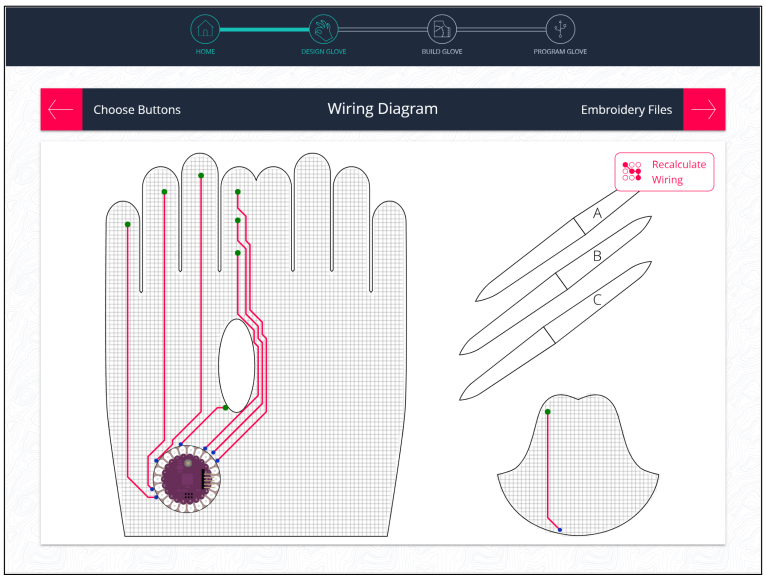


Figure 5.3: Design section: Displaying the wiring diagram

5.1.3 Download files

The concluding sub-step for designing a glove is to download the glove patterns, figure 5.4. It is pertinent to take into account that we currently use a standard glove size.

Image 5.5 shows the set of needed files to generate the embroidery patterns.

- 1.A Glove - Conductive File.svg
- 1.B Glove - Insulation File.svg
- 2.A Thumb - Conductive File.svg
- 2.B Thumb - Insulation File.svg

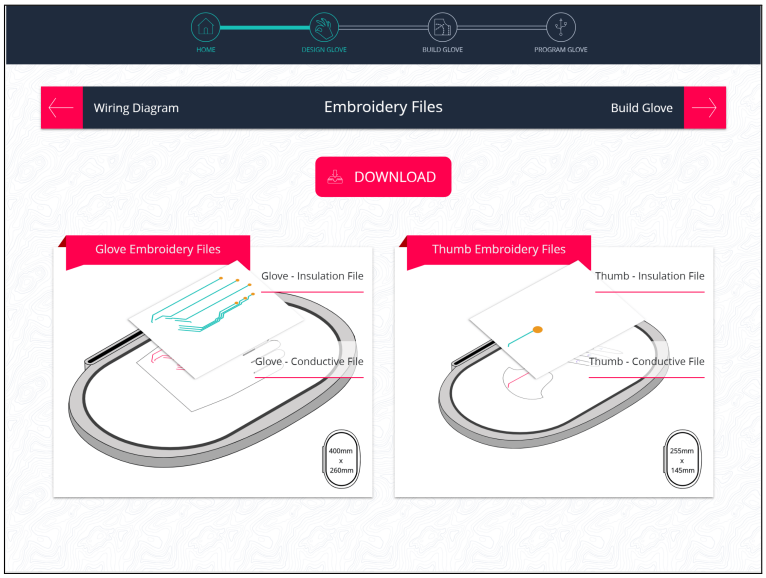


Figure 5.4: Design Section: Download smart-glove patterns' files

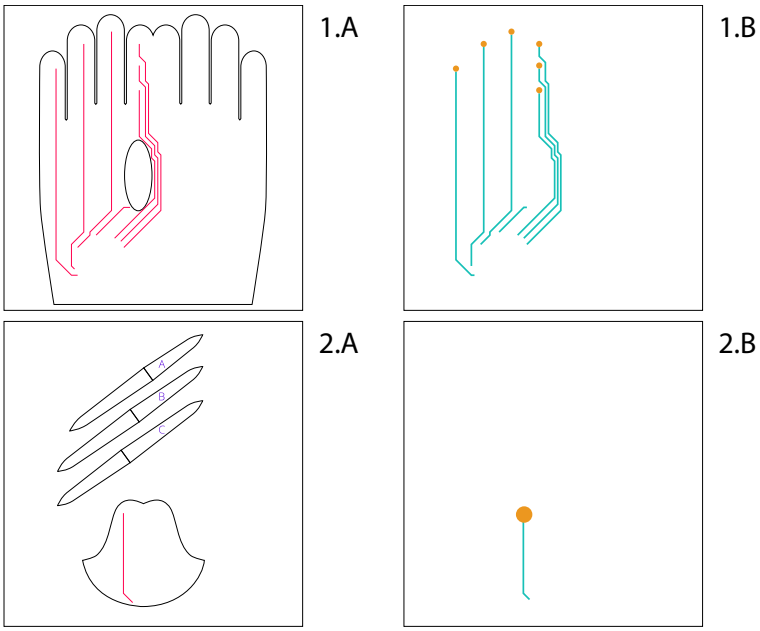


Figure 5.5: Embroidery Patterns Files: 1.A Glove - Conductive File, 1.B Glove - Insulation File, 2.A Thumb - Conductive File, 2.B Thumb - Insulation File.

5.2 Embroidering the smart-glove

huda Hamdan et al. [2018] introduced an interactive system for developing embroidered sensors. Their implementation stitches the underlying conductive thread that is later it is insulated with computerized embroidery. For our implementation, we use the same technique of stitching and embroidering.

5.3 Preparing pattern files

Before proceeding to embroider the files, they must be transformed into a format for the embroidery machine. In our case, to embroider our smart-glove pattern, we used the machine Bernina 880B along with its software DesignerPlus v.8.

To get the files for the Bernina 880B with DesignerPlus, we first digitize the file *1.A Glove - Conductive File* followed by the file *1.B Glove - Insulation File*. In a similar way for the thumb pattern but with file *2.A Thumb - Conductive File* followed by *2.B Thumb - Insulation File*. See figure 5.5 to relate the files.

The process to embroider the patterns is done in two parts. The first one to embroider the glove pattern and the second to embroider the pattern of the thumb and laterals. Each pattern is embroidered by layers. For this reason, it is critical to respect the order of the files when converting them to an embroidery-machine format.

After several implementations, we observed the following parameters to fit better to our design.

Function	Wire	Electrode	Insulation
Color	Red	Yellow	Green
Stitch	Single	Satin	Satin
Yarn Type	Conductive	Conductive	Regular
Width	1mm	2.5 - 3mm	2.5 - 3mm
Density	0.5mm	0.1mm	0.1mm

Table 5.1: Attributes and suggested measures for embroidering the buttons

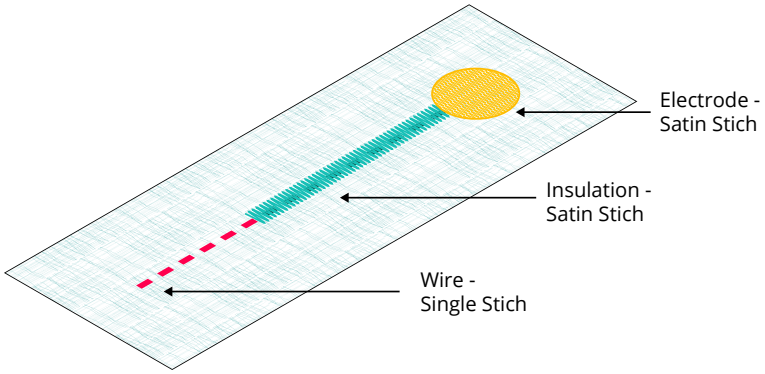


Figure 5.6: Layering of an embroidery touch button

Figure 5.6 shows the structure of a textile touch button and the type of stitch that should be used for each part of the button.

