A Musical Wearable:
Integrating Electronics into Clothing

by:

Yang Yang

Submitted to the Department of Architecture
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science at
the Massachusetts Institute of Technology

February 2011
January 19, 2011

Signature of Author.....
Yang Yang
Department of Architecture
January 19, 2011

Certified by .....
Terry Knight
Professor of Design and Computation
Thesis Supervisor

Certified by ..
.....
Leah Buechley
Assistant Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by .....
.....
Meejin Yoon
Associate Professor of Architectural Design
Director of the Undergraduate Architecture Program

A Musical Wearable:
Integrating Electronics into Clothing

by:
Yang Yang

Submitted to the Department of Architecture

January 19, 2011

in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science at
the Massachusetts Institute of Technology

Abstract

This project is an art project and a science project.
Traditional forms of art - music, dance, fashion -
are integrated with new technologies - electronics and software-
to create an item of clothing, or "wearable",
which creates music from the movements of the body through dance.

In this thesis I outline and explain how and what was done to create this wearable.

Thesis Supervisors:
Leah Buechley, Assistant Professor of Media Arts and Sciences, MIT
Terry Knight, Professor of Design and Computation, MIT

Table of Contents:

Introduction

Music, dance, and fashion have existed for a very long time. Electronics and software are new inventions by comparison.

Music, dance and fashion are instinctive human things. They are aspects of art, dealing with emotions and expression. Electronics and software on the other hand, are new and most often not understood. These new technologies appear magical – they can breathe life into inanimate objects and make almost anything happen. Integrating electronics and software with art allows for new inventions and interactions.

Electronic Fashion

In her book Fashionable Technology[1], Sabine Seymour writes that "all clothes have social, psychological, and physical functions... and through technology, the functions of clothing can be enhanced and new ones [can be] defined."[2]

Clothing has many functions.

Clothing is a tool that helps protect us from the natural elements. Clothing is a social construct, worn for modesty and for providing gender, social, political, and economical cues. Clothing is a framework for (self-) expression. It serves as a visual and external interface communicating who we are on the inside. We use it to signal, consciously and subconsciously, to others and to our environment. Sometimes items of clothing can be seen as works of art.

Fashion is a general term for the currently popular style or practice in clothing[3]. Clothing has existed since the beginning of humanity. As clothing evolved through time, a history of fashions is left behind. Our current time is frequently called 'the digital age'. It is characterized by the development of electronic technologies and the spread of these technologies to many aspects of our lives. And as fashion is a reflection of the current styles and trends, electronic fashion seems a very appropriate genre for the 21st century.

---

1 Seymour, Sabine 2009

2 Seymour, page 16, quoting Barnard (2002, 49 - 71)

3 "Fashion is the style and custom prevalent at a given time. In its most common usage however, "fashion" describes the popular clothing style. Many fashions are popular in many cultures at any given time...." - en.wikipedia.org/wiki/Fashion

How can the incorporation of electronics into clothing, enhance, extend, and change the aesthetics and functionality of clothing? To answer this question, I will first describe the technical requirements for the embedding of electronics into fashion, and then introduce three different groups of precedents from the realm of fashionable technology.

Technical Requirements

The integration of electronics is typically used to add dimensions of responsiveness or interactivity to clothing. There are several crucial components, listed and explained below:

An important element is a **microcontroller**. A microcontroller is a single chip computer capable of running and storing a program. Embedded in clothing, a microcontroller acts as a brain, collecting and computing data from various input sources and using this data to act on output devices. The Lilypad Arduino[4] is a microcontroller especially designed for use in e-textiles and electronic fashion.

As a second skin, an item of clothing worn on the body has available to it a great variety of possible **inputs**- means of sensing information about the world- both from the person wearing it and from the external environment. Some of the possible input sources from the wearer are pressure, bending, motion, orientation, displacement, and acceleration. Some possible inputs from the environment are light, sound, and temperature. Sensors are used to sense and convert the different types of input signals into electronic data (in the common form of varying voltages) for a microcontroller to read and use. A great variety of sensors exist for sensing all the different kinds of possible inputs. Typically, they are hard components, made of metal and plastic. Textile sensors[5], soft sensors made of conductive fabrics and threads instead of plastic and metal components also exist and are especially fitting for applications in clothing.

---

4 Appendix A; http://www.arduino.cc/en/Main/ArduinoBoardLilyPad

5 Appendix B; http://plusea.at/downloads/HandcraftingSensorsWS_sm.pdf

The data gathered from the input sources are used to act on **outputs**. Output devices generate the response that we can detect through our five senses of vision, touch, sound, scent, and taste. LEDs and thermochromatic inks are examples of visual output devices; speakers are examples of audio output devices; motors and shape memory alloys are examples of tactile output devices; scent and taste modules are examples of olfactory and gustatory output devices.

Software is the glue that integrates this system of microcontroller(s)-input devices-output devices. Software is a program (a set of instructions appropriately written for a microcontroller to understand) that tells the microcontroller how to gather, read, and manipulate input data, and how and when to act on output devices (depending on the input data, but not necessarily so).

An **energy source**(s) is needed to power the system. This can come from batteries, solar cells, or be generated by the wearer.

Lastly special materials like conductive fabrics and threads are integrated with traditional materials to create a functional garment.

Precedents

The field of electronic fashion has produced many interesting works. Below I introduce three works to highlight some different areas being explored in the field.

The following examples are works by three different designers. Each designer works to create items of clothing with embedded electronics, and all of the pieces include the key technical components explained above. But though each designer works within the framework of fashion-technology and share similar building blocks, their works are very different from each other.

Figure 1: Hussein Chalayan (image from Fashionable Technology by Sabine Seymour[1])

Hussein Chalayan is a fashion designer who uses embedded technology to enhance the aesthetics of clothing and to challenge the idea of clothing as a static artifact (figure 1).

His creations belong in the realm of high fashion, taking on innovative, sculptural forms that can be categorized as art. In *Airborne*, 15,600 LEDs display abstract films corresponding to the moods of different seasons. In *One Hundred and Eleven*, a mechanical dress transforms its shape morphing across 111 years of fashion history.

Chalayan's work *Airborne* takes the idea of clothing as visual display and pushes its theoretical implications with the help of technology, turning the entirety of the piece of clothing into a light-filled and changing display. In working with light, 'the only medium without a message, continually conveying itself' [2], Chalayan achieves a fashion effect impossible through traditional methods.

One Hundred and Eleven challenges the idea of clothing as a static artifact and is also a beautiful exploration and visualization of the evolution of fashion through time. This is again achieved through the integration of technology into fashion, and impossible through traditional methods.

---

1 Seymour, page 30

2 McLuhan

Figure 2. CuteCircuit (image from Fashionable Technology by Sabine Seymour[1])

The works of CuteCircuit (figure 2) explore ways that the integration of fashion and technology can extend the functionality of clothing.

The *M-Dress* is a silk jersey dress that accepts a SIM card and works as a functional mobile phone. A system of integrated sensors and gesture recognition software allows the wearer to pickup and answer calls by bringing her hand to her ears. The *Hug Shirt* uses wireless technology to exchange messages in the form of physical sensations between wearers. And the *SAAB Lifestyle Garment* combines display capabilities with network connectivity in garment form, allowing the user to access, manipulate, and display a wide range of information through the interface of clothing.

The looks of the works of CuteCircuit are rather conservative, taking the form of commonplace items of clothing, but the implications of these pieces of clothing are radical. They show how disparate functionalities can be incorporated into fashion, creating unique and practical tool-clothing.

1 Seymour, pages 38 - 40

Figure 3. Despina Papadopoulos (image from Fashionable Technology by Sabine Seymour[1])

Despina Papadopoulos's *Flicker Dress* and *Masai Dress* (figure 3) are also works that incorporate technology into fashion. The *Flicker Dress* has LEDs sewn under the fabric, 'each step activates the lights and the white wool becomes a shimmering, reflective surface'[2]. The *Masai Dress* allows us 'to create our own soundtrack as we move about in mysterious ways. With each step, strings of hand-formed silver beads that hang from the collar brush against conductive threads sewn into the dress, generating a series of sounds.' [3]

Papadopoulos's works use technology to subtly enhance the aesthetics of clothing, adding to it a whimsical and understated dimension of interactivity.

