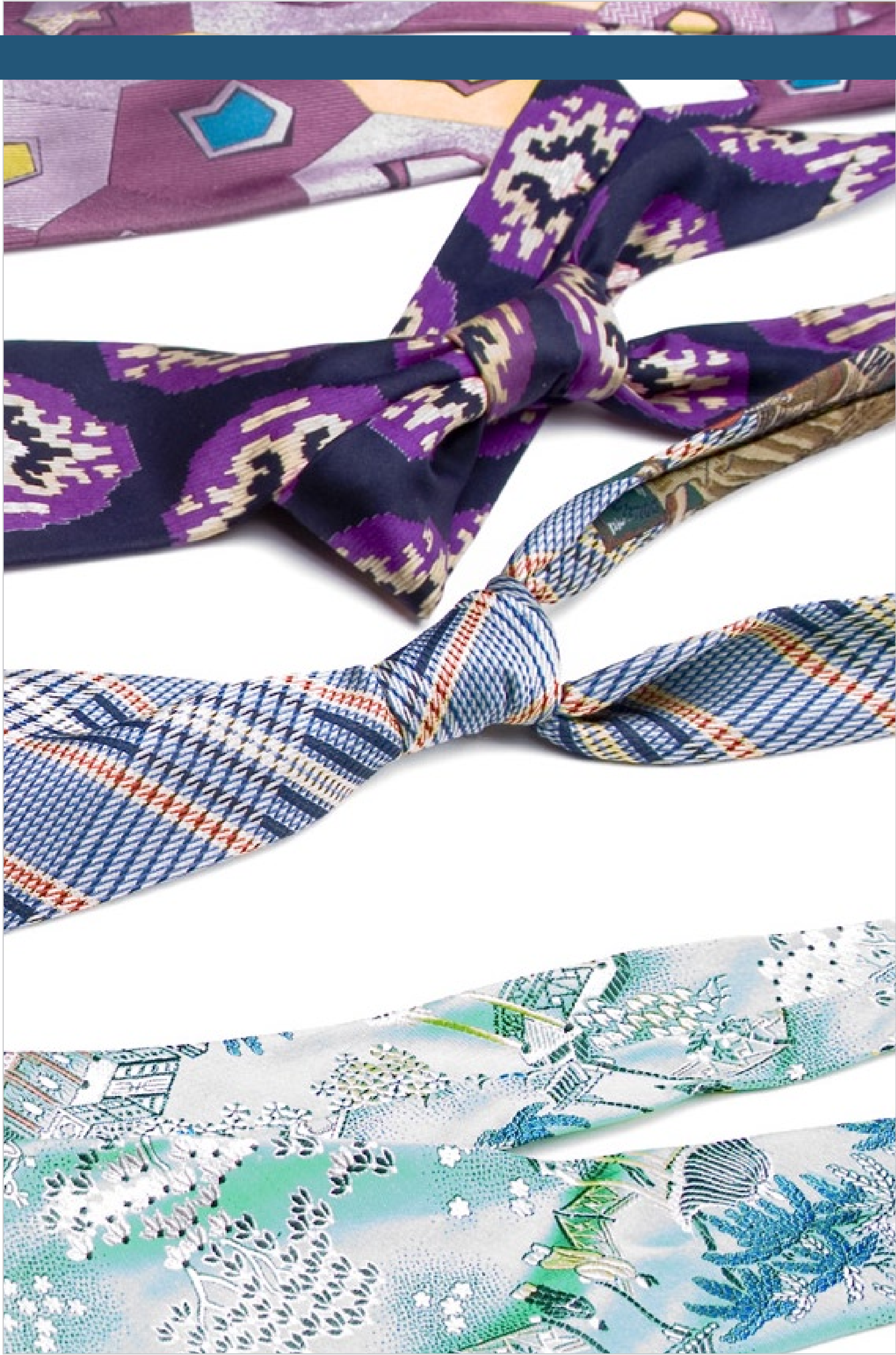
The Audio Tie Project

BY: Rouien Zarin
MA Interaction Design (Year 2)
Umea Institute of Design

Table of Contents



The Audio Tie Project

A one week project exploration of creating a wireless audio level meter and deploying it at an event or function where there is a DJ and lots of people.

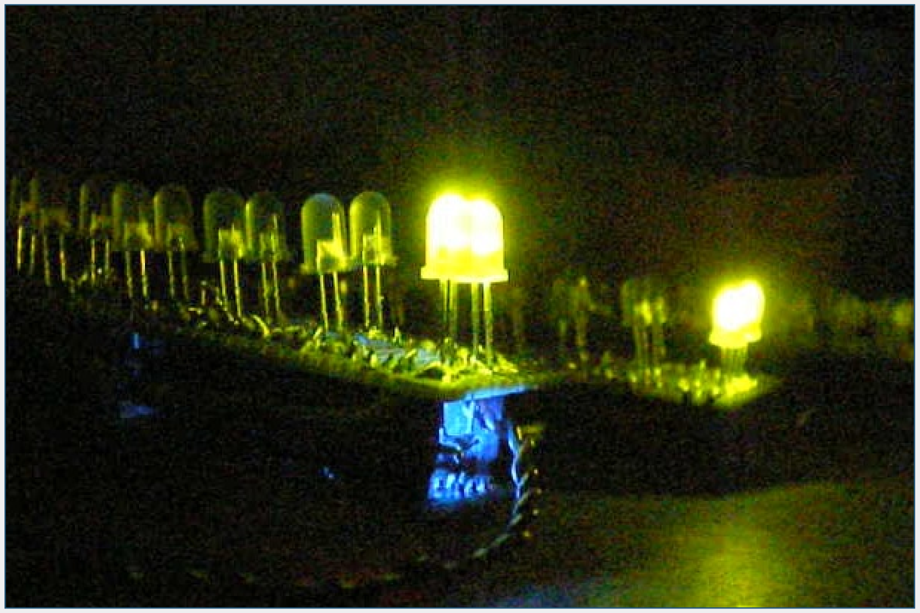
Each tie is associated with a specific level in the Audio Spectrum and moves in real-time in relation to the music at the venue.

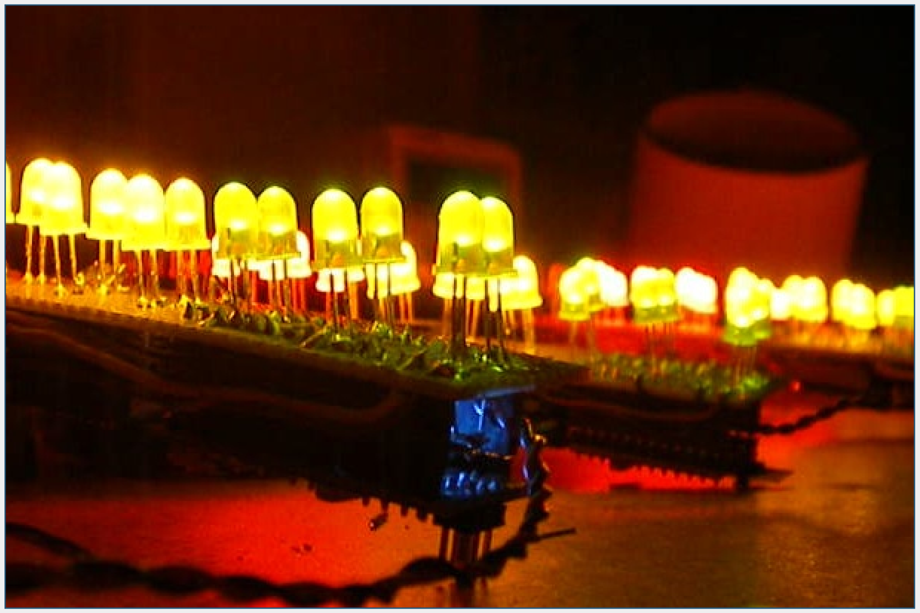
The tie's are completely wireless using a radio transmitter to get the signal from the DJ source. This gives the tie's a range of up to 1 mile (pending obstacles).