5.4 Embroidering the patterns

The order of the colors and the type of thread must also be respected when embroidering the patterns.

The order of colors and types of thread are the following:

1. Red with conductive thread
2. Yellow with conductive thread
3. Green with non-conductive thread
4. Black with non-conductive thread

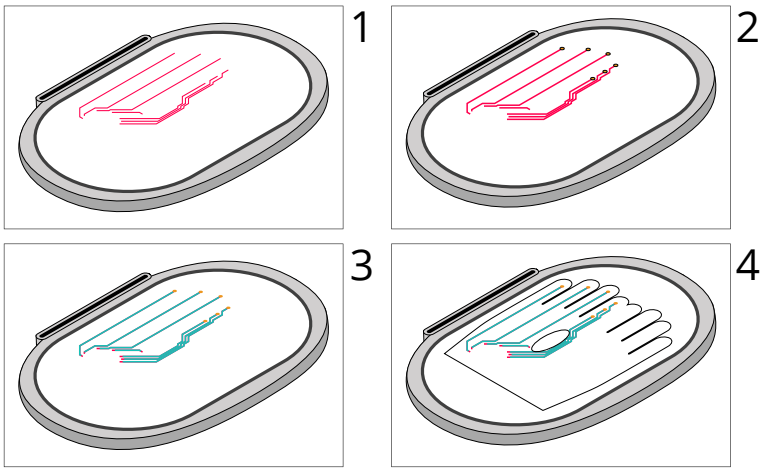


Figure 5.7: Sequence to embroider the patterns: 1. Wires, 2. Electrodes, 3. Insulation, 4. Outline

When the conductive traces (figure 5.7.1) and electrodes (figure5.7.2) are done, carefully, without removing the pattern from the hoop, prune the surplus stitches from both sides of the hoop if it is needed. Then on top of conductive traces, start embroidering with non-conductive thread the insulation traces (figure 5.7.3) followed by the embroidery of the glove's outline (figure 5.7.4). The same procedure must be done for the thumb, with patterns 5.5.2.A and 5.5.2.B.

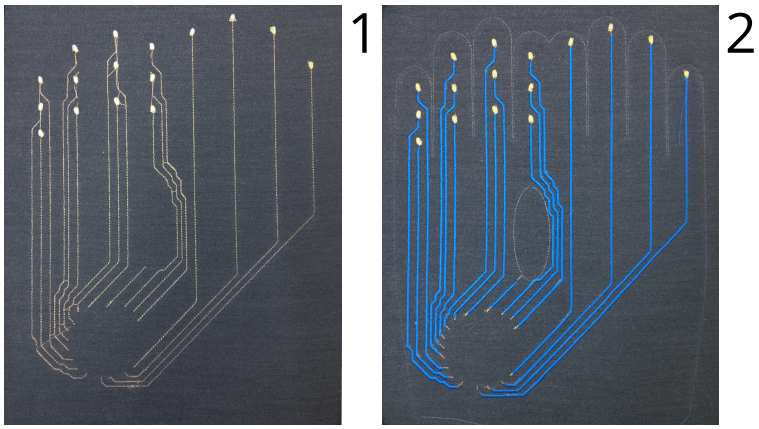


Figure 5.8: Embroidered smart-glove pattern: 1. Conductive layer, 2. Insulation layer

Figure 5.8 shows the layering of a smart-glove pattern.

5.5 Building the smart-glove

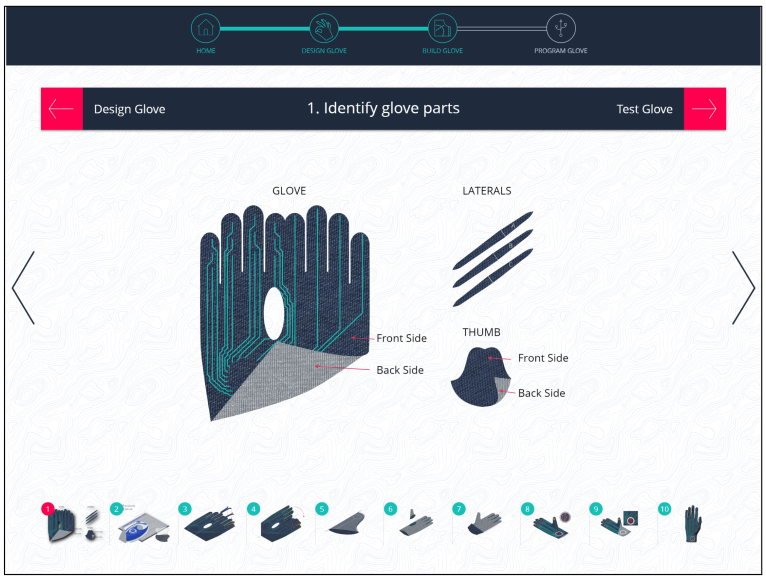


Figure 5.9: Building the smart-glove: Visual guide

To facilitate users to build the smart-glove from the already embroidered patterns, We provide in our user interface a visual guide. Each step in the guide contains descriptive images along with complementary instructions.

Below we describe step by step the instructions for building the smart-glove. See figure images/interface/directions.

1. Applying Fusible interfacing. (Figure 5.10.1) It may happen that in some occasions the isolation by the embroidery technique is not completely reliable. There are several reasons why exposed conductive segments or conductive threads may remain. One of the main reasons is due to the displacement of the fabric with respect to the hoop. This is principally because of the abrupt jumps of the machine when embroidering, also because of the use of fabrics with very thin and elastic fibers. Another common reasons are the density of the stitch, the thickness of the thread, and the speed of the stitching.

To reduce possible problems related to exposed circuit traces, apply a layer of fusible interfacing. To do so:

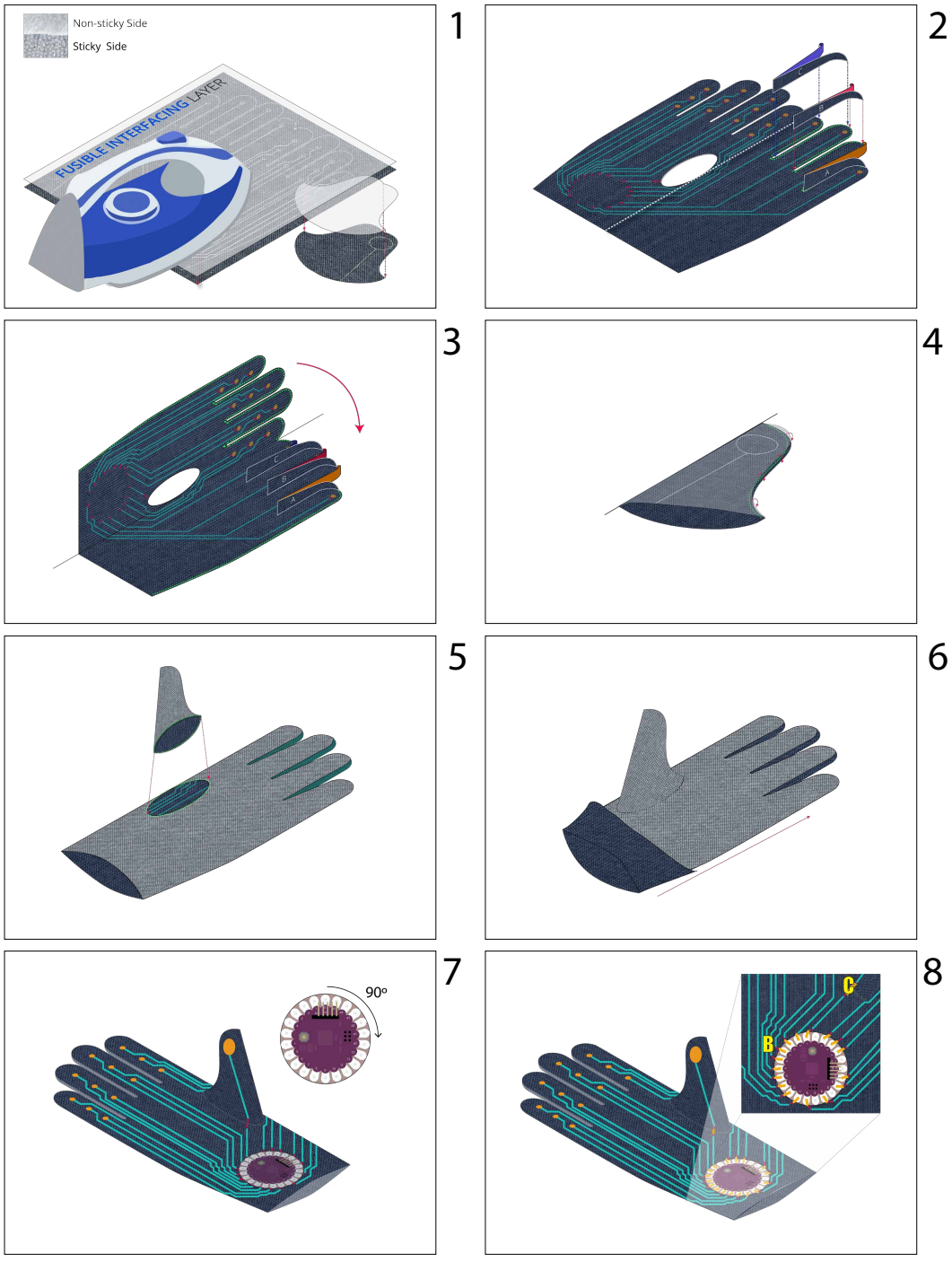


Figure 5.10: Visual guide - Instructions for making a smart-glove: 1. Apply fusible interfacing, 2. Sew Laterals, 3. Fold glove pattern and sew, 4. Sew up the thumb, 5. Attach thumb to the glove, 6. Flip the glove, 7. Place LilyPad, 8. Stitch Buttons to pins, 9. Upload Arduino Code

1. Place the pattern facing down on a flat surface.

2. Cut a piece of fusible interfacing that covers perfectly the pattern and place the sticky side down.

3. On top of it place a towel or another heat resistant fabric to avoid burning the interfacing. Iron at medium temperature until the interfacing is completely adhered. This process will take around 3 minutes.

4. Finally, cut the pattern leaving 2 millimeters outside the edge.

2. Sew Laterals. (Figure 5.10.2) Laterals are marked by a letter that represents their position:

Lateral A: Pinkie with ring finger

Lateral B: Ring with middle finger

Lateral C: Middle with index finger

If you are using a double sided color fabric, make sure that the front color of the lateral faces the front side of the pattern. In order to keep aesthetics, it is recommended to do the stitching on the back side of the pattern.

3. Fold glove pattern and sew. (Figure 5.10.3) Once the laterals are in place, fold the glove and start sewing it from fingertips to the wrist.

4. Sew up the thumb (Figure 5.10.4) As in the previous step, fold the thumb pattern and stitch it up. Do not forget to put the back side of the pattern on the outer side.

5. Attach thumb to the glove (Figure 5.10.5) Now it is time to put all together. Match both ellipses and start sewing, remember that the thumb must be pointing up.

6. Flip the glove. (Figure 5.10.6) At this step, the sewing of the glove pattern is done, flip the glove so that the real face is revealing.

7. Place LiliPad. (Figure 5.10.7) It is important to identify the correct position of the Arduino LilyPad. Sensors are designed in a way that the Arduino LilyPad must be placed with a rotation of 90 degree.

8. Stitch buttons to pins. (Figure 5.10.8) Thread a needle with conductive yard and sew each sensor line to its nearest pin on the Arduino LilyPad. Finally, stitch the line that leads from voltage to the thumb.

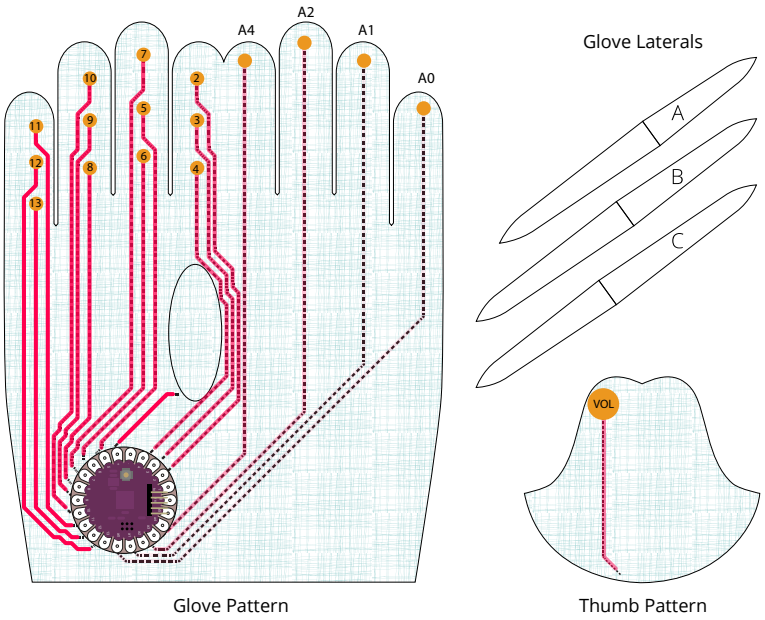


Figure 5.11: Circuit diagram with the connections between buttons and micro-controller.

Figure 5.12: Smart-Gloves: iteration refinement with embroidered touch buttons.

