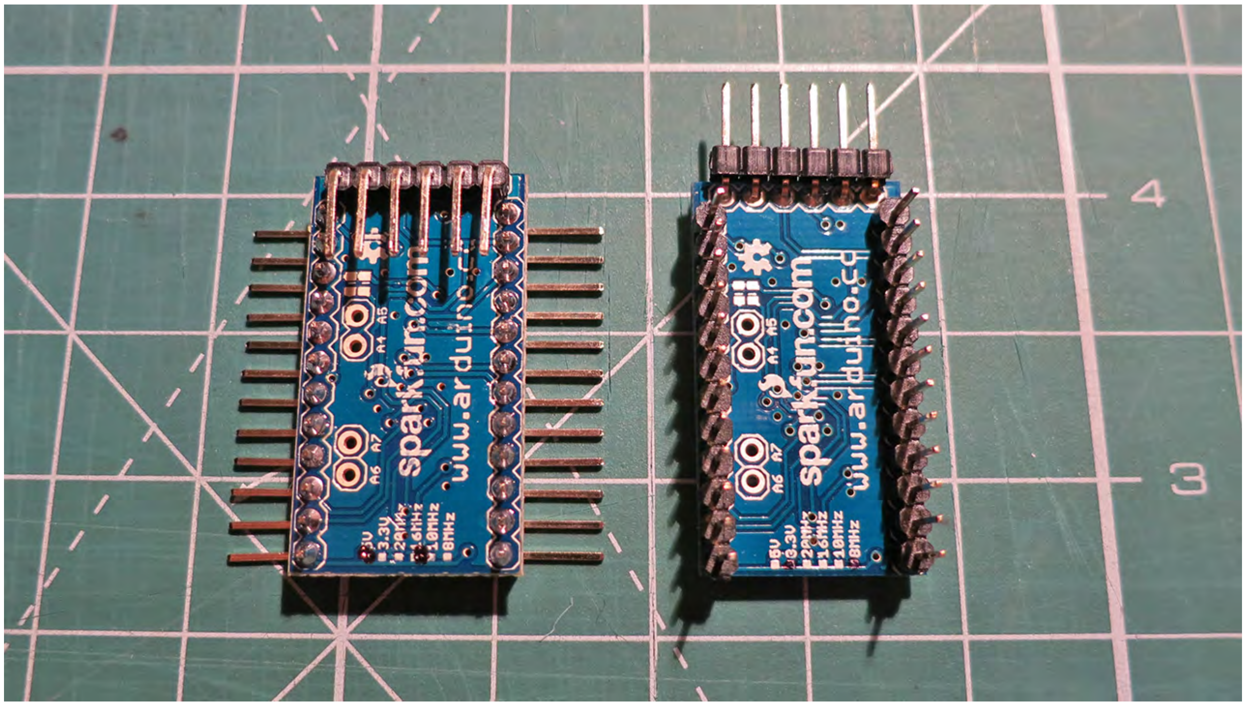
Fantasia

“... and the illumination is in fact performance” –Patricia Barber

Fantasia is a wearable technology project that translates a physical performance into a two dimensional time-based graphic narrative. Fantasia registers a performer’s hand movements and gestures (by means of a three axis accelerometer integrated into a wrist band), and sends the data wirelessly to the Processing application for further computation.