---

1 Seymour, pages 113 - 115
2 Seymour, page 115
3 Seymour, page 113

Comparison:

Roughly, the three different sets of precedents can be categorized as artistic, functional, and sensory.

Chalayan's work is artistic. It uses technology to create very original works which are visually stunning. In his works, the main use of technology is to help express his vision. Materials are chosen for their visual effects rather than physical comfort. His dresses are dynamic in a one-way output sense, responding only to an on/off switch.

CuteCircuit's works are functional. Technology is combined with fashion for practical purposes. The technology aspect is hidden, and becomes apparent only in reaction to use. For example, the *M-Dress* appears as a discrete dress under most circumstances, but shows its additional functionality as a mobile phone when the need arises.

Papadapoulos's works are sensory, taking into account the nuanced inputs from the wearer. In the *Flicker Dress*, motion input from the wearer outputs light variations. And in the *Masai Dress*, motion input (captured by a simple yet elegant tilt sensor mechanism made of silver beads and sewn-on silver threads) outputs sound variations. Papadopoulos's works places priority on comfort and wearability. The *Flicker Dress* is made of cashmere, and the *Masai Dress* is made of a cotton and algae blend. In both dresses, technology is not the showcase but rather used to enhance the clothing, adding to it a new responsive dimension.

My Project: The Musical Wearable

Images - sketches

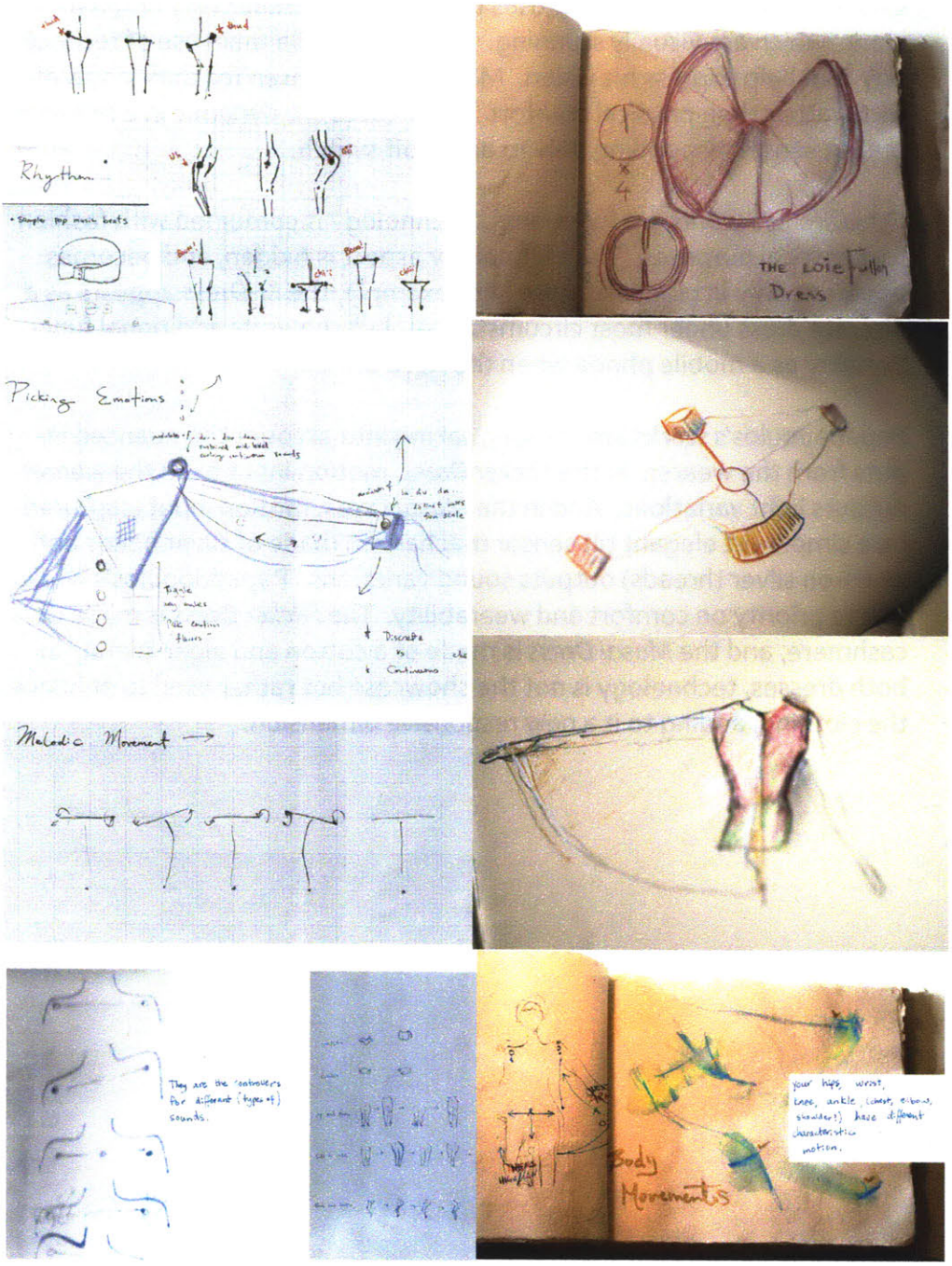
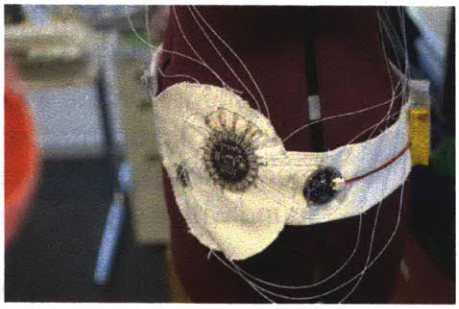
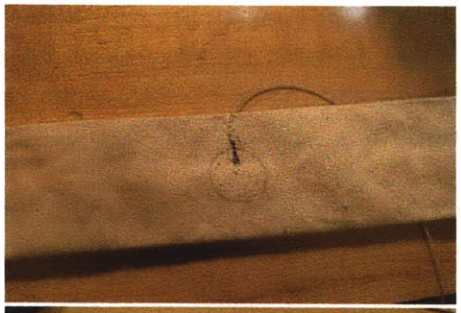


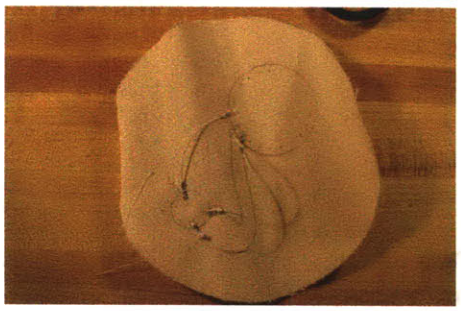
Figure 4. Sketches drawn during the Musical Wearable project.

Images -Prototype

Figure 5. Images of the Musical Wearable prototype during construction and testing.







Concept

The Musical Wearable was conceived as a joyous artifact that when worn, allows the wearer to make music from movements of dance. Since music and movement/dance are broad topics, they can be explored from different approaches. This allows for creation of many kinds of musical garments with different aesthetics that create very different sounds. Responsive musical garments of different sorts have been built many times before. Despina Papadapoulos's *Masai Dress*, discussed in the introduction, is one such example. XS Lab's *Accuphene*[1] and Kobakant's *the Language Game Series*[2] are two other good examples of musical garments.

My approach for making the Musical Wearable uses that fact that when dancing, people move different parts of their bodies in different ways. They may each also have individual rhythms and styles. By mapping different movements to different sounds, the Musical Wearable creates music that responds to and reinforces the act of dance.

The physical system of the Musical Wearable consists of the wearable and a laptop computer that generates sound based on the wearer's movement. On the wearable, three sensors are connected to a Lilypad Arduino microcontroller. The Lilypad is attached to an XBee wireless module that communicates data to the computer, which generates sound with a separate program.

The rest of this section will describe the garment aesthetics, the soundscape, the sensors, the wearable software, and the computer software.

---

1 Appendix C; http://www.xslabs.net/

2 Appendix C; http://www.kobakant.at/?p=27

Garment Aethetics

Originally I planned on embedding the above sensors in a more traditional dress-type garment. I experimented with designs whose drapery and form may encourage or limit certain kinds of movements.