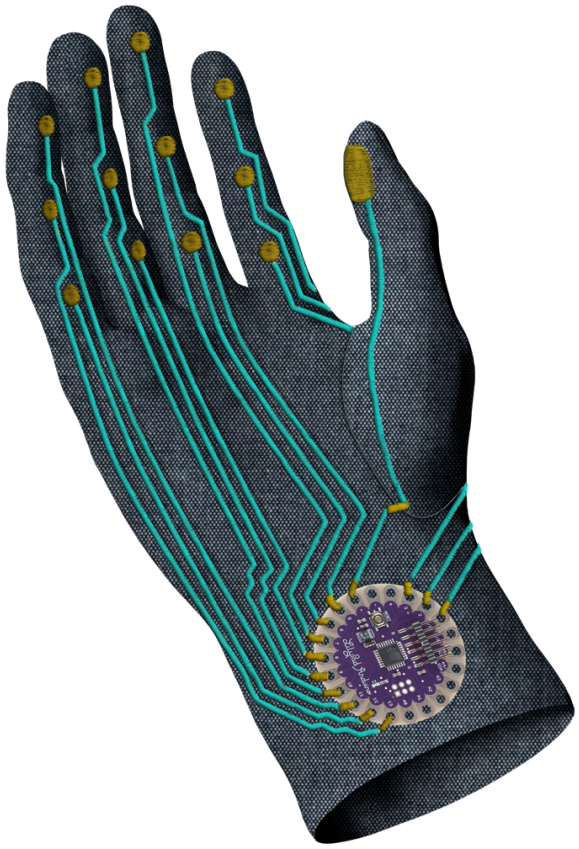


Figure 5.13: Final prototype iteration: We present a smart-glove for gesture detection

5.6 Programming and testing the smart-glove

5.6.1 Uploading Code to the Smart-Glove

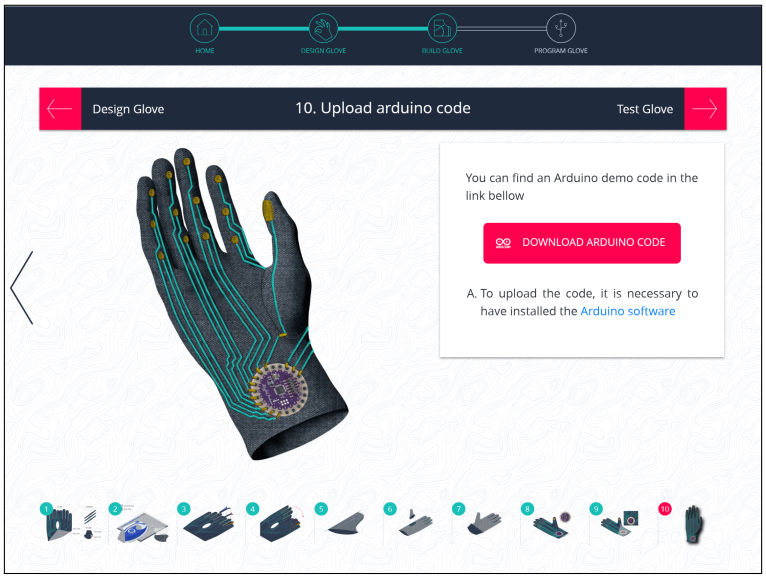


Figure 5.14: Downloading Arduino code for the smart-glove.

In order to start using our smart-glove, it is necessary to include code to detect the textile buttons' inputs. In our pipeline, we offer Arduino code ready to be loaded into the micro-controller. Directly download the code, see figure 5.14, connect the glove to a serial port USB and use the Arduino IDE to install the code to the glove.

At the moment we offer a straightforward solution for the detection of buttons' inputs. However, if necessary, users can program more robust routines that suit their needs.

5.6.2 Connecting the smart-Glove to the serial port

To use the glove in our system, the smart-glove must be connected to a serial port on the computer, then a panel

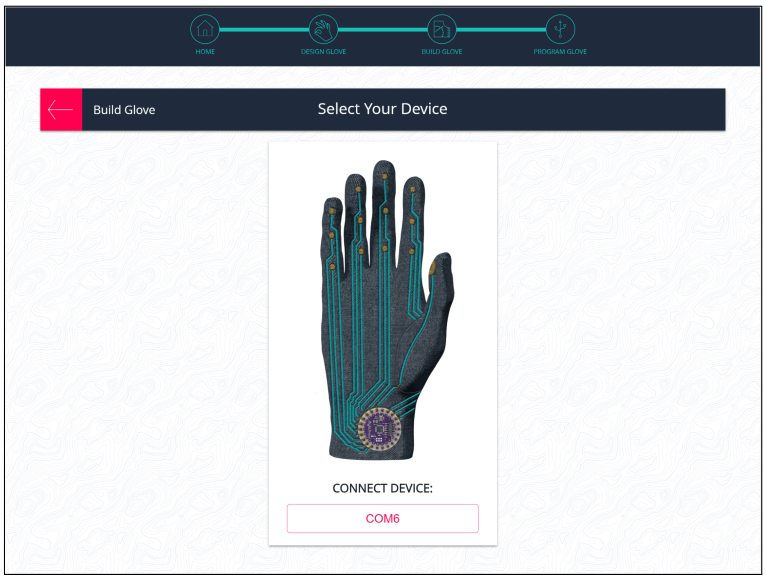


Figure 5.15: Connecting the smart-glove to the serial port.

with the available devices is displayed. When selecting the device, the glove will be automatically communicate with our system.

5.6.3 Gesture-based smart-glove programming

To assess the performance of our smart gloves, we offer a music player widget. The possible actions to manipulate the player are:

- Play/Stop a song
- Go forward through a song
- Go backward through a song

In figure 5.16, users can record the gestures for each action. The recording starts after clicking on the corresponding button of each action and followed by performing the desired gesture with the glove. After recording the action, a series of number is displayed, these numbers represent the sequence of the gesture.

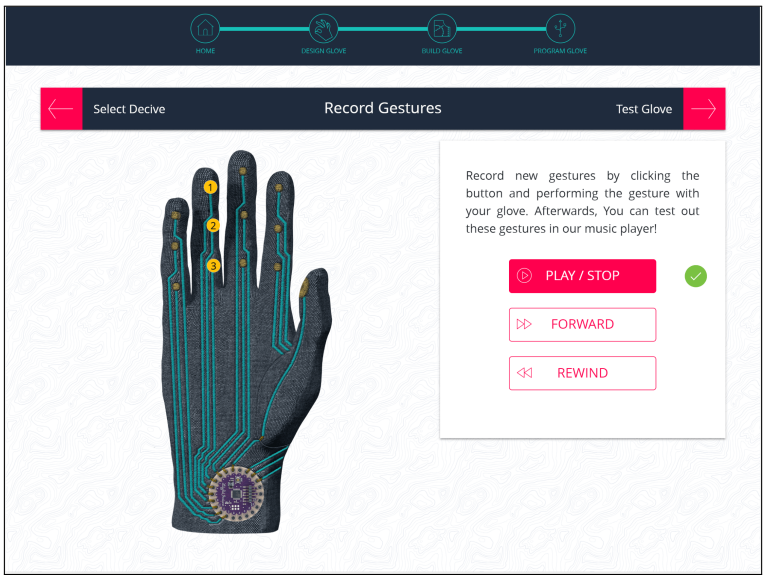


Figure 5.16: Smart-Glove programming using gestures

It is necessary to mention that for the moment the gestures are map to the interface, and not to the micro-controller itself. Therefore, if it is desired to develop other interfaces, it is the user's task to program the code for the detection of gestures. However, we offer this use case as an example of glove programming using gestures.

5.6.4 Controlling a Music Player Interface

Once the gestures have been recorded, users can proceed to control the music player widget (figure 5.17). Each time a button is touched, in the view, a golden circle will highlight the corresponding button sensor. By default, a test song is loaded. The music player will react depending on the gestures executed.