Each tie is powered by an ordinary 9V battery, giving the wearer the ability to move freely throughout the venue with the technology hanging around their neck.

Special Thanks

I would lke to thank Camille Moussette our tutor and lecturer for his guidance and support in addition to Oskar Fjellström.





To realize this project I connected a computer to the Dj's sound signal and analysed the sound using processing, then splitting it into 5 hertz averages to be sent over Serial to an Xbee device and finally broadcasted wirelessly to 5 different people.

Each person receiving these broadcasts had a circuit board with Xbee and Arduino Drving LEDs which displayed the audio information in real-time.

Another idea was also to have a signal reading so that if the signal strength of the Xbee's is weak then the leds get dimmer. For example when the volunteers get closer to the DJ booth the leds are at their fullest strength, but dims the further they are away from the source.

The materials required to wire five people:

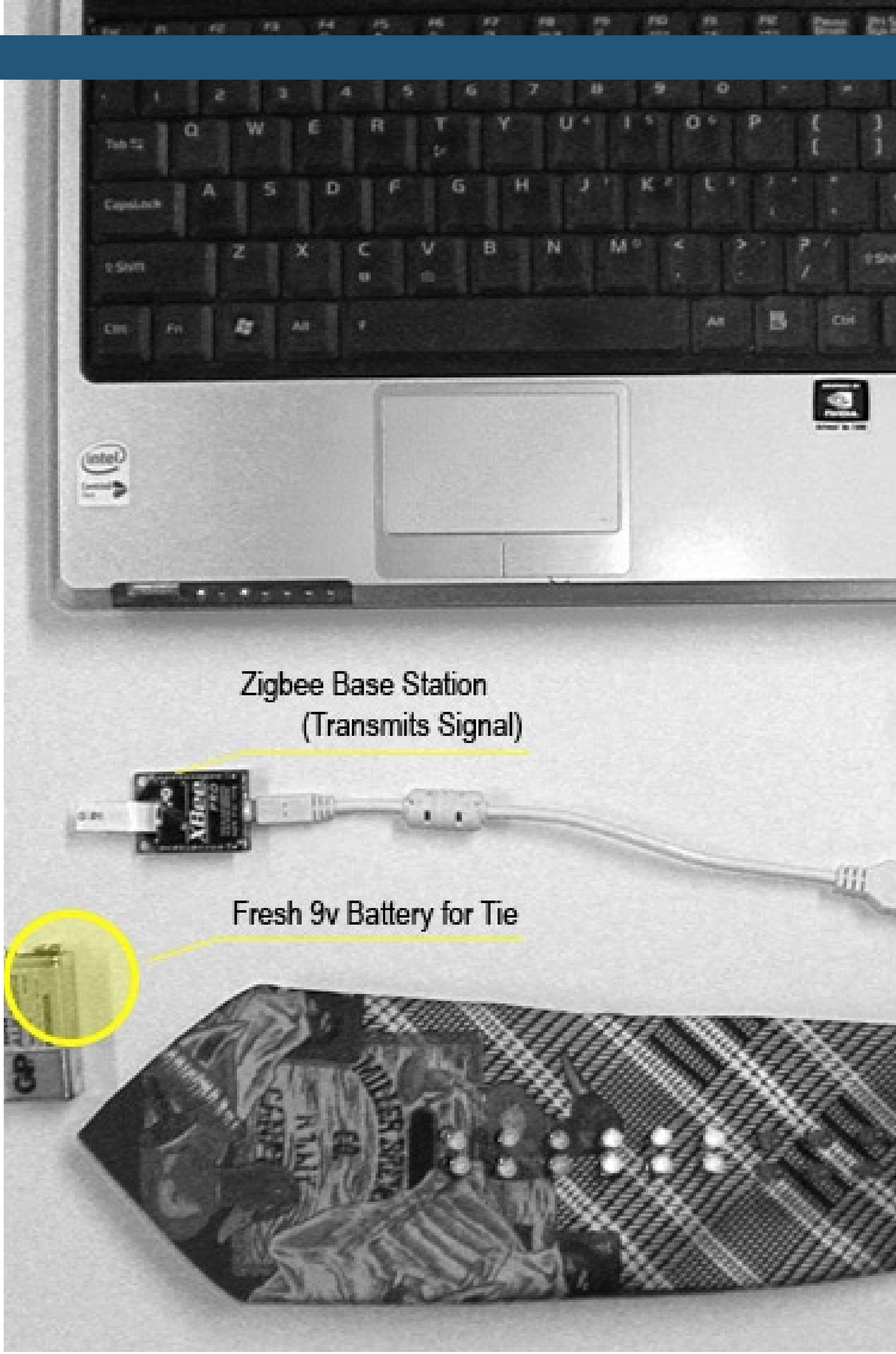
- LEDs in Different colours
  10 Red, 6 Green and 6 Yellow / Tie
- 5 x Arduino mini's
- 6 x Xbee transmitters
- 1 x Computer, running
  the processing sketch
- 5 x 9V Batteries
- 1 x USB to Serial converter
- 1 x sound source

A Bit about the Hardware and Software

Processing is an open source programming language and environment for people who want to program images, animation, and interactions.
more info at www.processing.org

XBee 802.15.4 and extended-range **XBee-PRO** 802.15.4 are low-power radio transmitters/receivers that use the IEEE 802.15.4 networking protocol for fast point-to-multipoint or peer-to-peer networking.
more info at www.digi.com

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.
more info at www.arduino.cc



Die Roboter

My first reason for choosing this approach vs. some of the other idea's I had was to really stick with the wireless theme and I also did not want a computer or LCD projector to play a central role in the design.

Also being a product of the 80's I grew up with bands like Kraftwerk who used a tie with some leds in a music video called 'Die Roboter'. Although the tie's were not actually linked to the music, it was still an interesting approach back then as it was an early example of the merg-ing of electronics and fabrics.

You can see the music video here on youtube.