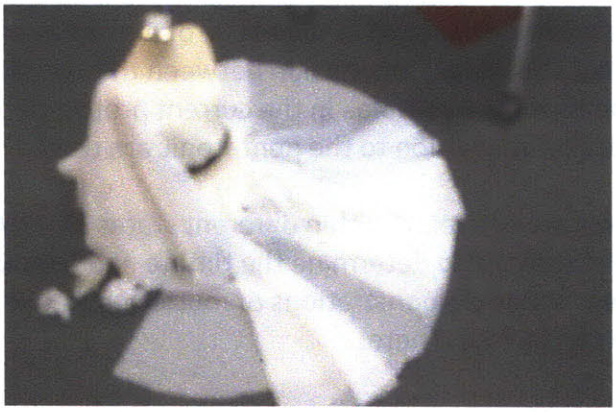


Figure 6. Image of an early garment prototype.

Working with the insulated stainless steel threads from the High-Low Tech Lab[1] inspired me to create a more minimalist wearable, with an "all –wires" aesthetic. The sensors are isolated on fabric islands, interconnected with wires.

The final prototype forms the skeleton of a garment- providing the structure attaching to the body but no covering. Besides the wiring and electronic components, the Musical Wearable contains canvas, velcro, elastic, and zippers. It is meant to be tried-on by different people and is adjustable to fit on different body shapes.

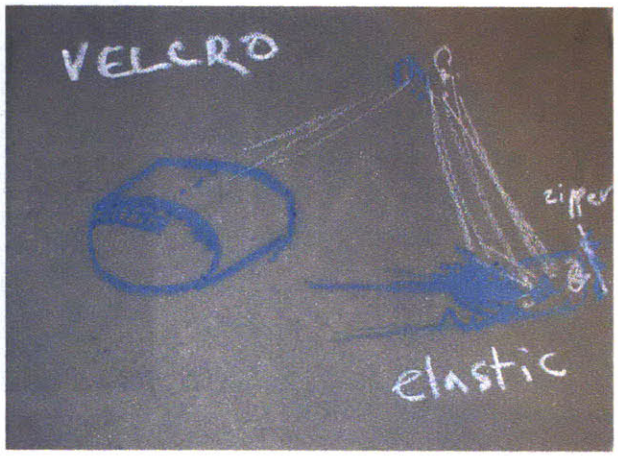


Figure 7. Concept sketch for final prototype.

1 http://hlt.media.mit.edu/

Soundscape

Early on I intended on using notes from the pentatonic scale as audio outputs. The pentatonic scale is unique in that combinations of notes from it always sound harmonic, so the music made while dancing in the dress will always sound pleasant. Additionally, the pentatonic scale is a musical scale traditionally used in the music of many East Asian countries, a fact that can allow for cohesion between the visual and the audio aspects of the Musical Wearable. In the current prototype, several different musical scales in addition to the pentatonic are used.

Different scales are activated within different states of the Musical Wearable. Different states also determine the timbre and the duration of the notes sounded. Video demonstrations of music generated by the Musical Wearable can be found here:

http://uroplog.posterous.com/music-wearable-demo-videos

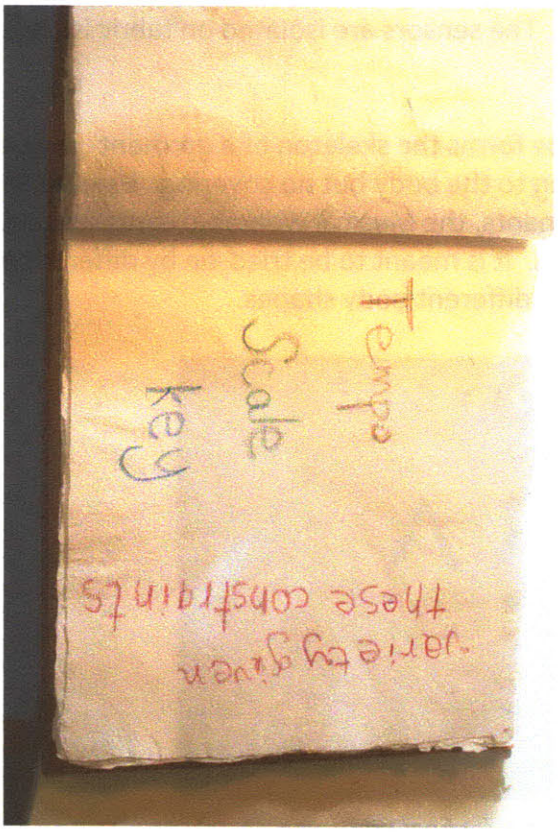


Figure 8. Sketch.

Figure 9. Stills from a video demonstration of the Musical Wearable.

Sensors

Sensor placement influenced the physical shape of the Musical Wearable by focusing on 3 specific body parts – the hip, the shoulder, and the wrist. The idea is to use dance movements from these three body parts to generate different elements of music. As we tend to shake our hips to the beat of the music, hip movements can be mapped to bottom-layer rhythmic sounds. Swaying shoulders can influence mid-layer chord changes, and wrist movements can trigger distinct notes to create the upper-layer melody.

Hip

A 3-axis Lilypad accelerometer is located on the hip.

Figure 10. Close up of the hip accelerometer.

This sensor measures the movement of the wearer's hips. If a constant 5 volts and 0 volts is connected to the '+' and '-' terminals of the accelerometer; a varying voltage will be measured on the 'x', 'y', and 'z' terminals depending on the motion/acceleration experienced at the moment.

The voltage measured from the 'x', 'y', or 'z' axis will be centered at 2.5 volts if it is still and when its axis is in line with gravity, so that it is experiencing a constant 1g of gravitational force. The voltage on each axis increases when it experience positive acceleration (+0.5 volts per 1g of force) and decreases when it experiences negative acceleration (-0.5 volts per -1g of force).

Shoulder

A flower-shaped fabric 6-state tilt sensor based on Hannah Lerner-Wilson's design[1] is located on the shoulder.

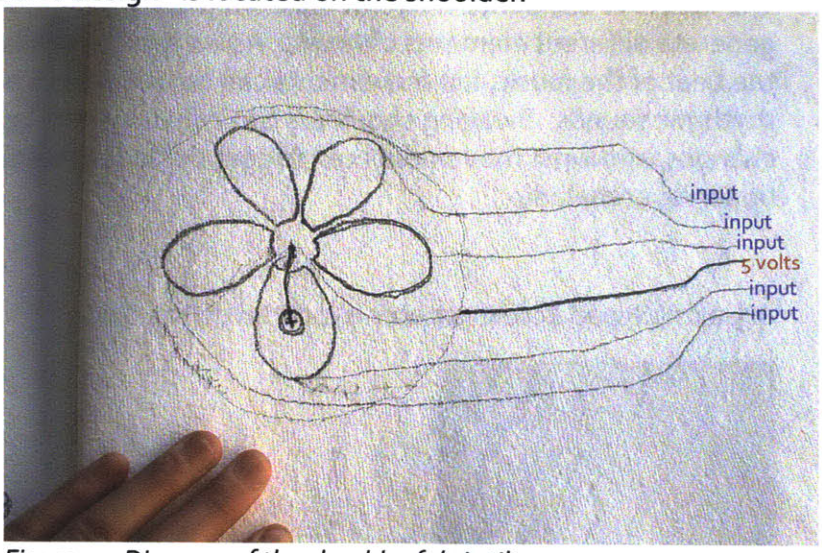


Figure 11. Diagram of the shoulder fabric tilt sensor.

This sensor detects shoulder movement. It is made with conductive fabric petals sewn on a non-conducting fabric backing. In the center of the flower is a stamen made of conductive thread and metal beads that is free to move around.

When the stamen is connected to a 5V output and each petal is connected to a microcontroller input pin, the position of the stamen can be determined by reading the input voltage of the petals. If the stamen is in contact with a particular petal, that petal will read a 5V input. This sensor can be used to roughly determine movement and tilt.

Wrist

A 3-axis Lilypad accelerometer is also located at the wrist.

