IAT 884
Lab 5
Interfacing with the Computer

Preparation:

A. Read in Programming Interactivity:
Audio In Processing (p.202 – 209)
Debugging Your Application (p.122-124)
Communicating with Other Applications: (p. 259 – 261)

B. Download and install the full ***Minim2.0.2*** **Library Only** distribution from:
http://code.compartmental.net/tools/minim/

To install the distribution, unzip the file to extract the minim folder. Move this folder into the **libraries** folder in your Processing application directory. Create the *libraries* directory if it doesn't exist.

After starting (or restarting) Processing you should now have a new set of example sketches available using the pull down menu: File→Examples → Libraries → Minim(Sound).

Open the file "LoadSample" located in the minim examples folder and compile and play it. If everything is installed correctly you should not get any errors. Click on the java popup window and press "k". You should hear a drumbeat and see a waveform on the screen.

C. Find a short (1-5 sec) MP3 audio sample that you would like to use for the in class assignment. You can download some at http://simplythebest.net/sounds/MP3/sound_effects_MP3/ .

Resources

Minim Resources:

Minim Java Docs:
http://code.compartmental.net/minim/javadoc/

Example code:
http://code.compartmental.net/tools/minim/manual-playable/

Serial Comminication Resources:

Resources on Serial Communication:
http://www.ladyada.net/learn/arduino/lesson4.html

Resources on Communicating between the Arduino and Processing:
https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing

Arduino Meets Processing:
http://webzone.k3.mah.se/projects/arduino-workshop/projects/arduino_meets_processing/instructions/index.html

Arduino's Built-in Pull-Up Resistors

http://www.arduino.cc/en/Tutorial/DigitalPins

Other Resources for further exploration:

Interfacing Arduino with Processing using Firmata:
http://www.arduino.cc/playground/Interfacing/Processing

Interfacing Arduino with Max/MSP
http://www.arduino.cc/playground/Interfacing/MaxMSP

In Class Exercise:

Equipment:

- Arduino Board
- Push Button or Switch
- 1 x 10K Ohm Resistor (Optional- Brown, Black, and Orange Striped)
- Wire
- Breadboard

In Workshop:

The goal of this workshop is to design a circuit that uses a push button switch to trigger a sound sample. The following steps break the task up into separate parts to make it easier to program the Arduino and build the circuit. If you want to skip all the steps and build everything in one go, that is fine as well.

A. Using serial communication, send a specific "trigger" number from the Arduino board to Processing. When the trigger number is received have Processing respond by displaying the number using the "println" command. (Use the Serial Communication Resource links above)

B. Modify your code from A to do the following: Instead of printing out the trigger number, have Processing play an audio sample using the Minim library.

C. Modify your code from B to do the following: Add a circuit that uses a pushbutton to play the audio sample. You can use Arduino's built-in Pull-up resistors to make this easier.) You should modify your code so that the Arduino board sends the trigger number when the button is pressed. (May require using "edge detection."
See http://sproutlab.com/arduino/tutorials/3-serial-communication/ for help with this.)

OPTIONAL:

Modify your code in part C do the following: When processing receives the character number have it return a different number back to the Arduino. When the Arduino board receives this number it should provide some indication that it has received confirmation. (The simplest solution to this is to light up the LED on pin 13 for a set amount of time. But you could also use a piezo buzzer or some other type of actuation.)

Serial Communication Code

Arduino Commands

```
 Serial.begin(9600);                          //initiate serial communication at 9600 baud

if (Serial.available() > 0) {                 // checks to see if there is serial data waiting
        incomingByte = Serial.read();         // read the incoming byte to a variable
}

   Serial.print(100, DEC);                   //send the number 100 to the serial channel
   Serial.print(10, BYTE);                    // send the byte 100 to the serial channel
```

Processing commands

```
import processing.serial.*;                   //serial library

println(Serial.list());                       //List all the available serial ports:

String portname = "COM4";                     // variable to store the name of the com port

Serial port;                                  //Create a new serial port called port

port = new Serial(this, Serial.list()[1], 9600);   //opens serial communication port

while(port.available() > 0){                  //See if there is data waiting to be received.
     String buf = port.read();                //Read serial data into the string  variable buf
}

port.write(buf);                              //Send data stored in buf to the serial channel
port.write(100)                               //send the number 100 to the serial channel
```

* For examples of usage see:
  http://sproutlab.com/arduino/2007/serial-communication-with-processing/

Code Samples: Serial Communication

Sending a simple serial message from the Arduino board

Output Message from Arduino (Serial Hello, World!)

```
void setup()
{
  Serial.begin(9600);        // ready serial comunication
}

void loop()
{
  Serial.print("Hello, ");    // prints a string
  Serial.println("world!");  // prints a string + newline

/* instead of using the Serial.println command, you could use the
Serial.print() command followed by a manual linefeed sent using the
command Serial.Print(10, BYTE).*/
```

Compile and upload this code to your Arduino board. To see the message that the Arduino is sending to your computer, you need to start the serial monitor by pressing the button on the far right that looks like this:

The message sent looks like this (H is the first character sent, so it's at the front of the message)

When the message is received on the other end, it's output like this (if output as a string)

(source: http://sproutlab.com/arduino/tutorials/3-serial-communication/)

Activating the built in Pull-Up Resistor on the Arduino

(Note the use of the digitalWrite command to an input pin)

```
pinMode(pin, INPUT);           // set pin to input
digitalWrite(pin, HIGH);       // turn on pullup resistors
```