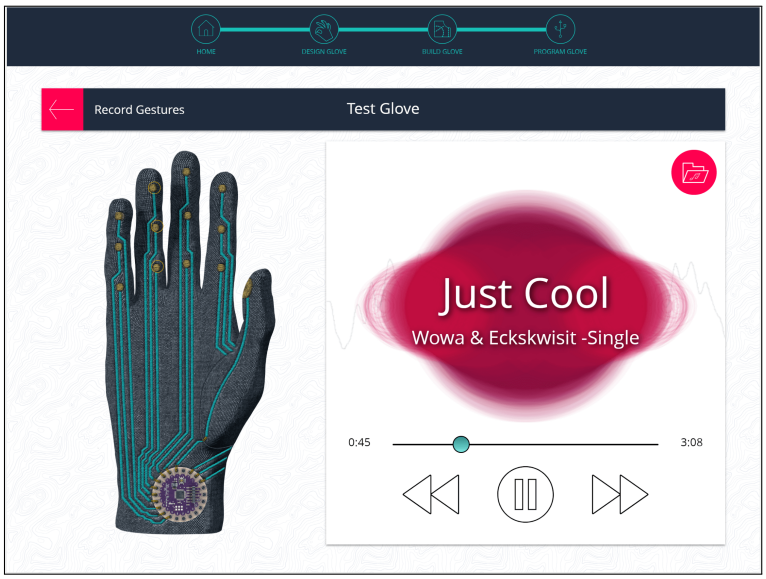


Figure 5.17: Controlling a music player widget

5.7 Software implementation

5.7.1 Back-end

To achieve communication between our system and the smart-glove, we developed a client-server architecture with the library *socket.io - npm v2.1.1*. For sending data through the serial port we use the library *serialport -npm, v7.0.2*.

5.7.2 Front-end

To develop the front-end, we use *ReactJS* as the main library for the development of the UI. We use *CSS3* for the aesthetics of the user interface and to provide a responsive UI, we use *Bootstrap v4.0*.

For animations, based on the ”3D hand” resource by SuperDasil under the 3.0 license. We make remixes of the 3D model with the help of *Blender*. The rest of the illustrations and images were made with *Illustrator* and *Photoshop*.

Chapter 6

Evaluation

The aim of this study was to analyze the usability of our pipeline and UI. In this sense, we wanted to verify that our system can guide the user in the design, construction and programming of a smart-glove with the least number of errors.

6.1 Design

In this experiment, participants were asked to complete our pipeline for fabricating a smart-glove by utilizing our system. The experiment was considered successful if participants managed to create a totally functional smart-glove and were capable to cope all steps that involves the pipeline.

For the evaluation of our pipeline, we have selected a combination of methods to ensure broad coverage of the results and a better identification of user interface problems. The methods included a task-based usability implementing the think-aloud method and proceeded with a series of questionnaires.

Task-based usability: The task-based usability evaluation consists of describing to the user the possible gestures that the smart-glove should be able to recognize at the end of our pipeline. The given gestures were selected in such a way that the user had to cover every element and screen of the system.

Video recording: The usability test was video recorded. The video recordings supported us to perform a better analysis of each participant throughout the pipeline by highlighting problems we did not realize during the experiment. In addition, they were asked to employ the think-aloud method, this method allowed us to obtain direct feedback on what participants were thinking during the trial.

Questionnaires: This approach were used to deepen the feedback of the user on each phase of the system. Two types of questionnaires were designed: one to evaluate the user interface and another to evaluate the functionality of the system. The purpose of implementing question sheets is to bid on qualitative measurements based on the usability trials. We presented two separate questionnaires to make the distinction of the participants' answers. If the questionnaires were delivered as one, it would have been difficult to detect whether the evaluation of a certain section highlights issues related to the interface or problems due to an inappropriate functionality of the system.

1. *Design questionnaire*: focuses on evaluating the clarity of the messages, alerts, sounds, animations and widgets.
2. *Functionality questionnaire*: aims on the performance of our pipeline. The questions asked in this questionnaire focus on detecting the degree of difficulty to perform certain task in the system, such as: select or modify the sensors in the design of the glove, download the files to generate the embroidery, connect the smart-glove and program its gestures, and so on.

In appendix B can be found the templates of the questionnaires that we used.

The results of this experiment were used to identify components or phases of our system that required redesign. The current version of our system reflects the new interface design and adjustments to our pipeline.

6.2 Task and procedure

The task consisted to fabricate a fully functional smart-glove utilizing our system. To achieve the task, the user should successfully complete the following steps:

1. Design of the glove with the adequate configuration of buttons:

For the design of the glove, the user was shown three gestures; two of these gestures involved the front of the palm and the last gesture the back of the hand. To exemplify the task in an explicit and straightforward way, the evaluator made the gesticulations required with his own hand. After the task was clear, the participant proceeded to do the design of the glove.

2. Construction of the smart-glove from the embroidered patterns of the glove:

For the construction of the glove, we use pre-embroidered patterns. The task of creating the embroidered patterns is a step that requires the use of embroidery machines. And also, additional software for the conversion of the files into a recognizable format for the embroidery machine. This step is not part of our UI, therefore, it was omitted. The task assigned to the participant, in this case, was to build the glove from the already embroidered patterns. The participant was given a finished embroidered glove pattern with the appropriate configuration of sensors.

For the glove construction, the user was provided with the necessary tools and materials. The toolkit included:

- Fusible interfacing
- Ironing towel
- Iron
- Needle

- Conductive thread
- Tread
- Scissors
- Arduino LilyPad

3. Programming and testing of the glove.

The participant was asked to program the glove based on gestures of his choice to later test the functionality of the glove when manipulating the music player interface.

6.3 Participants

Three participants were recruited, two women and one man. The age range of the participants was between 27 and 30 years old. Only women had previous experience in sewing.

6.4 Study setup

Figure 6.1: Study setup

During the experiment, all participants have performed the same task. The evaluator did not previously instruct the

participants in the use of the software but did clarify doubts regarding the assigned task. Once the trail had started, the evaluator could assist the participants only in the exceptional case that participants had doubts and were unable to resolve it by themselves. Figure 1.1 shows the setup for the study.

The time needed for completing this trail was around six hours and the protocol followed for this exercise consisted of the next three tasks:

1. *Glove Pattern Design*: This task was held in a private room for the user's comfort. In this place the participant received the requirements for the design of the glove. Doubts were clarified if existed, then the participant proceeded to generate the design of the glove. This task took approximately 5 minutes.

2. *Glove Construction*: A pre-embroidered glove pattern was provided to the participant for the construction of the glove. However, the user was taken to the FabLab to somehow involve him in the process of creating the embroidery pattern. Once in the FabLab the evaluator explained briefly the following necessary step to create a embroidered glove pattern:

   (a) Preparation of files for the sewing machine.

   (b) Installation of the fabric into the hoops.

   (c) Use of the sewing machine.

   Afterwards the participant was taken back to the initial room. Here he/she received a toolkit for the construction of the glove. This task took around 5 hours.

3. *Glove Programming*: Once the smart-glove was constructed, the participant proceeded to load the code in the Arduino LilyPad followed by the glove programming by gestures. To conclude this task, the participant had to control with the glove the interface of the music player that is built-in onto our UI. This step took approximately half an hour.

6.5 Results

The use of our system and the implementation of the pipeline has generally provided positive results. All the participants managed to complete the tasks and finish their smart-glove at the end of the study.

participants in general agreed that the design of the UI is clear and the functionality of the system is adequate

The result of the analysis of the questionnaires indicates that the participants in general agreed that the design of the UI is clear and the functionality of the system is adequate. However, there were some setbacks related to the construction of the glove, and some conflicts to understand the visual guide.

