Interactivity project:

To communicate between a sensor (let's say, attached to the Arduino) and Desktop software that can manipulate video, graphics and high quality audio, we use one of the built-in parts of a computer called a "serial port". The Arduino needs to send data at a specific data-rate to that (USB) port, and the Desktop software needs to be listening at the same port, at the same speed. If we want to print characters to the Arduino Serial Monitor for debugging and so forth, they need to be sent as ASCII codes using "`Serial.println()`". This is also the best format for sending data to Processing.
Max and puredata prefer to receive numbers. "`Serial.write()`" sends numbers.

The Arduino part: ------- (refer to code example 16 "Serial_Send_Analog_(throw ASCII)")

```
int analogPin = A5;            // Variable for which analog pin to read
int analogValue;               // Variable for storing analog readings

void setup()
{
  Serial.begin(9600);          // start serial port at 9600 bps:
}
void loop()
{
  analogValue = analogRead(analogPin);    // 1st Read the raw 10 bit analog number
                                          // then uncomment, to constrain and map (scale)
// analogValue = constrain(analogValue, 25, 975);
// analogValue = map(analogValue, 25, 975, 0, 255); // scale it to 8 bits

  Serial.println(analogValue);      // Send analogValue to serial (USB) as ASCII
// Serial.write(analogValue);       // Alternatively, Send analogValue as numbers
  delay(10);                        // pause for 10 milliseconds
}
```

The Processing part: ------- (refer to code example 15 "Simple_Processing_Serial_Receive")

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

Serial myPort;           // The serial port
String serialString;     // variable for storing serial data
int val;                 // for storing number re-cast from string
        analogValue = map(analogValue, 0, 1023, 0, 255); // remap (scale) 10 bit to 8 bit
 void setup() {
   size(200, 200);           // Set size of canvas
   printArray(Serial.list()); // -Gives list of available ports

   myPort = new Serial(this, Serial.list()[2], 9600); //-- Change index number to your port
}

  void draw() {
   println(val);
   background(val);
}       //-- End of draw loop


//---- The SerialEvent function runs continuously in the background and
//---- stores new values whenever new data comes in on the serial port (USB)

void serialEvent(Serial myPort) {
   serialString = myPort.readStringUntil(10); // Read until end of line character
   if (serialString != null) {
     serialString = trim(serialString);
     val = int(serialString);
     println(val);
   }                   // end if
}                      // end function definition
```