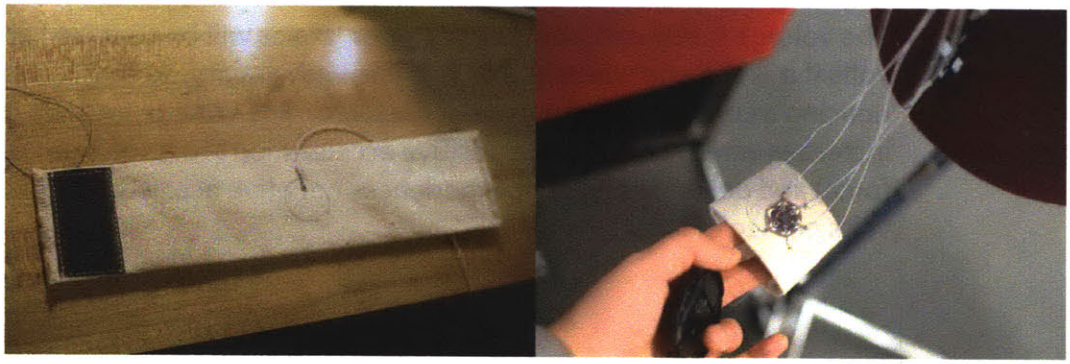


Figure 12. The wrist accelerometer.

1 fabric tilt sensor: http://www.kobakant.at/DIY/?p=201

Wearable Software

This section describes how sensor data is read from the sensors worn on the body and sent to the computer. The sensors are connected to a Lilypad Arduino microcontroller. The Lilypad reads the sensor data and sends the data to a laptop wirelessly with the XBee module.

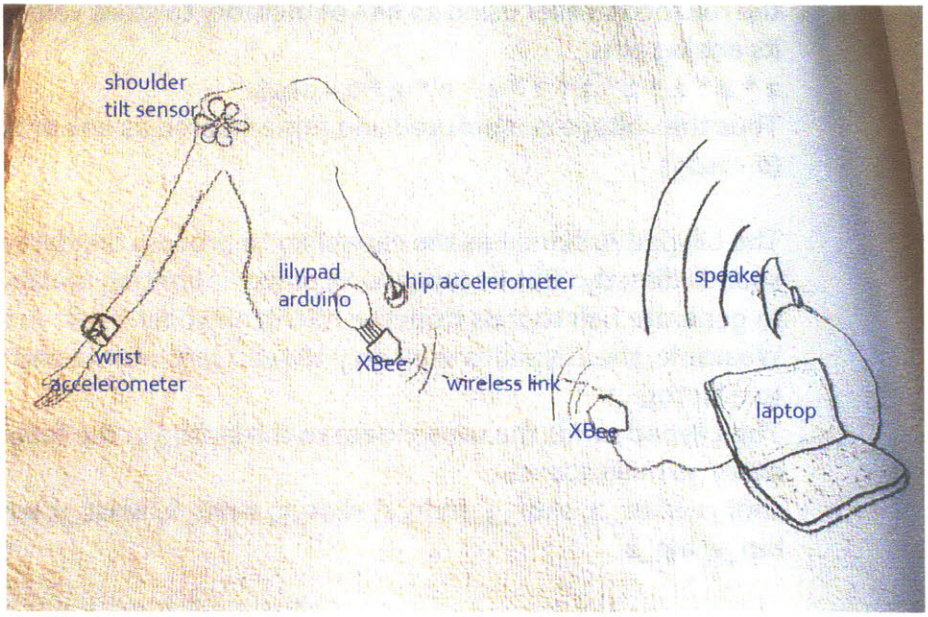


Figure 13. A diagram of the Musical Wearable.

The three sensors (two accelerometers, one fabric tilt sensor) are connected to and powered by a Lilypad Arduino microcontroller. The Lilypad Arduino microcontroller was programmed to read sensor data from the shoulder tilt sensor, the wrist accelerometer, and the hip accelerometer.

The shoulder tilt sensor outputs 5 binary states. Each of its petals will output a '0', unless the stamen is touching it, in which case that petal will measure 5 volts and output a '1'.

The x, y, and z pins of the accelerometer outputs analog values that range between 0 – 5volts. The voltages are read by the microcontroller as integers between 0 – 1023. The 0 – 1023 number range is the result of the microcontroller using 10 bits of memory to store values from each of its analog pins.
2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 = 1024
Thus the voltage is measured and remembered as one of 1024 numbers (0 - 1023).

The Lilypad Arduino has the capability to process the data and to output sounds directly. But its processing power is limited, making it impossible to generate rich sounds from the microcontroller itself. In the Musical Wearable, the Lilypad is used only to read sensor data and to send data to a laptop.
The Lilypad sends the sensor data to the laptop in the following form every 30 milliseconds:
shdr_1, shdr_2, shdr_3, shdr_4, shdr_5, wrist_x, wrist_y, wrist_z, hip_x, hip_y, hip_z

Here is an example stream of sensor data:
0,0,0,1,0, 547, 345, 678, 655, 554, 437

shdr_1, shdr_2, shdr_3, shdr_4, shdr_5 each give the state of each petal of the flower tilt sensor. Each will be a '0' or a '1', depending on whether the stamen is touching the petal or not. Most often one petal will read '1', while all the others will read '0'.
wrist_x, wrist_y, wrist_z, hip_x, hip_y, and hip_z measure the x, y, z streams of accelerometer data from the wrist and hip accelerometers. Each entry will be an integer between 0 – 1023.

XBee radios are used to wirelessly transmit the sensor data from the wearable to the computer where it is mapped to sound.

Computer Software

Sensor data is sent to the laptop from the microcontroller as an 11 entry long string of the form:
shdr_1, shdr_2, shdr_3, shdr_4, shdr_5, wrist_x, wrist_y, wrist_z, hip_x, hip_y, hip_z
Here is an example stream of sensor data:
0,0,0,1,0, 547, 345, 678, 655, 554, 437

These numbers, as they change in time, describe the motion from the hip, wrist, and shoulder.

These numbers are processed using Processing – "an open source programming language and environment for people who want to create images, animations, and interactions."[1]

In Processing, this packet of 11 numbers is divided into 3 groups.

Shoulder

The first group gives the position of the stamen on the flower petals of the shoulder tilt sensor. This toggles the states of the Musical Wearable. Each state determines the key from which notes are drawn and corresponds to a different overall sound effect.
For example:
0,0,0,0,1 means that the stamen is on the left most petal, and when this is the case, the Musical Wearable creates bell-like, high notes as the wearer dances.
0,0,1,0,0 means the stamen is on the middle petal, and when this is the case, mellow tones are produced from dance.

---

1 http://processing.org/

Wrist

Motion from the wrist triggers the melodic notes and tones.
The second group of numbers – the accelerometer readings from the wrist, comes in the form of 3 streams of changing analog values.

Each of the three streams is digitalized to produce only '0's or '1's, instead of the original numbers ranging from 0 – 1023. Using 512 as a threshold, numbers below 512 are converted to a '0' and numbers above 512 are converted to a '1'. This step converts 3 streams of analog data to a more limited range. The readings from the 'x', 'y', and 'z' axis will be either '0' or '1', giving 8 possible different configurations:

0,0,0
0,0,1
0,1,0
0,1,1
1,0,0
1,0,1
1,1,0
1,1,1

Each of these 8 configurations can be imagined as occupying one of the 8 quadrants in 3D space, and each quadrant is associated with different sounds. By moving and rotating the wrist, the wearer induces changes between these quadrants.
Whenever a transition between quadrants occurs, if a change has not occurred very shortly before, the sound associated with the new quadrant is played.

Hip

The third group of numbers is the three streams of data from the hip accelerometer. In generating sounds, only one stream of data from the z-axis is used. The particular orientation and placement of the hip accelerometer in the Musical Wearable makes its z-axis the most sensitive to hip-shaking motions.

The data from the z-axis (a changing number between 0 – 1023) is passed through a low-pass filter. Where raw data may be spiky, changing sharply from a low number to a high number and back down, the low-pass filter smoothes the data, making changes more gradual and smooth by creating an exponential-weighted moving average of the current data and its history.[1]

When hips are shaken, the ringing of bells is sounded. This is also implemented with the threshold – trigger idea. Using a threshold of 430 and the low-pass filtered data, whenever a transition in the data occurs from below the threshold to above the threshold, the sound of ringing bells is triggered. This setup prevents the bells from ringing too frequently. The data manipulation and sound generation on the laptop is accomplished through code written in the Processing programming language, using Processing's built in Minim audio library. Code in appendix D[2].

All of the sounds that the Musical Wearable makes are sampled sounds. Sampled sounds are existing sound files which the processing program can access and play. Sampled sounds are pre-recorded, as opposed to purely computer generated sounds, which are made in real-time by combining and manipulating 'oscillators'.