Circuitry

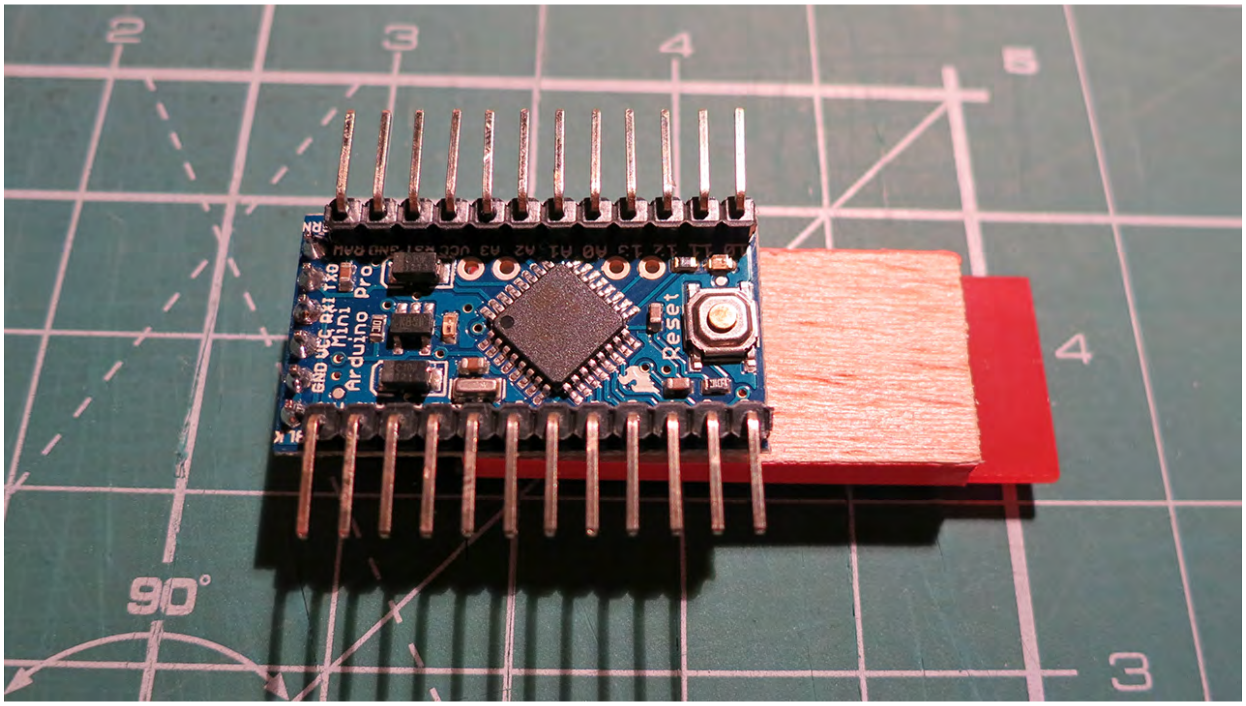
The heart of the circuit is an Arduino Pro Mini 328 (Sparkfun #DEV-11113). Initially I had planned to use a LilyPad SimpleSnap (Sparkfun #DEV-10941) due to its ability to provide power in a small package, but during my prototyping process I found out the this model is not capable of powering an external shield via its FTDI connector. For the wireless connection I used a Bluetooth Mate Silver (Sparkfun #WRL-10393). Adding the shield to the Arduino, I needed to find a way to package the whole circuit in a smaller space (when testing the first prototype, I realized that if the performer bends her wrist upward, it would put a lot of pressure/stress on her wrist/the circuit). Images below demonstrates my alternate soldering technique/solution for this challenge.