Kraftwerk - Die Mench Maschine

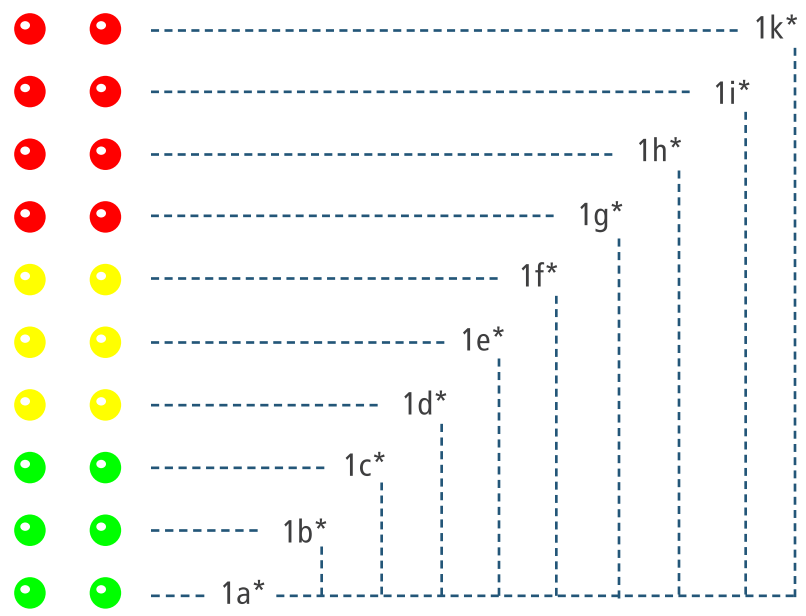
Concept Sketches

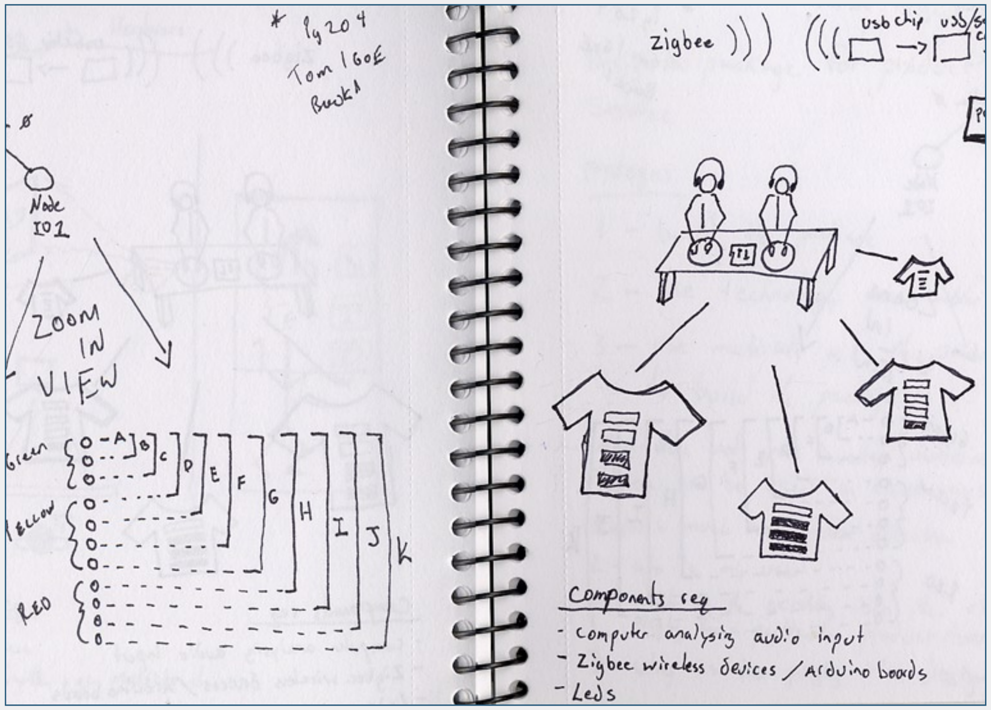
I sketched out an idea of how the setup might look like. The first approach was to have the LEDs embedded in T-shirts to show the audio. But since the venue was Halloween, most people would have a costume and therefore the T-shirts weren't going to fit well with the costumes.

As a result I chose to look at embedding the Leds in Tie's. Because of the shape of the ties, they leant themselves well to the
long thin bars of an audio meter. Also they could be temporary, easy to put on and take off.

Data Transfer

The other thing I looked at was data transfer and how I was going to actually get the audio signal to each Tie. After reading abit about the Zigbee's I decided to come up with my own protocol for activating the LED sequence for each Tie. In the example below, 1 signifies the ID number of the tie.





Rough Sketch of Layout and Data Formatting

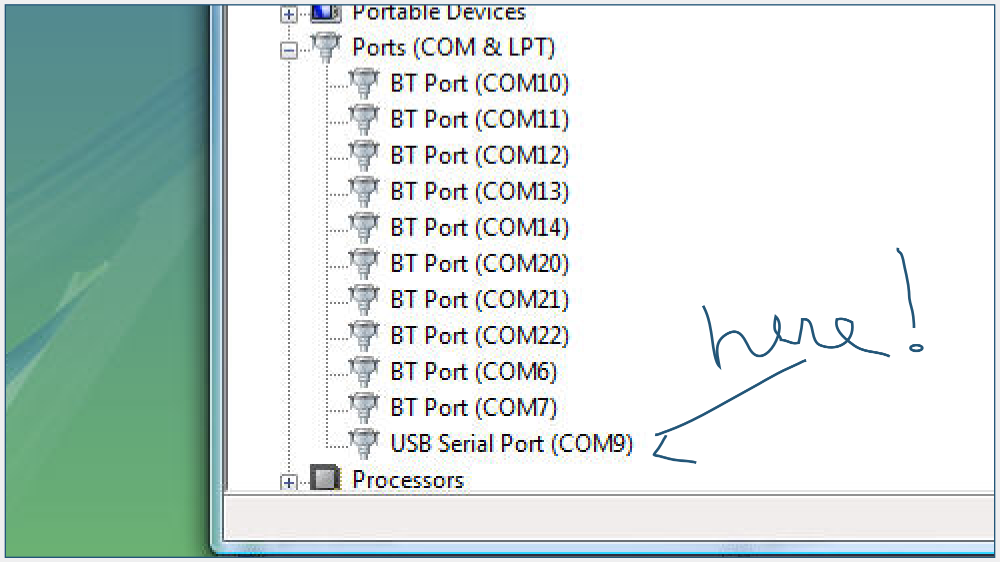
I first started working with Processing and the Minim Libraries to analyze sound signals. Within a day I had a line going to my Mic-in port and visually displaying an equalizer in the Processing output window.

Then it was a matter of connecting the Xbee base station (via a ComPort) to the computer and identifying which ComPort it was attached to in the Sketch.