The advantage of using sampled sounds is that complex and rich notes can be captured and played. Oscillators tend to create cold, computer-like sounds.

The notes associated with the wrist are sound files made with an electric guitar and Audacity, a digital audio editor. The bell sounds associated with the wrist were found on the internet, and also edited with Audacity.

---

1 Reliable Data Transport Protocols
http://web.mit.edu/6.02/www/f2010/handouts/lectures/L20.pdf
page 8, Exponential Weighted Moving Average

2 Appendix D. Codes for Lilypad and Processing

Conclusion

My motivation for creating the Musical Wearable was to make and learn about music, electronics and software through the familiar medium of sewing and garment making. Working on the Musical Wearable, I learned how to work with sensor data, to create a serial link between a microcontroller and an external application, to use XBee radios to create a wireless link, and to program sounds and visualizations for sensor data.

There are 3 major areas to work on to further develop the Musical Wearable project. Currently, the Musical Wearable is less of a garment and more of a wearable contraption. To develop the project further as a fashion piece, it would be useful to try incorporating the hardware again into a more traditional garment form.

The design of the garment can incorporate the conductive threads and Lilypad sensors and microcontroller to enhance the overall aesthetic. It would be interesting to use the conductive threads and sensors visibly as design elements of the garment itself rather than hiding them in the fabric. For example, the garment can have openings in the fabric where the conductive threads and sensors are highlighted with a background of bare-skin.

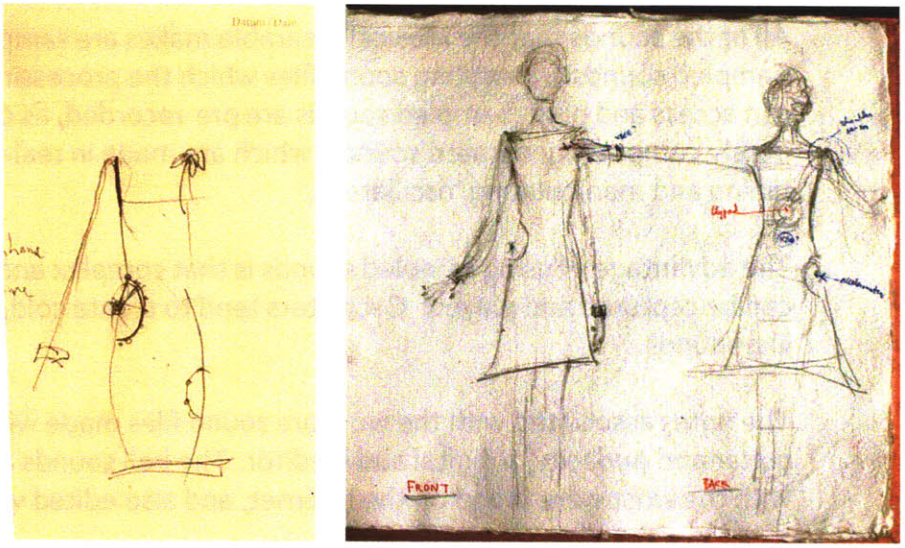


Figure 14. Garment prototype designs.

Another area to work on is to further develop the musical wearable as a musical instrument that you play with your body.

Currently arm, shoulder and hip movements loosely control the music and sounds played. I would like to incorporate another, more finely controllable method for sound making into the musical garment.

A pair of gloves/finger sensors would be a good way to implement this. For example, pinching different fingers of the left hand can set different chords, and while the left hand configuration is maintained, pinching the fingers on the right hand in different ways play different notes from the particular chord. To implement this, it would be helpful to learn to make several Xbees talk back and forth to each other.

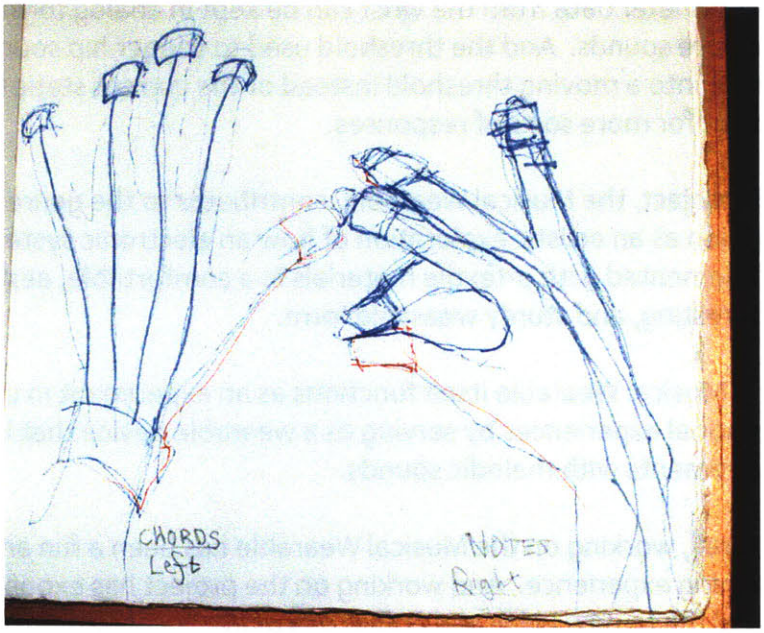


Figure 15. Finger sensors sketch.

And lastly, the music programming can be further worked on. Currently, the sound output device is the laptop computer. The sounds are produced a distance away from the body. The fashion-movement-sound link will be stronger if sound output devices are integrated into the clothing. One possible way to achieve this is to use a mobile phone to process sensor data and run the sound generating application.

It would be nice to incorporate ways to control more aspects of the sounds produced, ways to control the volume, duration, and pitch of the notes played.
For example, new code can be added to relate the level of total sensor readings to the volume. Perhaps frequency of changes can relate to the duration of notes and certain specific movements can alter pitches.

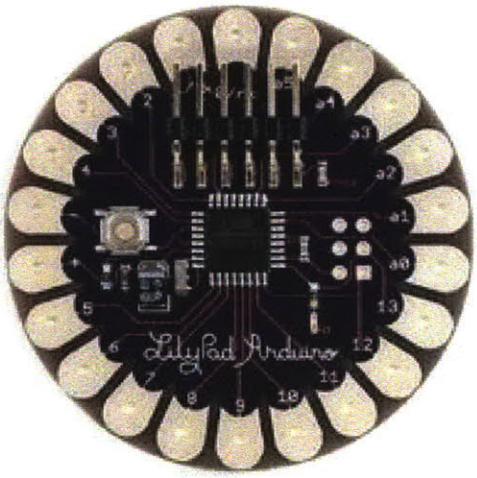
Current elements of the code can also be improved. For example the accelerometer data from the wrist can be kept in analog form, and mapped to more sounds. And the threshold used to trigger hip sounds can be made into a moving threshold instead of the current static threshold to adjust for more sorts of responses.

My project, the Musical Wearable, contributes to the genre of electronic fashion as an artistic exploration of how an electronic system can be implemented with e-textile materials in a comfortable, aesthetically interesting, and sturdy wearable form.

The Musical Wearable itself functions as an experiment in creating new sensorial experiences by serving as a wearable device that links dance movements with melodic sounds.

Overall, working on the Musical Wearable has been a fun and challenging learning experience. And working on the project has exposed and interested me to new ideas and areas to explore.

Appendix A - Lilypad Arduino



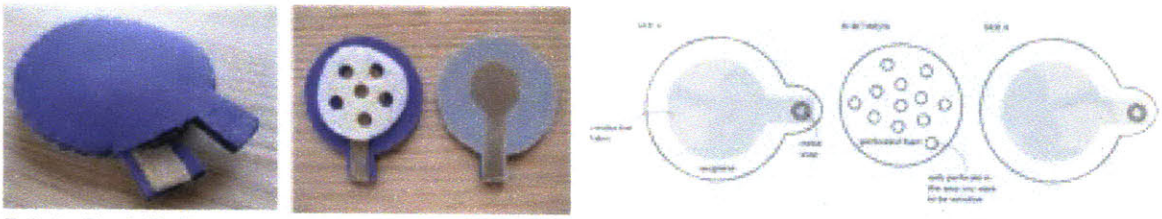
Summary:
The LilyPad Arduino is a microcontroller board designed for wearables and e-textiles. The Lily- Pad Arduino is a circle, approximately 50mm (2") in diameter. The board itself is .8mm (1/32") thick (approximately 3mm (1/8") where electronics are attached). It can be sewn to fabric and similarly mounted power supplies, sensors and actuators with conductive thread. The board is based on the ATmega168V (the low-power version of the ATmega168) or the ATmega328. The LilyPad Arduino was designed and developed by Leah Buechley and SparkFun Electronics. The LilyPad Arduino can be programmed with the Arduino software.