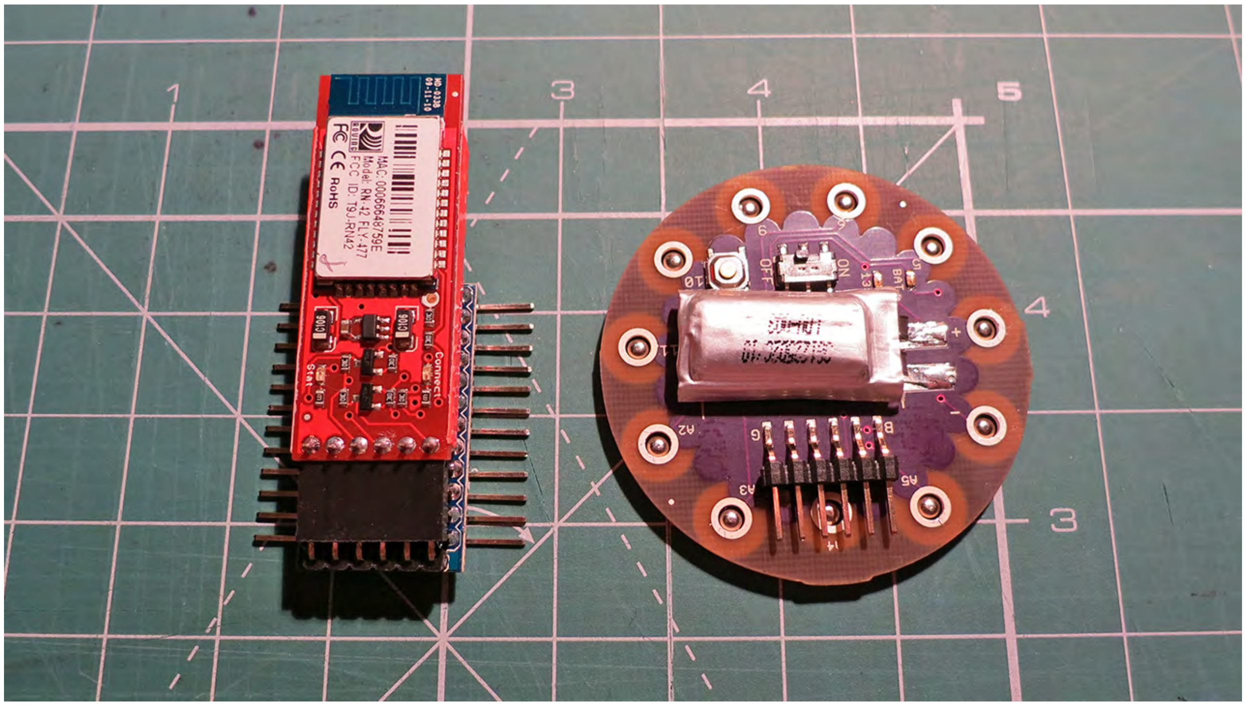


Another challenge in regards to the circuitry was the required voltage for the Bluetooth shield. When I tested with rechargeable Polymer Lithium Ian Battery (Sparkfun #PRT-00341), which has a nominal output of 3.7 volts at 850mAh, the Bluetooth shield was not working. Therefore I had to use a power source around 5 volts which can be charged/renewed easily. I ended up using a high performance AAA Alkaline Battery (e.g.

Fantasia

Duracell Ultra Power) with LilyPad Power Supply (Sparkfun #DEV-11259). This combination can supply about 20 +/- 2 minutes of power with a circuit like Fantasia which uses the Bluetooth shied.





Fantasia

Code

Arduino Code

To communicate the data (gathered by the accelerometer on the circuit) with the processing program, I used this simple sketch which writes the analog data to the serial port.

```
int firstSensor = 0;    // first analog sensor on the accelerometer (X axis)
int secondSensor = 0;   // second analog sensor on the accelerometer (Y axis)
int thirdSensor = 0;    // third analog sensor on the accelerometer (Z axis)
int inByte = 0;         // incoming serial byte

void setup()
{
  Serial.begin(115200);
  establishContact();  // this function sends a byte to establish contact
                       // until receiver (processing application) responds
}

void loop()
{
    // if we get a valid byte, read analog pins
    if (Serial.available() > 0) {
    // get incoming byte:
    inByte = Serial.read();
    // read first analog input
    firstSensor = analogRead(A1);
    // delay 10ms to let the Analog Digital Converter (ADC) recover
    delay(10);
    // read second analog input
    secondSensor = analogRead(A2);
    thirdSensor = analogRead(A3);
    // send sensor values to the serial port
    Serial.write(firstSensor);
    Serial.write(secondSensor);
    Serial.write(thirdSensor);
  }
}
  // establish connection function that was called in void setup
void establishContact() {
    while (Serial.available() <= 0) {
    Serial.print('A');   // send a capital A to serial monitor for testing the connection
    delay(300);
  }
}
```