The problems that arose during the construction of the glove were mainly due to the lack of ability to sew. The users indicated that this part was the most tired, but there was a participant who mentioned that sewing the glove was relaxing. The results of this study dictate a subsequent redesign to correct possible problems of usability and improvement of the interface.

6.6 Discussion

The results of this study allowed us to detect and fix the usability problems of our system and pipeline

The results of this study allowed us to detect the usability errors of our system and pipeline. The problems related to the UI were corrected. In general, some icons, images, and animations were modified or replaced to make these resources more significant. We also restructured the interface in order to correct some navigation problems among the views. In the same way, we unify all the components of the interface to provide greater sense and coherence in the flow of our system. After the analysis of the videos, and the review of the questionnaires the following problems for each task were highlighted:

Glove Pattern Design

1. Selection of button sensors:

- *Problem*: Participants could easily intuit the concept of selecting the button set, however, there were design problems in the UI. Participants found the representation of the palm and the back of the hand slightly ambiguous. In addition, they felt that the icons on the buttons to switch between the front and back position of the hand were meaningless.
- *Solution*: The illustrations to represent the hand were replaced by a high detail 3D animation. In the current design of the UI, the buttons to switch between the sides of the hand were removed. Now to change the sides, it is enough to make a swipe on the 3D hand.

2. Download of pattern files:

- *Problem*: Participants found that the hints offered in this section were not clear enough to understand what to do with the downloaded files.
- *Solution*: In this view, animations were added to describe how to use downloaded files depending on the pattern (glove and thumb pattern). Also, the animations indicate the size of the hoop and the order in which the files should be embroidered.

Glove Construction

1. Understanding the visual guide:

- *Problem*: Regarding the visual guide, some participants commented that some illustrations lacked meaning. As feedback, they suggested incorporating more details in the illustrations. -e.g, point out where the stitches should be made when placing the parts of the glove together. As extra request, they suggested improving the navigation between instructions.

- *Solution*: In the current visual guide, the instructions are presented with less text and the information is more concise. We modified some of the illustrations and added more signifiers to increase their meaning. Finally, we restructured the way of visualizing each step of the guide. In such a way that now it is possible to obtain clues about what the following steps are.

2. Problems when sewing:

- *Problem*: Extensive problems were identified when constructing the glove. Unfortunately, this step requires great skill and effort. To complete the construction of the glove, it is needed an average of 4 hours. The direct reason was due to the little practice or experience to sew. We observed that this problem, in the less experienced users, led to insecurity and frustration.

- *Solution*: To minimize the time of this step, we explored other techniques. Initially, we tried with staples. Although we managed to unite the parts of the glove, the result was not aesthetic. Significant holes in the glove were exposed, and the staples did not support the constant use of the glove. As a second alternative, we modified the current design of the patterns so that it was possible to glue the parts of the glove together. However, we discovered that by using glue the process is even slower, even though we used special instant adhesives for textiles. Furthermore, as the glue dried, the glove became rigid and lost its mobility. For now, we can not clear up the problem and still need to try other alternatives to facilitate the construction of the smart-glove.

Glove Programming

1. Recording the gestures

- *Problem*: Participants had difficulties in understanding how to record gestures since the music player interface was shown at the beginning and the button's icon to shift between the views of recording gestures and the music player was not clear enough.
- *Solution*: To solve the issue, we restructured the way to program and test the gloves. First, the interface to record the gestures is presented, after recording, the interface to manipulate the music player is shown.

Chapter 7

Summary and future work

7.1 Summary and contributions

This research work was inspired by the process that the DIY community typically performs to make a smart-glove. The fabrication of an smart-glove requires several steps and many skills, from the design of the glove pattern, to the design of the connections, arrangement of the circuit, sewing and insulating the wires, joining all the glove parts, and finally, programming the glove.

Our goal is to unify the process for the creation of smart gloves.

The objective of this work was to develop a unified interface that will steer people through these steps. Using routing algorithms to reduce the time and margin of error in the design of textile sensors and wiring. We employed computerized embroidery in smart stitch patterns as an automatic manufacturing tool.

Our work is primarily divided into 4 phases: the first one focuses on the implementation of textile sensors and betas prototypes of the smart-gloves. The second addresses the development of the routing algorithm for the generation of glove patterns. The third describes the unification of the pipeline through a UI. Finally, the concluding phase focuses

on the evaluation of our pipeline.

Prototypes

During this phase, we explored the current implementations of resistive and capacitive sensors based on e-textile. These sensors included: touch sensors, pressure sensors, and bending sensors. The reason was to find out their performance, materials, and techniques for their construction. Based on the implementation of these textile sensors, we developed the first prototypes of intelligent gloves. At this point the manufacturing process was manual. To test the functionality of the gloves, we develop software to visualize the force exerted on the pressure sensors. And software to manipulate the pitch of the audio through flex sensors. We also investigate the locations to optimize inputs through the hand. In our design, based on the contribution of Oh and Findlater [2015] we use the phalanges to place sensors.

Routing algorithm

We developed an algorithm for the auto-routing of traces on non-rectangular surfaces

In this phase, we look for alternatives to automatically generate the glove pattern design. Our alternative was to use Eagle Software, but after several attempts, we discovered several setbacks,- i.e., lack of tools to generate auto-routing on non-rectangular surfaces. Faced with this situation, we developed an algorithm capable of performing the auto-routing of traces on non-rectangular surfaces. It is very easy to restrict the areas where routing is possible through an SVG format image as an input parameter. In addition, our algorithm can be configured to determine the minimum spacing between traces,- e.g, 2mm between strokes. The algorithm produces glove patterns, which are easy to process and convert to files recognizable by embroidery machines. Using the embroidery technique as a digital tool we ensure precision, consistency, and sturdiness of the stitches.

Pipeline and UI

We simplified the fabrication process into four non-technical steps.

At this point, we unified all the work done through a UI. Our end-to-end pipeline reduces the number of steps involved in the fabrication of the smart-gloves into 4 steps:

the design of the glove, the automation of embroidery of the circuit, the construction of the glove and its programming based on gestures.

Design: The flow of the pipeline begins with the selection of sensors from a catalog. Based on the selected sensors, our system generates files that contain the traces that will represent part of the embroidery circuit. At the moment our system does not generate the files with a recognizable format for sewing machines. However, with the use of additional software specific to each embroidery machine, the conversion of the files is relatively easy.

Embroidery of the pattern: Once having these files, and alternating between our system and the embroidery machines. By using the embroidery technique as an automated tool we ensure precision, consistency, and sturdiness in the stitches of the embroidered circuits.

Construction of the glove: To facilitate the user to build their smart-glove from the already embroidered glove patterns we included in our UI a visual guide. Our UI instructs people in every step of the construction through illustrations, animations, and text.

Glove programming based on gestures: Our system provides a basic source code so that the smart-glove is able to recognize the inputs of the buttons. Our interface provides a section where users can program the gloves by performing gestures. Gestures are mapped into actions which can control the interface of a music player.

Evaluation

The evaluation of our pipeline and UI allowed us to detect usability problems. The problems related to the UI were corrected. In general, some icons, images, and animations were modified or replaced to make these resources more significant. We also restructured the interface in order to correct some navigation problems among the views. However, extensive problems were identified when constructing the glove. To complete the construction of the glove, it is needed an average of 4 hours. The direct reason was due

to the little practice or experience to sew.

7.2 Future work

There are still interesting aspects that could be improved as future work.