Microcontroller	ATmega168V or ATmega328V
Operating Voltage	2.7-5.5 V Input Voltage2.7-5.5 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA

Flash Memory 16 KB (of which 2 KB used by bootloader)
SRAM 1 KB EEPROM
512 bytes Clock Speed 8 MHz

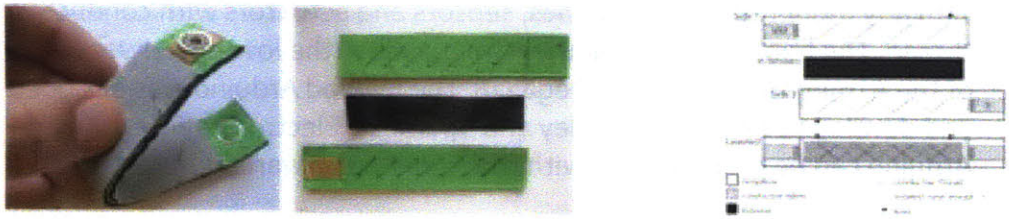
Information from www.arduino.cc

Appendix B - Textile Sensors

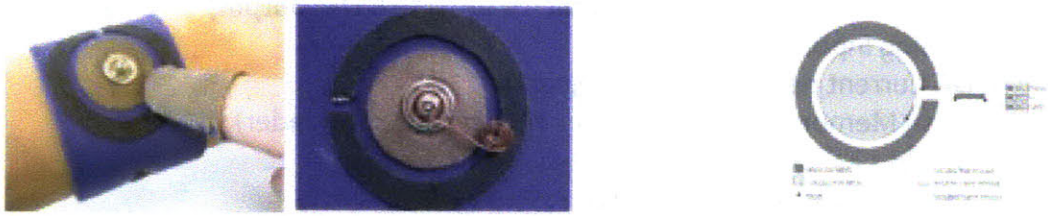


Fabric Push Button

Fabric Pressure Sensor



Fabric Bend Sensor



Fabric Potentiometer

Images from Hannah Perner-Wilson and Leah Buechley

Appendix C - Musical Garments

Accouphene
By Vincent Leclerc, Joanna Berzowska

"The Accouphene tuxedo is decorated with 13 soft speakers, cre- ated by embroidering decorative coils of highly conductive yarn on the front of the jacket. A central circuit sends pulses of energy through the coils. Sounds are generated when then sleeve of Ac- couphene, which contains a stitched magnet, is moved over the coils. Accou- phene creates a 3D sonic environment around the human body that can be acti- vated and modulated through hand move- ment and the twisting and compression of the cloth. When powered, the speak- ers generate a weak magnetic field that reacts to a strong magnet positioned in close proximity to the fabric. The magnet's strength and its distance from the embroi- dered coil deter- mine the amplitude of the Sound." -Sabine Seymour

Language Game Series
by Mika Satomi and Hannah Perner-Wilson

"A series of performances that use custom- ized editions of the Puppeteer motion-cap- ture suit to communicate con- cepts con- cerning the use of motion, gesture and the spo- ken word as forms of communication. In this first edition of the Language Game, the performer, wearing a custom designed soft sensor suit, puppeteers Le- mur's sur- rounding robotic musical instruments. As the per- former begins to experiment with the system's potential, her dance becomes both action and reaction." - KOBAKANT

Appendix C - Code

Lilypad Arduino Code

```
const int shoulder5v = 6;
const int shoulder1 = 7;
const int shoulder2 = 8;
const int shoulder3 = 9;
const int shoulder4 = 10;
const int shoulder5 = 11;
const int shoulderov = 2;

const int hipAcc5v = 13;
const int hipAccov = 12;
const int hipX = 14;
const int hipY = 15;
const int hipZ = 16;

const int wrist5v = 5;
const int wristov = 4;
const int wristX = A3;
const int wristY = A4;
const int wristZ = A5;

void setup(){
 Serial.begin(9600);
 pinMode(shoulder5v, OUTPUT);
 pinMode(shoulderov, OUTPUT);

 pinMode(hipAcc5v, OUTPUT);
 pinMode(hipAccov, OUTPUT);

 pinMode(wrist5v, OUTPUT);
 pinMode(wristov, OUTPUT);

 digitalWrite(shoulder5v, HIGH);
 digitalWrite(hipAcc5v, HIGH);
 digitalWrite(wrist5v, HIGH);

 digitalWrite(shoulderov, LOW);
 digitalWrite(hipAccov, LOW);
 digitalWrite(wristov, LOW);

 pinMode(shoulder1, INPUT);
 pinMode(shoulder2, INPUT);
 pinMode(shoulder3, INPUT);
 pinMode(shoulder4, INPUT);
```

```
 pinMode(shoulder5, INPUT);
 }

void loop(){
   int shoulder_1 = digitalRead(shoulder1);
   int shoulder_2 = digitalRead(shoulder2);
   int shoulder_3 = digitalRead(shoulder3);
   int shoulder_4 = digitalRead(shoulder4);
   int shoulder_5 = digitalRead(shoulder5);

   int wrist_X = analogRead(wristX);
   int wrist_Y = analogRead(wristY);
   int wrist_Z = analogRead(wristZ);
   int hip_X = analogRead(hipX);
   int hip_Y = analogRead(hipY);
   int hip_Z = analogRead(hipZ);

   Serial.print(shoulder_1);
   Serial.print(',');
   Serial.print(shoulder_2);
   Serial.print(',');
   Serial.print(shoulder_3);
   Serial.print(',');
   Serial.print(shoulder_4);
   Serial.print(',');
   Serial.print(shoulder_5);
   Serial.print(',');

Serial.print(wrist_X);
   Serial.print(',');
   Serial.print(wrist_Y);
   Serial.print(',');
   Serial.print(wrist_Z);
   Serial.print(',');

   Serial.print(hip_X);
   Serial.print(',');
   Serial.print(hip_Y);
   Serial.print(',');
   Serial.println(hip_Z);

   delay(30);
}
```