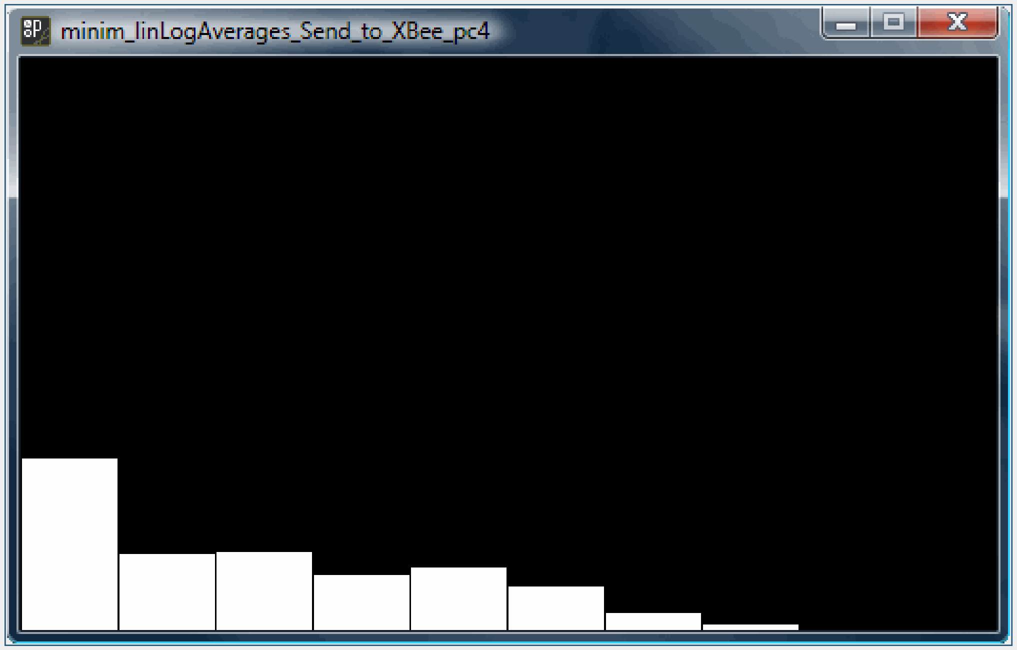
```
void setup()
{
  size(512, 300);
  frameRate(30);
 // ######  get the serial ready for Xbee ##
  String portName = Serial.list()[11];
  println(Serial.list());
  myPort = new Serial(this,portName, 19200);
// ####### Done ###########################
  height3 = height/3;
  height23 = 2*height/3;
  // always start Minim
  // minim = new Minim(this); // mac only
  Minim.start(this);
// load line in, default buffer is 1024
  djsource = minim.getLineIn(Minim.STEREO, 1024);
```

In the above snippet, the available serial ports are printed out in the output window as a list. You then look for the right ComPort Number that your device is attached to. In this case COM9 maps to Serial 11. Then you update the code accordingly and the sketch should be able to communicate to your Xbee device.

*Tip, double check your transfer speeds. If you are getting some strange characters in your debug Arduino eg. @#$# then chances are your baud rate is incorrect.


```
Portable Devices
Ports (COM & LPT)
  BT Port (COM10)
  BT Port (COM11)
  BT Port (COM12)
  BT Port (COM13)
  BT Port (COM14)
  BT Port (COM20)
  BT Port (COM21)
  BT Port (COM22)
  BT Port (COM6)
  BT Port (COM7)
  USB Serial Port (COM9)
Processors
```

Using Device Manager (Vista) to determine the right ComPort



Processing the Audio Source

In the beginning, I had the Arduino connected to the computer and I was controlling my LEDs using processing.

I had a Xbee configured with transfer speed and unique ID and I had a base station configured to broadcast, then came the task of sending data over serial through the Base station so that it would be picked by my Xbee slave.

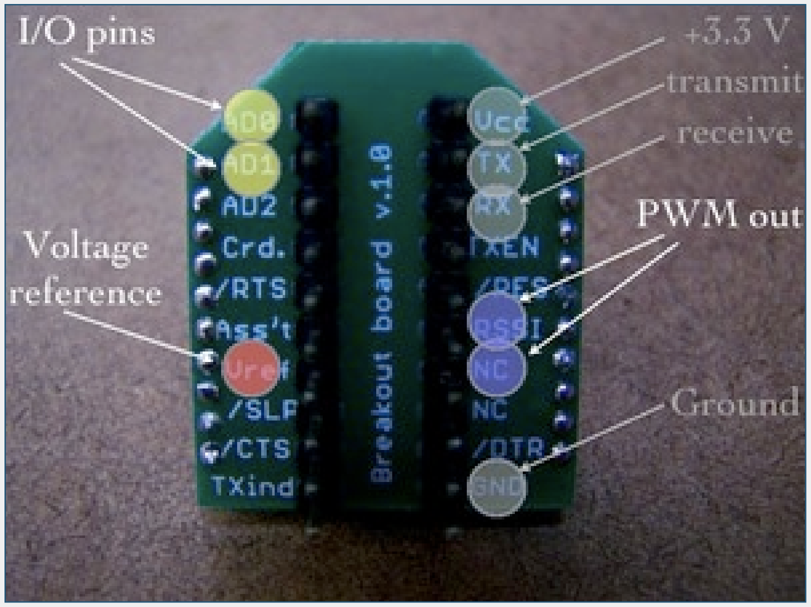
Debug Mode

The way I was able to see what data was being sent to my Xbee slave was to use another arduino and the softSerial library included with Arduino 0007 and later.

```
#include <AFSoftSerial.h>
#include "string.h"

AFSoftSerial debug =  AFSoftSerial(3, 2);
```

Using the above code I was able to plug into ports 3, and 2 of the Arduino receiving data, and opening a listening connection on the ComPort that the debug Arduino is connected to. Then it was just a matter of using `debug.println` to print out values to my debug Arduino, and it should show up in serial monitor program that you have opened.



Xbee Breakout Board

Arduino Sketch

```
void setup() {
  // configure serial communications:
  Serial.begin(19200);
  // my own debug serial port
  debug.begin(9600);
  debug.println("I'm awake");
  pinMode(13,OUTPUT);
  blink();
  blink();
  blink();
  debug.println("Configuring Module");
  // set XBee's parameters
  setupXbee();
  debug.println("Configuration done");
}
```

Configuring an Xbee Base Station

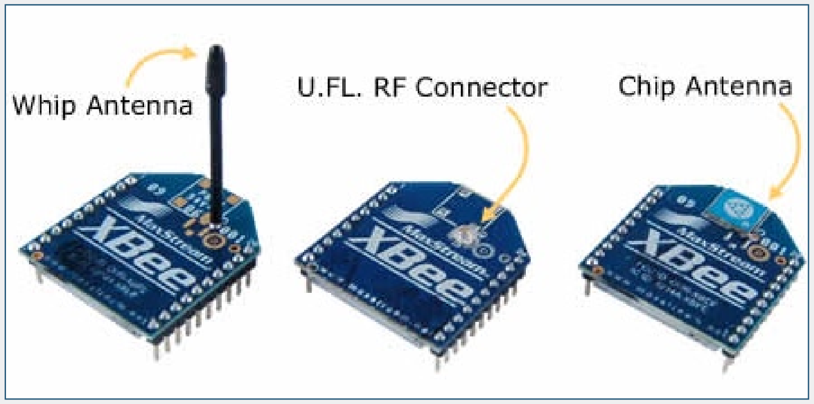
The Base station was programmed using a console/terminal program like Zterm (mac) goSerial (pc), Putty (pc), HyperTerminal (pc). *note: needs to be plugged into the computer with a serial - USB converter.