Currently, our algorithm generates glove patterns using the Arduino Lilypad as the base microprocessor and in a way, each of the sensors is statically linked to a specific pin. Our algorithm could be extended to detect other types of microprocessors with a bigger or smaller number of pins. It could also include features in the interface to determine either automatically or manually which pin corresponds to each sensor.

Another aspect to take into account is the sizes of the gloves, for now, we only offer one size for the gloves. It would be convenient to look for alternatives to measure the exact size of the hand and based on this generate the dimensions of the patterns.

In our smart-glove design, we only implemented touch sensors. It would still be necessary to incorporate some other types of sensors.-e.g., pressure sensors, bending sensors, and sensors for detecting the position of the fingers and hand. The objective of this improvement would be to enrich the interaction of the gloves and perform more expressive gestures.

Finally, but not least. It is relevant to explore another type of materials and fabrics that adapt better to the shape of the hand. This will be important to evaluate if these materials improve the portability and use of the smart-gloves.

Appendix A

APPENDIX FOR THE ROUTING ALGORITHM

A.0.1 Auto-routing example

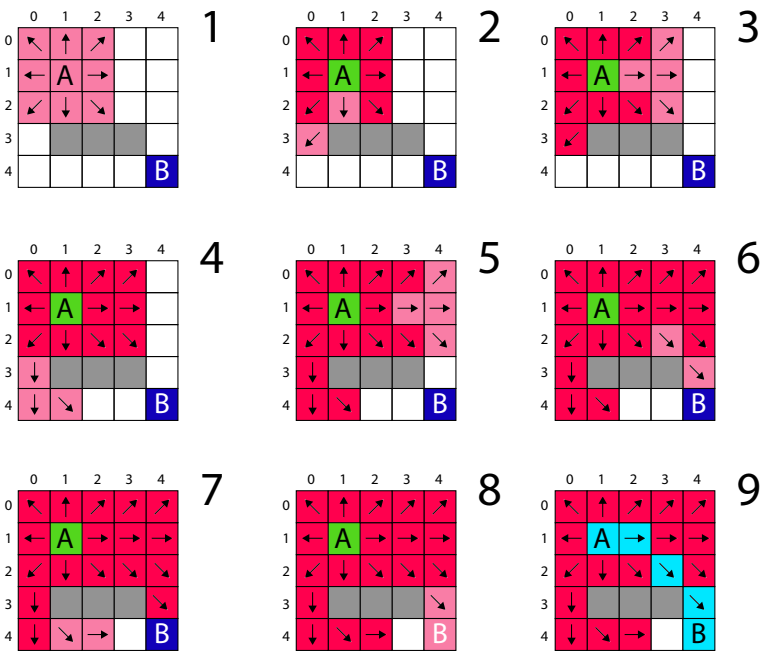


Figure A.1: Auto-routing example

Step 1: define the queue

$$Q = []$$

Step 2: Add node A into Q

$$Q = [(1,1)]$$

Step 3: Extract the first element of Q

$$Q_0 = (1,1),$$
$$Q = []$$

Step 4: propagate in the eight directions and add nodes to Q if they have an *"Empty"* state: (figure A.1.1)

$"South" : Q = [(2,1)]$
$"East" : Q = [(2,1),(1,2)]$
$"North" : Q = [(2,1),(1,2),(0,1)]$
$"West" : Q = [(2,1),(1,2),(0,1),(1,0)]$
$"Southeast" : Q = [(2,1),(1,2),(0,1),(1,0),(2,2)]$
$"Northeast" : Q = [(2,1),(1,2),(0,1),(1,0),(2,2),(0,2)]$
$"Northwest" : Q = [(2,1),(1,2),(0,1),(1,0),(2,2),(0,2),(0,0)]$
$"Southwest" : Q = [(2,1),(1,2),(0,1),(1,0),(2,2),(0,2),(0,0),(2.0)]$

Repeat until node *B* is found or the graph is traversed

(figure A.1.2)
Extract the next node:

$$Q_0 = (2,1),$$
$$Q = [(1,2),(0,1),(1,0),(2,2),(0,2),(0,0),(2.0)]$$

Propagate node *Q0*:

$$Q = [(1,2),(0,1),(1,0),(2,2),(0,2),(0,0),(2,0),(3.0)]$$

(figure A.1.3)
Extract the next node:

$$Q_0 = (1,2),$$
$$Q = [(0,1),(1,0),(2,2),(0,2),(0,0),(2,0),(3.0)]$$

Propagate node *Q0*:

$Q = [(0,1),(1,0),(2,2),(0,2),(0,0),(2,0),(3,0),(1,3),(2,3),(0,3)]$

Extract the next node *Q0 = (0,1)* "no propagated"
Extract the next node *Q0 = (1.0)* "no propagated"
Extract the next node *Q0 = (2,2)* "no propagated"
Extract the next node *Q0 = (0.2)* "no propagated"
Extract the next node *Q0 = (0,0)* "no propagated"
Extract the next node *Q0 = (2.0)* "no propagated"
Extract the next node *Q0 = (3.0)* "no propagated"

(figure A.1.4)
Extract the next node:

$$Q_0 = (3,0),$$
$$Q = Q = [(1,3),(2,3),(0,3)]$$

Propagate node *Q0*:

$$Q = [(1,3),(2,3),(0,3),(4,0),(4,1)]$$

(figure A.1.5)
Extract the next node:

$$Q_0 = (1,3),$$
$$Q = Q = [(2,3),(0,3),(4,0),(4,1)]$$

Propagate node *Q0*:

$$Q = [(2,3),(0,3),(4,0),(4,1),(1,4),(2,4),(0,4)]$$

(figure A.1.6)
Extract the next node:

$$Q_0 = (2,3),$$
$$Q = Q = [(0,3),(4,0),(4,1),(1,4),(2,4),(0,4,4))]$$

Propagate node *Q0*:

$$Q = [(0.3),(4.0),(4.1),(1.4),(2.4),(0.4),(3.4)]$$

Extract the next node *Q0 = (0.3)* "no propagated"
Extract the next node *Q0 = (4.0)* "no propagated"

(figure A.1.7)
Extract the next node:

$$Q_0 = (4, 1),$$
$$Q = Q = [(1, 4), (2, 4), (0, 4), (3, 4)]$$

Propagate node *Q0*:

$$Q = [(1, 4), (2, 4), (0, 4), (3, 4), (4, 2)]$$

Extract the next node *Q0 = (1,4)* "no propagated"
Extract the next node *Q0 = (2,4)* "no propagated"
Extract the next node *Q0 = (0.4)* "no propagated"

(figure A.1.8)
Extract the next node:

$$Q_0 = (3, 4),$$
$$Q = Q = [(4, 2)]$$

Propagate node *Q0*:

$$Q = [(4, 2), (4, 4)]$$

(figure A.1.9)

$$Route = [(1, 1), (1, 2), (2, 3), (3, 4), (4, 4)]$$

Appendix B

APPENDIX FOR THE EVALUATION

Appendix B contains the Questionnaires used after the Elicitation Study

Functionality Questionnaire

Rate your agreement with the following statements by checking the corresponding column	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Was the initial task given by the Moderator clear?					
Was it clear to follow the pipeline in general?					
Was it easy to navigate along the pipeline?					
HOME SECTION					
Was it clear HOME section?					
SENSORS SECTION					
Was it clear SENSORS section in general?					
Was it clear how to set and unset sensor?					
Was it easy to change between palm and back hand mode?					
Was it easy to generate/trace the glove pattern?					
Was it easy to modify the pattern?					
DOWNLOAD SECTION					
Was it clear DOWNLOAD sections?					
BUILD GLOVE					
Was it clear BUILD GLOVE section?					
Did the information provided in this section was enough to build successfully your glove?					