Processing Code for the Musical Wearable

```
import processing.serial.*;
import ddf.minim.*;
Minim minim = new Minim( this );

AudioSample E;
AudioSample Fs;
AudioSample Gs;
AudioSample A;
AudioSample B;
AudioSample Cs;
AudioSample Ds;
AudioSample E2;

AudioSample a_1;
AudioSample a_2;
AudioSample a_3;
AudioSample a_4;
AudioSample a_5;
AudioSample a_6;
AudioSample a_7;
AudioSample a_8;

AudioSample b_1;
AudioSample b_2;
AudioSample b_3;
AudioSample b_4;
AudioSample b_5;
AudioSample b_6;
AudioSample b_7;
AudioSample b_8;

AudioSample c_1;
AudioSample c_2;
AudioSample c_3;
AudioSample c_4;
AudioSample c_5;
AudioSample c_6;

AudioSample d_1;
AudioSample d_2;
AudioSample d_3;
AudioSample d_4;
AudioSample d_5;
AudioSample d_6;
AudioSample d_7;
```

```
AudioSample d_8;

AudioSample longBell;
AudioSample shortBell;
AudioSample multiBell;
AudioSample subWoofer;


/////global variables
int midPoint = 510;
int rawX, rawY, rawZ;
int X, Y, Z;
int previousMetric;
int currentMetric;
long lastPlay;

int[] previousState = {-1, -1, -1};
int[] currentState = {-1, -1, -1};

//shoulderstuff
int[] shoulderArray = {0,0,0,0,0};
int shoulderPosition;

//hipstuff
int threshold = 430;
int state = 0;
int[] lpfHip = {0,0,0};

int linefeed = 10;
Serial myPort;


void setup(){

 println(Serial.list());
 myPort = new Serial(this, Serial.list()[1], 9600);
 E = minim.loadSample("soundBits/E164_81.mp3");
 Fs = minim.loadSample("soundBits/F#185_00.mp3");
 Gs = minim.loadSample("soundBits/G#207_65.mp3");
 A = minim.loadSample("soundBits/A220_00.mp3");
 B = minim.loadSample("soundBits/B246_94.mp3");
 Cs = minim.loadSample("soundBits/C#277_18.mp3");
 Ds = minim.loadSample("soundBits/D#311_13.mp3");
 E2 = minim.loadSample("soundBits/E329_63.mp3");

 a_1 = minim.loadSample("soundBits/a_1.mp3");
 a_2 = minim.loadSample("soundBits/a_2.mp3");
 a_3 = minim.loadSample("soundBits/a_3.mp3");
```

```
  a_4 = minim.loadSample("soundBits/a_4.mp3");
  a_5 = minim.loadSample("soundBits/a_5.mp3");
  a_6 = minim.loadSample("soundBits/a_6.mp3");
  a_7 = minim.loadSample("soundBits/a_7.mp3");
  a_8 = minim.loadSample("soundBits/a_8.mp3");

  b_1 = minim.loadSample("soundBits/b_1.mp3");
  b_2 = minim.loadSample("soundBits/b_2.mp3");
  b_3 = minim.loadSample("soundBits/b_3.mp3");
  b_4 = minim.loadSample("soundBits/b_4.mp3");
  b_5 = minim.loadSample("soundBits/b_5.mp3");
  b_6 = minim.loadSample("soundBits/b_6.mp3");
  b_7 = minim.loadSample("soundBits/b_7.mp3");
  b_8 = minim.loadSample("soundBits/b_8.mp3");

  c_1 = minim.loadSample("soundBits/c_1.mp3");
  c_2 = minim.loadSample("soundBits/c_2.mp3");
  c_3 = minim.loadSample("soundBits/c_3.mp3");
  c_4 = minim.loadSample("soundBits/c_4.mp3");
  c_5 = minim.loadSample("soundBits/c_5.mp3");
  c_6 = minim.loadSample("soundBits/c_6.mp3");

  d_1 = minim.loadSample("soundBits/d_1.mp3");
  d_2 = minim.loadSample("soundBits/d_2.mp3");
  d_3 = minim.loadSample("soundBits/d_3.mp3");
  d_4 = minim.loadSample("soundBits/d_4.mp3");
  d_5 = minim.loadSample("soundBits/d_5.mp3");
  d_6 = minim.loadSample("soundBits/d_6.mp3");
  d_7 = minim.loadSample("soundBits/d_7.mp3");
  d_8 = minim.loadSample("soundBits/d_8.mp3");

  shortBell = minim.loadSample("soundBits/shortBell.mp3");
  longBell = minim.loadSample("soundBits/longBell.mp3");
  multiBell = minim.loadSample("soundBits/multiBell.mp3");
  subWoofer = minim.loadSample("soundBits/subWoofer.mp3");

  myPort.bufferUntil(linefeed);
}

void draw(){
  background(204);
  if (millis() - lastPlay > 300){
  if (currentMetric >= 0){
    if (previousMetric != currentMetric){
    println("Draw Sound");
    println("shoulderPos:" + shoulderPosition);

    play(shoulderPosition, currentMetric);
```

```
  lastPlay = millis();
  }
 }
 }
 println("lpf: " +lpfHip[2]);
  if (lpfHip[2]< threshold){
// println(1);
  if (state == 0){
  //randomTrigger(int(random(3)));
   multiBell.trigger();
   state = 1;
  }
 }

 else if (lpfHip[2]>= threshold){
 //println(0);
  if (state == 1){
  state = 0;
  }
 }
}

void serialEvent(Serial myPort) {
 String data = myPort.readStringUntil(linefeed);
 data = trim(data);
 if (data != null){

  int datalist[] = int(split(data, ','));  ///this is the incoming string of data!!!

  arrayCopy(datalist, 0, shoulderArray, 0, 5);
  shoulderPosition = shoulderArray[0] + 2*shoulderArray[1] + 3*shoulderAr-
ray[2] + 4*shoulderArray[3] + 5*shoulderArray[4];


  for (int i = 0; i < lpfHip.length; i++){
  lpfHip[i] = lpf(lpfHip[i], datalist[i + 8], .15);
  }


  rawX = datalist[5]; rawY = datalist[6]; rawZ = datalist[7];
  X = lpf(X, datalist[5], .50);
  Y = lpf(Y, datalist[6], .50);
  Z = lpf(Z, datalist[7], .50);


previousState[0] = currentState[0];
previousState[1] = currentState[1];
previousState[2] = currentState[2];
```

```
currentState[0] = X/midPoint;
currentState[1] = Y/midPoint;
currentState[2] = Z/midPoint;


previousMetric = 4* previousState[0] + 2*previousState[1] + previousState[2];
currentMetric = 4* currentState[0] + 2*currentState[1] + currentState[2];
 }
}


//////////functions
int lpf(float output, float input, float _alpha){
 output = _alpha* input + (1 - _alpha)*output;
  return int(output);
}
void storeDataPoint(int[] array1, int reading)
{
 for (int i = array1.length -1; i >0; i--){
  array1[i] = array1[i-1];
 }
 array1[0] = reading;
}

void graph(int[] array1){
 for (int i =1; i < array1.length; i++){
  line(i, (width - array1[i]), i-1, (width - array1[i-1]));
 }
}


///////
void play(int shoulderPos, int metric){
 if (shoulderPos == 3){
  if (metric == 0){
   a_1.trigger();
  }
  if (metric == 1){
   a_2.trigger();
  }
  if (metric == 2){
   a_3.trigger();
  }
  if (metric == 3){
   a_4.trigger();
```

```
 }
 if (metric == 4){
  a_5.trigger();
 }
 if (metric == 5){
  a_6.trigger();
 }
 if (metric == 6){
  a_7.trigger();
 }
 if (metric == 7){
  a_8.trigger();
 }
}

if (shoulderPos == 2){
 if (metric == 0){
   b_1.trigger();
 }
 if (metric == 1){
   b_2.trigger();
 }
 if (metric == 2){
   b_3.trigger();
 }
 if (metric == 3){
   b_4.trigger();
 }
 if (metric == 4){
   b_5.trigger();
 }
 if (metric == 5){
   b_6.trigger();
 }
 if (metric == 6){
   b_7.trigger();
 }
 if (metric == 7){
   b_8.trigger();
 }
}

 if (shoulderPos == 1){
  if (metric == 0){
    c_1.trigger();
  }
  if (metric == 1){
    c_2.trigger();
```

```
}
if (metric == 2){
 c_3.trigger();
}
if (metric == 3){
 c_4.trigger();
}
if (metric == 4){
 c_5.trigger();
}
if (metric == 5){
 c_6.trigger();
}
if (metric == 6){
 c_1.trigger();
}
if (metric == 7){
 c_2.trigger();
}
}

if (shoulderPos == 4){
 if (metric == 0){
  d_1.trigger();
 }
 if (metric == 1){
  d_2.trigger();
 }
 if (metric == 2){
  d_3.trigger();
 }
 if (metric == 3){
  d_4.trigger();
 }
 if (metric == 4){
  d_5.trigger();
 }
 if (metric == 5){
  d_6.trigger();
 }
 if (metric == 6){
  d_7.trigger();
 }
 if (metric == 7){
  d_8.trigger();
 }
}
```

```
if (shoulderPos >= 5){
        if (metric == 0){
      E.trigger();
    }

    if (metric == 1){
      Fs.trigger();
    }

    if (metric == 2){
      Gs.trigger();
    }

    if (metric == 3){
      A.trigger();
    }

    if (metric == 4){
      B.trigger();
    }

    if (metric == 5){
      Cs.trigger();
    }

    if (metric == 6){
      Ds.trigger();
    }

    if (metric == 7){
      E2.trigger();
    }
}


if (shoulderPos == 0){
    if (metric == 0){
      E.trigger();
    }

    if (metric == 1){
      Fs.trigger();
    }

    if (metric == 2){
      Gs.trigger();
    }
```

```
      if (metric == 3){
       A.trigger();
      }

      if (metric == 4){
       B.trigger();
      }

      if (metric == 5){
       Cs.trigger();
      }

      if (metric == 6){
       Ds.trigger();
      }

      if (metric == 7){
       E2.trigger();
      }
  }
}

//////

void randomTrigger(int number){
if (number == 0){
 longBell.trigger();
}
if (number == 1){
 shortBell.trigger();
}
if (number == 2){
 multiBell.trigger();
}
}


///stop method
void stop()
{
 // always close Minim audio classes when you are done with them
 E.close();
 Fs.close();
 Gs.close();
 A.close();
 B.close();
 Cs.close();
 Ds.close();
```

```
E2.close();

a_1.close();
a_2.close();
a_3.close();
a_4.close();
a_5.close();
a_6.close();
a_7.close();
a_8.close();

b_1.close();
b_2.close();
b_3.close();
b_4.close();
b_5.close();
b_6.close();
b_7.close();
b_8.close();

c_1.close();
c_2.close();
c_3.close();
c_4.close();
c_5.close();
c_5.close();

d_1.close();
d_2.close();
d_3.close();
d_4.close();
d_5.close();
d_6.close();
d_7.close();
d_8.close();

minim.stop();

super.stop();
}
```