+++	// (is the instruction to enter command mode)
ATRE	// (restore config to defaults)
ATID0	// sets the Personal Area PAN ID. // Change ID if you want to create your own network (this is a zero)
ATMY0	// Sets the module ID address. Must be unique (this is a zero also)
ATDLFFFF	// Sets destination address to broadcast to the whole PAN network.
ATBD4	// Sets the interface data rate to 19200. BD3 is the default with 9600.
ATWR	// Writes the changes/settings to memory
ATCN	// Exits command mode

Configuring an Xbee Slave

The Xbee Slave Nodes were programmed using an Arduino Sketch. Since you can emulate the inputs using programming rather than typing manually.

This saves time and you can keep track of what was programmed into the Xbee chip.



Various Xbee RF Chips

Arduino Xbee Config Sketch

```
void setupXbee() {
  Serial.print("+++"); //command mode:
  delay(1100);
    i = readUntil('\r'); // wait for "OK\r"
// set the destination address, 16-bit addressing.
// if using two radios, one radio's destination
// should be other radio's MY address, & vice versa:
  Serial.print("ATDH0, DL0\r");
// set my address using 16-bit addressing:
  Serial.print("ATMY");
  Serial.print(my_id);
  Serial.print("\r");

// set the PersonalAreaNetork ID
  Serial.print("ATID0\r");
  Serial.print("ATBD4\r"); //transfer speed 19200KB/s
  //commit changes to memmory
  Serial.print("ATWR\r");
  // put the radio in data mode:
  Serial.print("ATCN\r");// exit command mode
}
```

Prototyping

Once the Processing Sketch was working properly I decided to start prototyping the circuit using a basic breadboard.

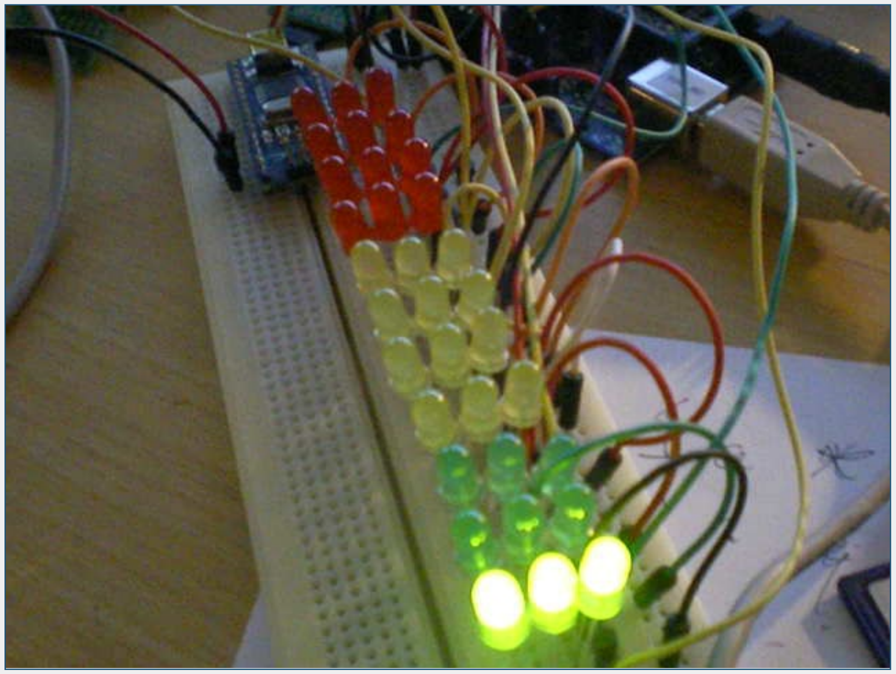
In this setup it was easy to add rows of LEDs and wire them to an Arduino mini.

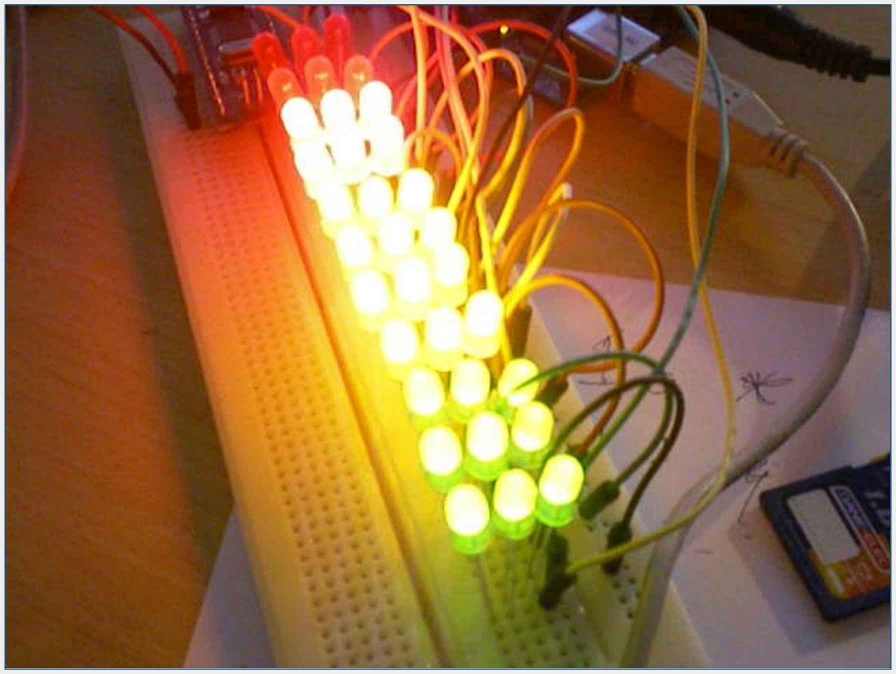
One thing that became apparent during this setup was that I was not going to be able to dim the LEDs as I had intended in the original brief, as the mini only had a limited number of PWM (Pulse Width Modulation) outputs.

Deciding on Height

Also I had decided that my maximum rows (height) of LEDs that I was going to have was 11, as there are exactly 11 outputs on the Arduino (not counting pins 1TX, 2RX or 13).

Apparently there is a way to control more than 11 but this inolved some duplexing and considering the time restraints I decided that this was out of the scope of this project.





Once I had the base station connected to the computer and was getting Processing to send my data through the Xbee, I connected it to the Prototype.

Using a Buffer

I noticed that the data that I was sending was almost always getting to the Xbee but not quite, sometimes a character would be lost here and there and would throw off the values I was expecting. It was for this reason that I decided to create a buffer array in Arduino.