Figure B.1: Interface Questionnaire - page 1

Rate your agreement with the following statements by checking the corresponding column	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
TEST GLOVE SECTION					
Was it clear what to do in **TEST GLOVE** section?					
Was it clear how to connect and select your device?					
Was it clear how to test the glove? (how to perform the gestures)					
Was it clear the interaction between the glove and the music player?					
Was it clear how to record the gestures?					
Was it clear how to overwrite gestures?					
Was it clear how to display recorded gestures?					
Was it clear to identify when a sensor was touched?					
Was it clear how to perform gestures in general?					

Do you have any further comments regarding the functionality of our software?

Figure B.2: Interface Questionnaire - page 2

Interface Questionnaire

Rate your agreement with the following statements by checking the corresponding column	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Did the system respond always consistently to user actions?					
Were messages, alerts and sounds informative and accurate?					
Was it clear what different parts of the system do?					
Was the navigation bar useful to point out the progress made in the pipeline?					
Did the system protect against errors in user actions?					
Does the whole system design is homogeneous? (Components, widgets, buttons, images, etc.)					
Overall, the interface was pleasing and easy to use					
HOME SECTION					
Where animations clear and reflect the intended action between designing a glove and testing out a glove?					
SENSORS SECTION					
Were buttons for changing hand side intuitive?					
Was it clear how to change between selecting sensors and modify sensors					
DOWNLOAD SECTION					
Was it clear how to download the generated pattern?					
Was it easy to navigate along the directions?					

Figure B.3: Functionality Questionnaire - page 1

Rate your agreement with the following statements by checking the corresponding column	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
BUILD GLOVE					
Was it easy to follow instructions one by one?					
Did images match the intention of the instruction?					
Were images helpful to build your glove?					
Was it clear how to upload code into your lilypad?					
TEST GLOVE SECTION					
Were the music player components intuitive?					
Was it clear how to load a song?					
Was it clear the functionality of the “eye icon”?					
Was it easy to interpret the sequence of number that described the recorded gesture?					
Was the sound useful to let you know you trigger and action into the music player?					

Do you have any further comments regarding the design of our software?

Figure B.4: Functionality Questionnaire - page 2

Bibliography

L. Dipietro, A. M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, July 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.923862.

G. F. Eichinger, K. Baumann, T. Martin, and M. Jones. Using a pcb layout tool to create embroidered circuits. In *2007 11th IEEE International Symposium on Wearable Computers*, pages 105–106, Oct 2007. doi: 10.1109/ISWC.2007.4373789.

Ulrike Gollner, Tom Bieling, and Gesche Joost. Mobile lorm glove: Introducing a communication device for deaf-blind people. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, TEI '12, pages 127–130, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1174-8. doi: 10.1145/2148131.2148159. URL `http://doi.acm.org/10.1145/2148131.2148159`.

Carlos Gonçalves, Alexandre Ferreira da Silva, João Gomes, and Ricardo Simoes. Wearable e-textile technologies: A review on sensors, actuators and control elements. *Inventions*, 3(1), 2018. ISSN 2411-5134. doi: 10.3390/inventions3010014. URL `http://www.mdpi.com/2411-5134/3/1/14`.

Kim Huber, Mona Salmani, and Ylva Fernaeus. The making of the teleglove: Crafting interactions for basic phone use in the cold. In *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction*, TEI '14, pages 241–244, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2635-3. doi: 10.

1145/2540930.2540954. URL http://doi.acm.org/10.1145/2540930.2540954.

Nur Al huda Hamdan, Simon Voelker, and Jan Borchers. Sketch&stitch: Interactive embroidery for e-textiles. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 82:1–82:13, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5620-6. doi: 10.1145/3173574.3173656. URL https://doi.org/10.1145/3173574.3173656.

Sean Jason Keller, Tristan Thomas Trutna, David R. Perek, Bruce A. Cleary, and Brian Michael Scally. Embroidered strain sensing elements, 12 2015. URL https://patents.google.com/patent/US9816799B2/en.

C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, EC-10 (3):346–365, Sept 1961. ISSN 0367-9950. doi: 10.1109/TEC.1961.5219222.

Uran Oh and Leah Findlater. A performance comparison of on-hand versus on-phone nonvisual input by blind and sighted users. *ACM Trans. Access. Comput.*, 7(4):14:1–14:20, November 2015. ISSN 1936-7228. doi: 10.1145/2820616. URL http://doi.acm.org/10.1145/2820616.

Patrick Parzer, Florian Perteneder, Kathrin Probst, Christian Rendl, Joanne Leong, Sarah Schuetz, Anita Vogl, Reinhard Schwoediauer, Martin Kaltenbrunner, Siegfried Bauer, and Michael Haller. Resi: A highly flexible, pressure-sensitive, imperceptible textile interface based on resistive yarns. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 745–756, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5948-1. doi: 10.1145/3242587.3242664. URL http://doi.acm.org/10.1145/3242587.3242664.

Hannah Perner-Wilson and Leah Buechley. Making textile sensors from scratch. In *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '10, pages 349–352, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-841-4. doi: 10.

1145/1709886.1709972. URL http://doi.acm.org/10.1145/1709886.1709972.

L. Plant, B. Noriega, A. Sonti, N. Constant, and K. Mankodiya. Smart e-textile gloves for quantified measurements in movement disorders. In *2016 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pages 1–4, Nov 2016. doi: 10.1109/URTC.2016.8284077.

Kathika Roy, Durga Prasad Idiwal, Annapurna Agrawal, and Bani Hazra. Flex sensor based wearable gloves for robotic gripper control. In *Proceedings of the 2015 Conference on Advances In Robotics*, AIR '15, pages 70:1–70:5, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3356-6. doi: 10.1145/2783449.2783520. URL http://doi.acm.org/10.1145/2783449.2783520.

Lucian Alexandru Sandru, Marius Florin Crainic, Diana Savu, Cristian Moldovan, Valer Dolga, and Stefan Preitl. Automatic control of a quadcopter, ar. drone, using a smart glove. In *Proceedings of the 4th International Conference on Control, Mechatronics and Automation*, ICCMA '16, pages 92–98, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-5213-0. doi: 10.1145/3029610.3029619. URL http://doi.acm.org/10.1145/3029610.3029619.

Jürgen Steimle. Printed electronics for human-computer interaction. *interactions*, 22(3):72–75, April 2015. ISSN 1072-5520. doi: 10.1145/2754304. URL http://doi.acm.org/10.1145/2754304.

Anita Vogl, Patrick Parzer, Teo Babic, Joanne Leong, Alex Olwal, and Michael Haller. Stretcheband: Enabling fabric-based interactions through rapid fabrication of textile stretch sensors. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 2617–2627, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025938. URL http://doi.acm.org/10.1145/3025453.3025938.

I. . Yen, R. M. Dubash, and F. B. Bastani. Strategies for mapping lee's maze routing algorithm onto parallel architectures. In *[1993] Proceedings Seventh International Parallel*

Processing Symposium, pages 672–679, April 1993. doi: 10.1109/IPPS.1993.262800.

Index

Typeset November 12, 2018