Fantasia

Processing code

I have written/modified several Processing sketches to represent triple axis data received through the serial port in different forms/styles. Below is a simple sketch that helped me figure out the results of moving the wristband in different directions/speeds.

```
import processing.serial.*;

Serial myPort;                        // The serial port
int[] serialInArray = new int[3];    // where the received data is stored
int serialCount = 0;                 // a count of how many bytes we receive
int xpos, ypos, zpos;                // ending coordinates of the colored lines
float xposMap, yposMap, zposMap;     // mapped results of the above coordinates
boolean firstContact = false;        // whether the micro-controller is sending data

void setup() {
  size(500, 500);  // stage size (for full screen enter screen maximum dimensions)
  background(0);   // stage background color
  smooth();        // for anti-aliased smooth lines
  frameRate(10);   // faster frame rates draws new lines faster (maximum 60 fps)

  // print a list of the serial ports, for debugging purposes:
  println(Serial.list());

  // open whatever port is being used on your machine
  String portName = Serial.list()[6];
  myPort = new Serial(this, portName, 115200);
}

void draw() {
  // to gradually clean the stage
  fill (0, 1);  // black colour with an opacity of 1%
  noStroke();
  rect(0, 0, width, height);

  stroke (xpos, ypos, zpos);  sets the line color in RGB format
  line (width/2, height/2, xposMap, yposMap); // draws the line starting from the centre of the stage
}

void serialEvent(Serial myPort) {
  // read a byte from the serial port
  int inByte = myPort.read();
  // if this is the first byte received, and it's an A,
  // clear the serial buffer and note that you've
  // had first contact from the micro-controller,
  // otherwise, add the incoming byte to the array
  if (firstContact == false) {
    if (inByte == 'A') {
      myPort.clear();          // clear the serial port buffer
      firstContact = true;     // you've had first contact from the micro-controller
      myPort.write('A');       // ask for more
  }
 } else {
  // Add the latest byte from the serial port to array
  serialInArray[serialCount] = inByte;
  serialCount++;

  // If we have 3 bytes (from the three analog sensors)
  if (serialCount > 2 ) {
   xpos = serialInArray[0];
   xposMap = map(xpos, 0, 255, 0, width); // maps the results to the width dimension of the stage
   ypos = serialInArray[1];
   yposMap = map(ypos, 0, 255, 0, height); // maps the results to the height dimension of the stage
   zpos = serialInArray[2];

   // print the values (for debugging purposes only)
   println(xpos + "\t" + ypos + "\t" + zpos);

   // Send a capital A to request new sensor readings
   myPort.write('A');
   // Reset serialCount:
   serialCount = 0;
  }
 }
}
```

Other technical issues

I am not sure if it is the code that I am using or I have a faulty Bluetooth device, but every time I stop the processing application, and try to run it again and receive data from the wristband, I have to turn off/on the power on the circuit, trigger the Bluetooth connection with the computer (using the Bluetooth preferences pane on the Mac OS), in order to be able to receive data for the Processing sketch.

Also, due to electrical interferences, sometimes there is a delay in communication between Arduino and Processing.

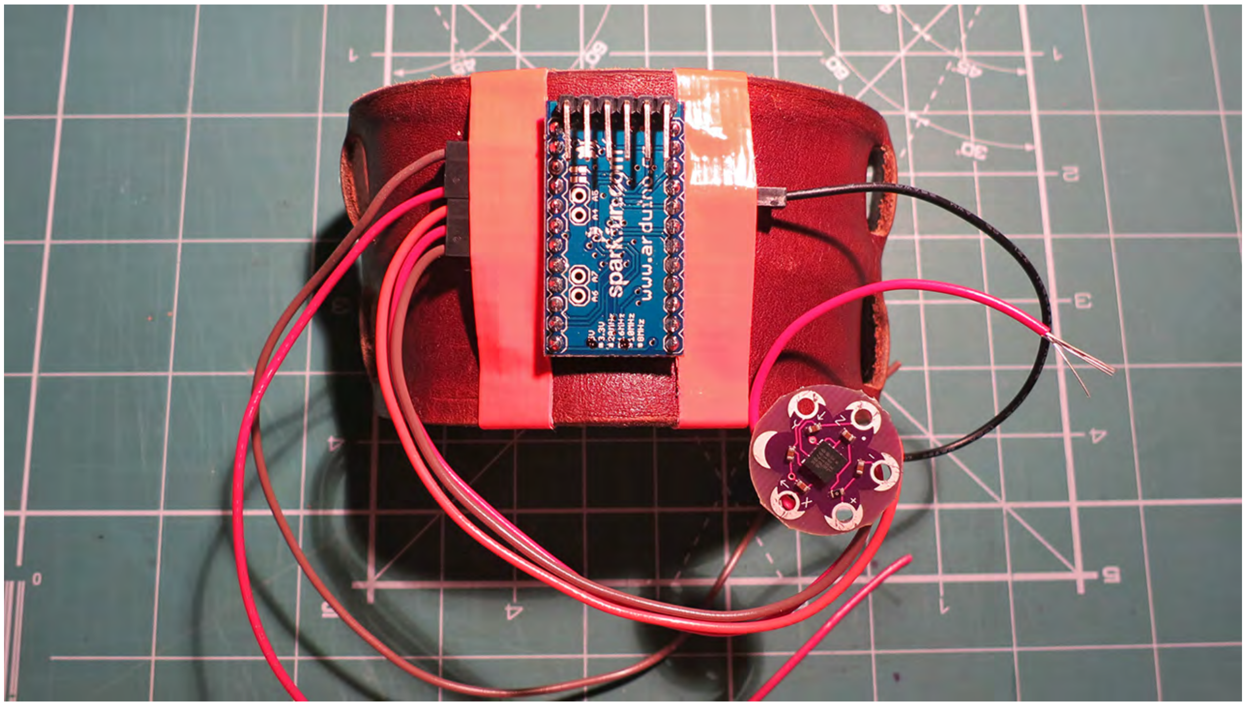
These are on top of my list that I would like to improve in future iterations of this project. I am also planing to use a XBEE shield (instead of the Bluetooth) in the next version.