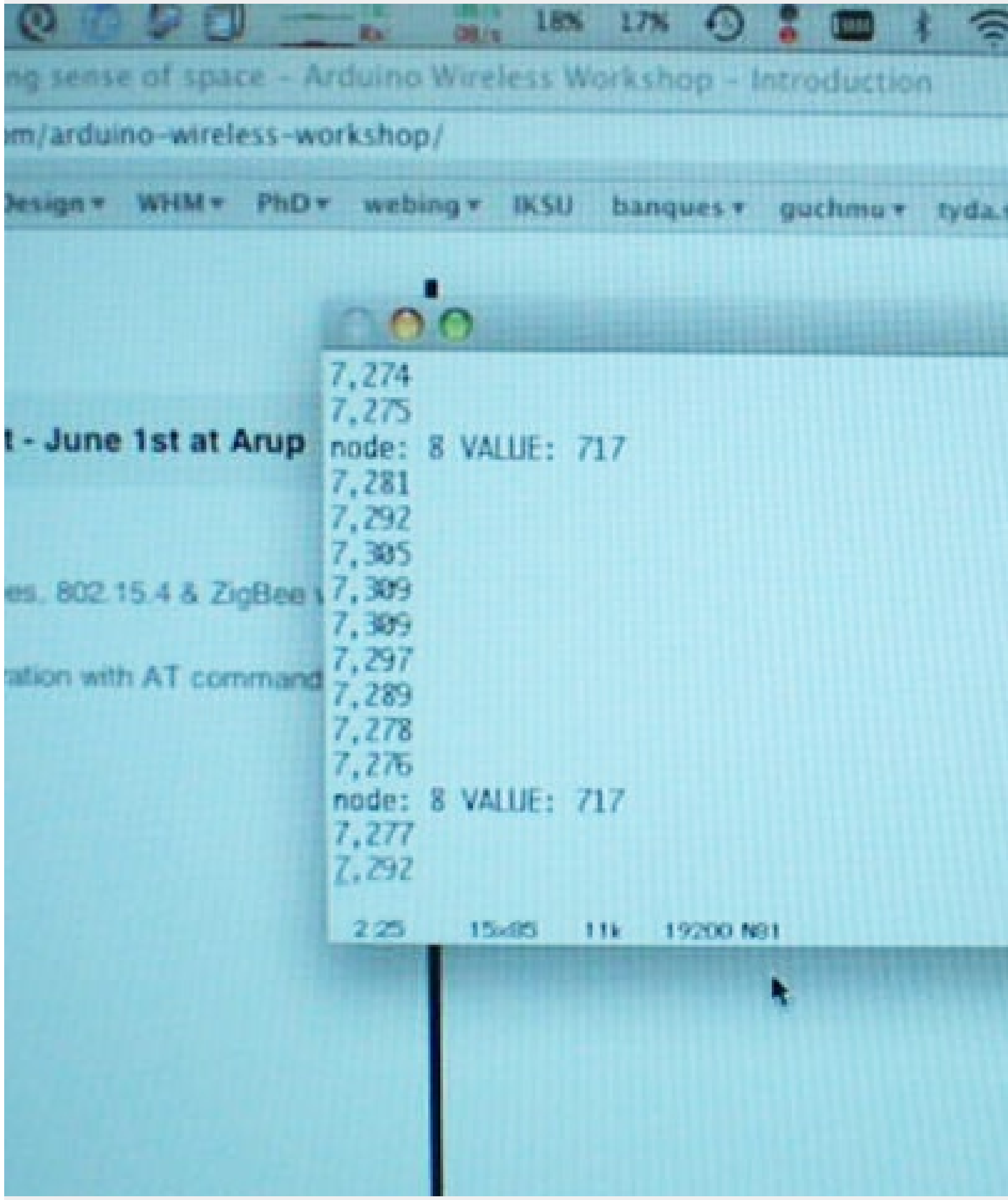
```
void handleSerial(){

  inByte = Serial.read();
  //debug.print(inByte);
  if (inByte != 42){
    inString[stringPos] = inByte;
    stringPos++;
  }
  else{
    // process incoming
```

In the above snippet of code my function **`handleSerial()`** is getting the incoming raw data.

I am then going through the variable **`inByte`** and putting it into an array called **`inString`**. I also have a check for ascii value **42** which is the ***** delimiter I am using to signify the end of my data packet.

As soon as the Arduino comes across the ***** then it starts to evaluate which LED sequence to light.



```
7,274
7,275
node: 8 VALUE: 717
7,281
7,292
7,305
7,309
7,309
7,297
7,289
7,278
7,276
node: 8 VALUE: 717
7,277
7,292
```

ng sense of space - Arduino Wireless Workshop - Introduction
m/arduino-wireless-workshop/
Design ▾ WHM ▾ PhD ▾ webing ▾ IKSU banques ▾ guchmu ▾ tyda
t - June 1st at Arup
es. 802.15.4 & ZigBee
ation with AT command
2:25 15x85 11k 19200 N81