Processing Code for a Sensor Visualizer

```
import processing.serial.*;

/////global variables
int linefeed = 10;
Serial myPort;
int[] shoulderArray = {0,0,0,0,0};

////these are normalized values
float rawWristX, rawWristY, rawWristZ;
float wristX, wristY, wristZ;
float[] rawWristXarray, rawWristYarray, rawWristZarray;
float[] wristXarray, wristYarray, wristZarray;
float rawHipX, rawHipY, rawHipZ;
float hipX, hipY, hipZ;
float[] rawHipXarray, rawHipYarray, rawHipZarray;
float[] hipXarray, hipYarray, hipZarray;


void setup(){
 size(250,800);

///initialize lists
rawWristXarray = new float[width];
rawWristYarray = new float[width];
rawWristZarray = new float[width];

wristXarray = new float[width];
wristYarray = new float[width];
wristZarray = new float[width];

rawHipXarray = new float[width];
rawHipYarray = new float[width];
rawHipZarray = new float[width];

hipXarray = new float[width];
hipYarray = new float[width];
hipZarray = new float[width];


 println(Serial.list());
 myPort = new Serial(this, Serial.list()[1], 9600);

 myPort.bufferUntil(linefeed);
}
```

```
void draw(){
 background(204);
 noFill();
 dividers();
 shoulder();
//
 graph(rawWristXarray, 110, 80);
 graph(rawWristYarray, 110, 190);
 graph(rawWristZarray, 110, 300);
 graph(rawHipXarray, 110, 410);
 graph(rawHipYarray, 110, 520);
 graph(rawHipZarray, 110, 630);

 stroke(0,255,0);
 graph(wristXarray, 110, 80);
 graph(wristYarray, 110, 190);
 graph(wristZarray, 110, 300);
 graph(hipXarray, 110, 410);
 graph(hipYarray, 110, 520);
 graph(hipZarray, 110, 630);

}

void serialEvent(Serial myPort) {
 String data = myPort.readStringUntil(linefeed);
 data = trim(data);
 if (data != null){

  int datalist[] = int(split(data, ','));  ///this is the incoming string of data!!!
  arrayCopy(datalist, 0, shoulderArray, 0, 5);

  rawWristX = norm(datalist[5], 0, 1023); storeDataPoint(rawWristXarray, raw-
WristX);
  wristX = lpf(wristX, rawWristX, .15);  storeDataPoint(wristXarray, wristX);

  rawWristY = norm(datalist[6], 0, 1023); storeDataPoint(rawWristYarray, raw-
WristY);
  wristY = lpf(wristY, rawWristY, .15);  storeDataPoint(wristYarray, wristY);

  rawWristZ = norm(datalist[7], 0, 1023); storeDataPoint(rawWristZarray, raw-
WristZ);
  wristZ = lpf(wristZ, rawWristZ, .15);  storeDataPoint(wristZarray, wristZ);

  rawHipX = norm(datalist[8], 0, 1023); storeDataPoint(rawHipXarray,
rawHipX);
  hipX = lpf(hipX, rawHipX, .15);  storeDataPoint(hipXarray, hipX);
  println("rawx: " + rawHipX);
//  println(wristX);
```

```
  rawHipY = norm(datalist[9], 0, 1023); storeDataPoint(rawHipYarray,
rawHipY);
  hipY = lpf(hipY, rawHipY, .15);  storeDataPoint(hipYarray, hipY);
  println("rawY: " + rawHipY);
  rawHipZ = norm(datalist[10], 0, 1023); storeDataPoint(rawHipZarray,
rawHipZ);
  hipZ = lpf(hipZ, rawHipZ, .15);  storeDataPoint(hipZarray, hipZ);
  println("rawZ: " + rawHipZ);

/////
///// printing out serial data
/////

///printing the shoulder sensor states (0 or 1)
  for (int sensorNum = 0; sensorNum < 5; sensorNum++){
   print("shoulder" + sensorNum + ": " + datalist[sensorNum] + "\t");
  }
  println();  // space

  ///printing the wrist accelerometer readings
  for (int sensorNum = 5; sensorNum < 8; sensorNum++){
   print("wrist " + char(sensorNum + 83) + ": " + datalist[sensorNum] + "\t");
  }
  println();  // space

  ///printing the hip accelerometer readings
  for (int sensorNum = 8; sensorNum < 11; sensorNum++){
   print(" hip " + char(sensorNum + 80) + ": " + datalist[sensorNum] + "\t");
  }
  println();
  println();
/////
///// printing out data
/////

 }
}




//////////functions
float lpf(float output, float input, float _alpha){
 output = _alpha* input + (1 - _alpha)*output;
 return output;
}
void storeDataPoint(float[] array1, float reading)
```

```
{
 for (int i = array1.length -1; i >0; i--){
  array1[i] = array1[i-1];
 }
 array1[0] = reading;
}

void graph(float[] array1, int scaling, int offset){
 for (int i =1; i < array1.length; i++){
  line(i, scaling * array1[i] + offset, i-1, scaling*array1[i-1] + offset);
 }
}

void dividers(){
 stroke(0,0,0);
 rect(10, 25, 30, 30);
 rect(60, 25, 30, 30);
 rect(110, 25, 30, 30);
 rect(160, 25, 30, 30);
 rect(210, 25, 30, 30);
 line(0, 80, 250, 80);  //line1
 line(0, 110+80, 250, 110+80);  //line2
 line(0, 110*2+80, 250, 110*2+80);  //line3
 line(0, 110*3+80, 250, 110*3+80);  //line4
 line(0, 110*4+80, 250, 110*4+80);  //line5
 line(0, 110*5+80, 250, 110*5+80);  //line6
}

void shoulder(){
if(shoulderArray[0] == 1){
 ellipse(25, 40, 30, 30);
}
if(shoulderArray[1] == 1){
 ellipse(75, 40, 30, 30);
}
if(shoulderArray[2] == 1){
 ellipse(125, 40, 30, 30);
}
if(shoulderArray[3] == 1){
 ellipse(175, 40, 30, 30);
}
if(shoulderArray[4] == 1){
 ellipse(225, 40, 30, 30);
}

}
```

List of Figures

Bibliography

Fashion. (n.d.). In Wikipedia. Retrieved January 17, 2011, from http://en.wikipedia.org/wiki/Fashion

High-Low Tech. Retrieved January 17, 2011, from http://hlt.media.mit.edu/

KobaKant. Retrieved January 17, 2011, from http://www.kobakant.at/?p=27

Lilypad Arduino. In arduino.cc. Retrieved January 17, 2011, from http://www.arduino.cc/en/Main/ArduinoBoardLilyPad

MacLuhan, Marshall: *Understanding Media – The Extension of Man*. MIT Press, Cambridge, MA, USA 1995.

Perner-Wilson, Hannah: *Handcrafting Textile Sensors from Scratch.* Retrieved January 17, 2011, from http://plusea.at/downloads/HandcraftingSensorsWS_sm.pdf

Processing. Retrieved January 17, 2011, from http://processing.org/

Reliable Data Transport Protocols. Retrieved January 17, 2011, from http://web.mit.edu/6.02/www/f2010/handouts/lectures/L20.pdf

Seymour, Sabine: *Fashionable Technology – The Intersection of Design, Fashion, Science, and Technology.* SpringerWien, New York, NY, USA 2009

XSlabs. Retrieved January 17, 2011, from http://www.xslabs.net/