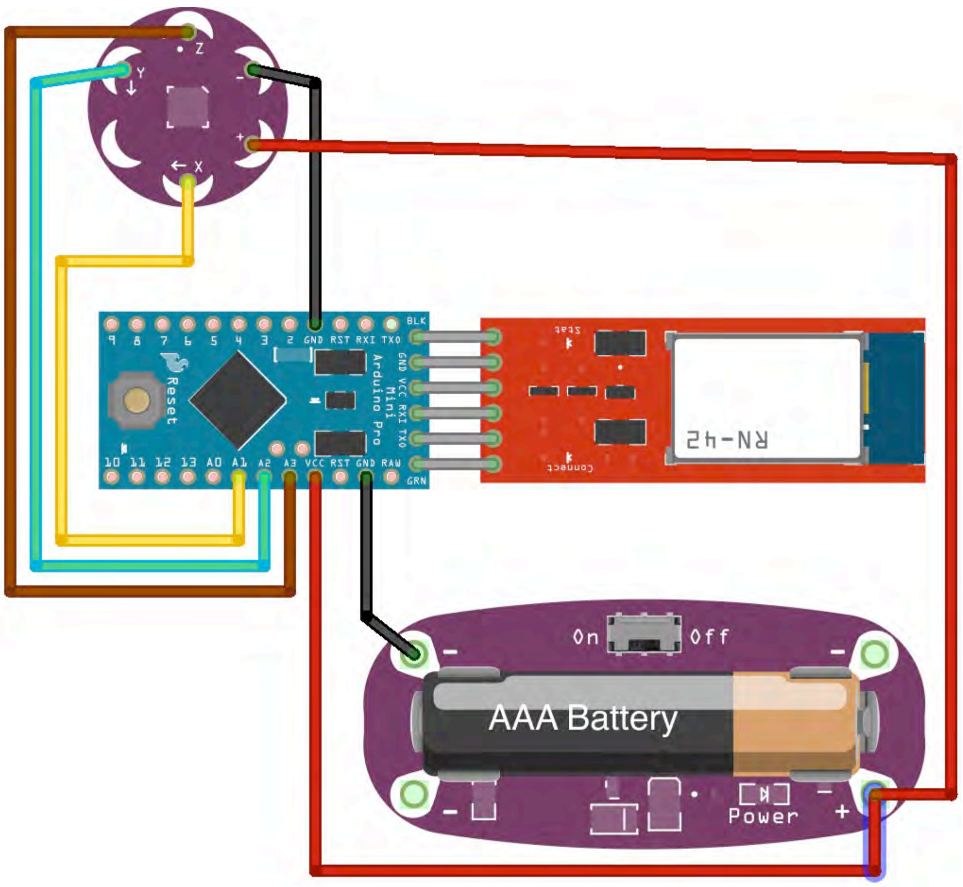
Video links

Screen capture of different Processing sketches in action
www.borxu.com/videos/fantasia.mov

Process video
www.borxu.com/videos/fantasia-process.mov

Fantasia




Z
Y
X
Reset
Arduino Mini Pro
GND RST RXI TXO
GND VCC RXI TXO
VCC RST GND RAW
BLK
GRN
RN-42
On
Off
AAA Battery
Power