Serial Terminal Program displaying Xbee Values

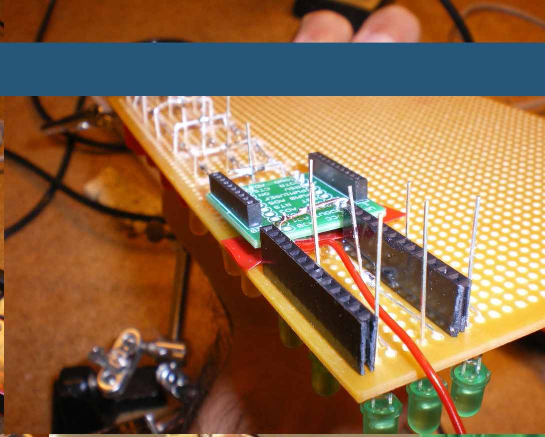
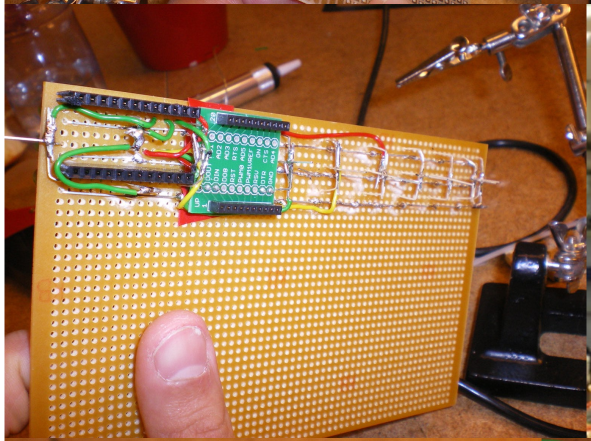
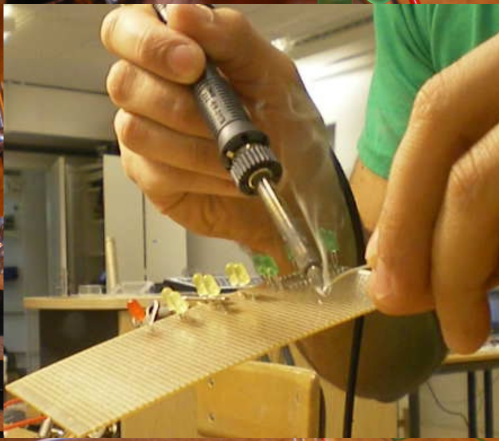
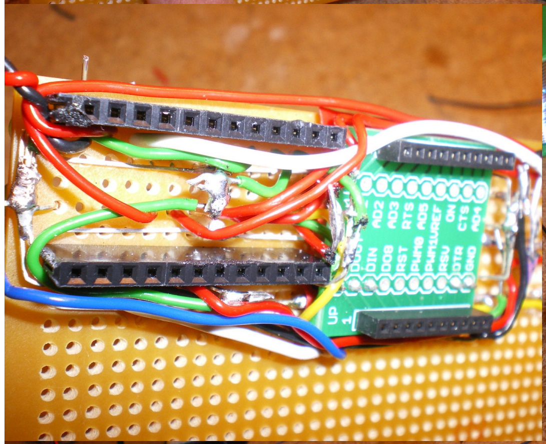
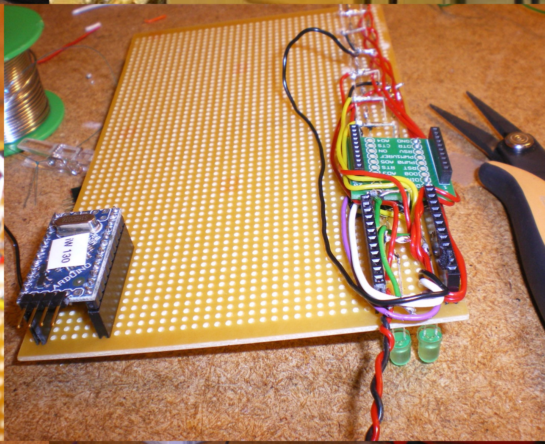
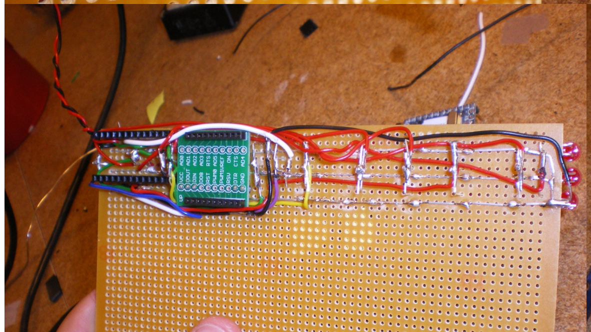
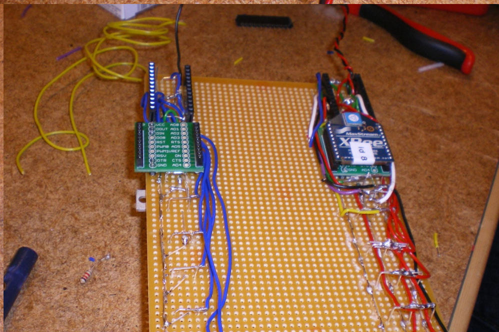
The first one was the hardest. Just figuring out where everything goes and not being all that great at soldering. But I managed to get a pretty compact'ish circuit board that worked.

Scaling the Production

Once the first one was out of the way, the rest was pure grunt work. I also scaled down slightly the number of LED's so that I could make it a bit less soldering for myself and meet my self imposed five tie quota by Friday.

Friday Oct 31st, 2am

Spent alot of late nights soldering the boards but finally made it. I had one more day to embed the circuits into the tie's but I was confident it wouldn't be too much trouble.

Playing with Fabric

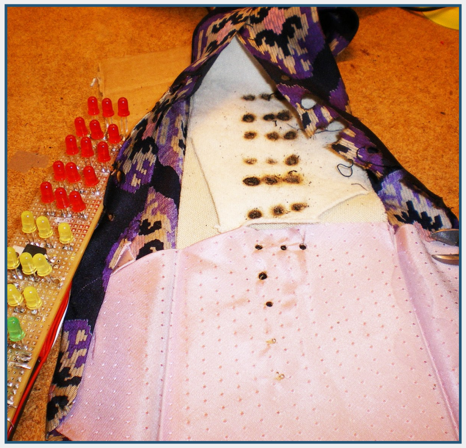
I now needed to burn holes through the ties to the spacing and size of the LEDs. I did this using a soldering gun which worked nicely but since the tie's were of a polyester fabric, it was quite toxic.

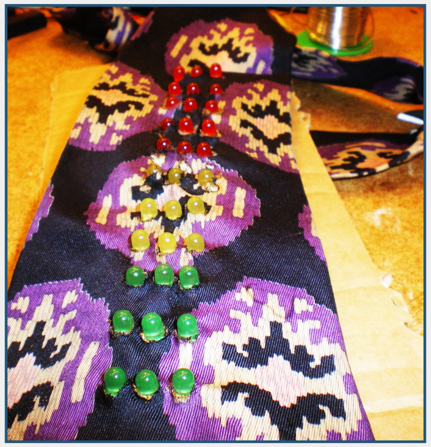
Then I poke the LEDs through the holes, trying to be careful enough not to break any connections, as it could be a nightmare to trouble-shoot.

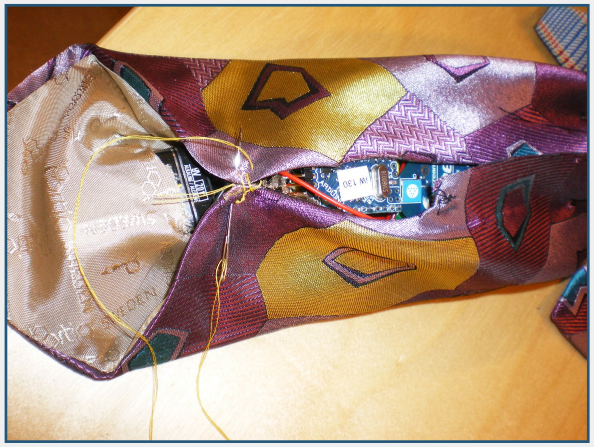
After the circuit's were in place I stitched a seam in the back of the tie to bind it enough so no loose material was visible from the front, but you could still access the Xbee and Arduino Mini if you needed to.

The final step was to create a small pocket/compartment for the 9V battery which would eventually power the Tie.

My Five Tie's


Burning the Holes


Poking the LEDs through


Stitching the Back

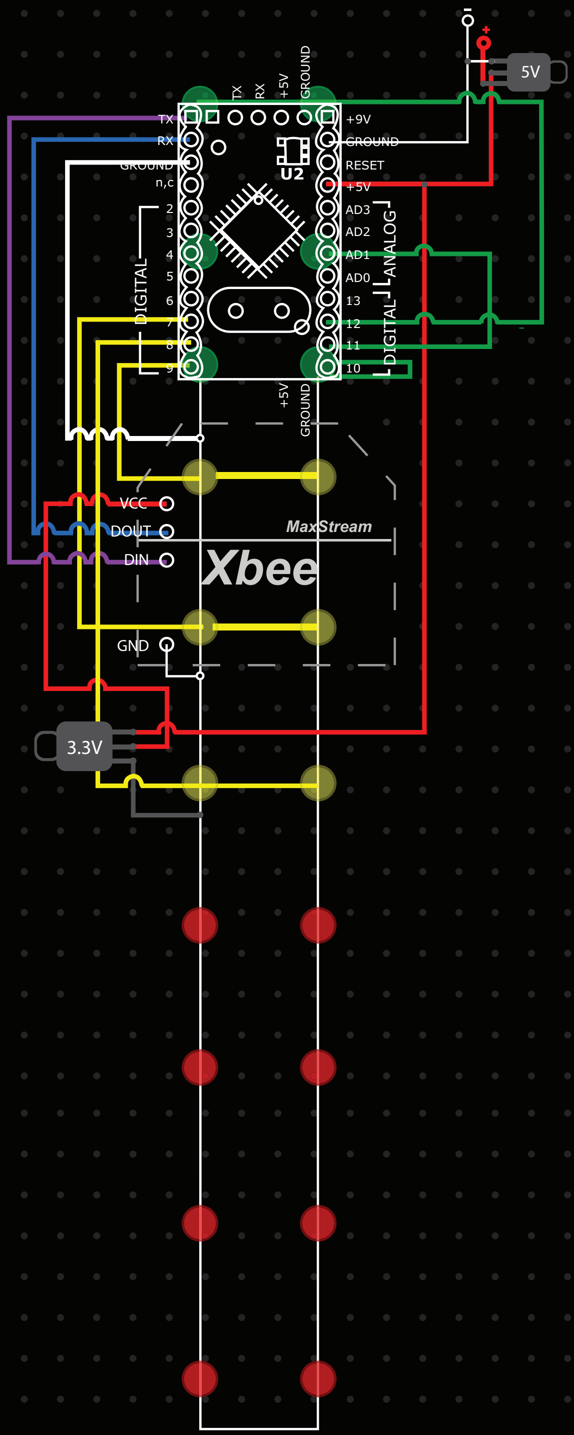
Power

To help assist in the compact design I wanted to use a 9V battery but in some initial tests, found that it was too much juice for the Arduino Mini to handle. I decided to put a 5V Voltage Converter between the 9V and the Mini. This helped cap the Voltage.

I used another one in the setup betweeen the Mini and the Xbee converting from 5V down to 3.3V so as not to damage the Xbee.

One thing that I missed in this setup, if I was to do it again was to include some resistors for the LEDs. They simply draw too AMPs from the 9V and it only takes roughly 45mins to drain the battery.

Another feature that I should have included are switch's so that you don't have to constantly plug/unplug the battery for demo's.



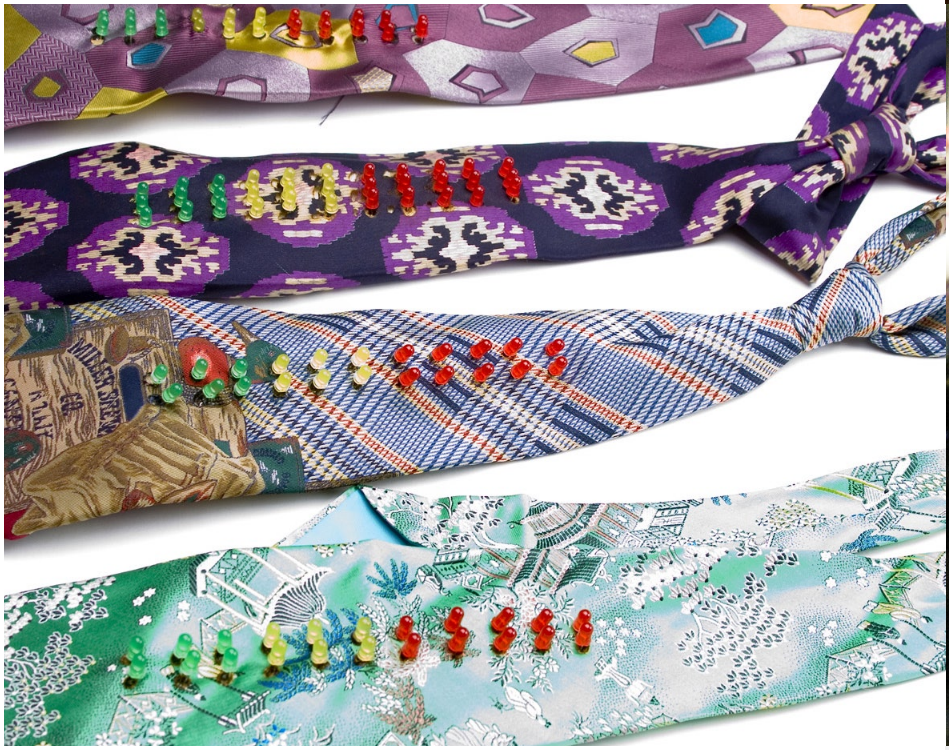
It was a busy week, and I encountered some of my own hardware and software limitations but I felt that by the end I had overcome them.

The venue was a great way to showcase our efforts and try the concepts/installations out on real people. There was the added dimension of making everything "party proof".

My intention was to have the music that was already playing play another role and be dispersed around the room visually.

In the end I was happy with the results as it added to the atmosphere of the event and prompted discussion from other people in the room.

To view the video of the Tie's at the Halloween Party click here



Monday

- ✓ Got processing to work with the minim sound library
- ✓ Got Xbee broadcasting values to other Xbees on the same Personal Area Network

Tuesday

- ✗ Trouble with data over serial and data being lost
  - ↳ Solution: Built a buffering array in the receiving Xbee
- ✗ Had to get rid of the idea of having LEDs dimming due to number of PWM outputs.
  - ↳ Solution: Decided instead to simply turn on and off the LEDs and use the max number of output pins
- ✓ Created working prototype of audio meter.

Wednesday

- ✓ Got the buffer working, started transmitting wirelessly
- ✗ Having trouble getting an immediate response with audio over Xbee
  - ↳ Solution: Cleaned up some of my processing code, found unnessesary loops, was multiplying data

Thursday

- ✗ Size of the setup is too bulky using the Arduino Diecimila
  - ↳ Solution: migrated over to the Arduino mini's and Nano.
- ✓ Finished soldering my first board

Friday

- ✗ Have to solder 4 more boards to meet my quota.
  - ↳ Solution: stayed till 2am but got them all soldered
- ✗ When testing boards we found circuits use too much power and drain battery quickly
  - ↳ Solution: Decided to release the Tie's at the venue for only 1 hour

Saturday

- ✓ Started embedding circuits into the ties.
- ✗ Burning holes through the polyester was really toxic
  - ↳ Solution: conduct it in a better ventilated environment or wear mask
- ✓ Recruited 5 volunteers to showcase the ties at the event

Links

Example Xbee Scripts from an Arduino Workshop
http://www.makingsenseofspace.com/arduino-wireless-workshop/

Minim Library Reference
http://code.compartmental.net/tools/minim/

Arduino SoftSerial Interface Reference
http://www.arduino.cc/en/Tutorial/SoftwareSerial

Books

Making Things Talk- Practical Methods for Connecting Physical Objects
Author: Tom Igoe, ISBN 13: 9780596510510