

Peter Rogers

# Physical Computing workshops

## Intro to coding in Arduino and Processing

The Arduino project is a physical computing platform based on a simple board and a development environment that implements the Processing/Wiring language. Basically you code your program in the arduino app (or IDE) and then upload it to your board.

The arduino board comes in many flavours, there are ordinary boards, mini boards, boards that can be sewn into clothing and washed, boards with extensions so that they can communicate over bluetooth, wireless etc. Arduino is completely open source which means you can download the design for boards and build your own or even design new ones.

## Coding in Arduino

Coding in arduino is very simple to begin with. The general pattern of any program is straightforward, with just two simple functions; the `setup()` and `loop()` functions. Your programs can and probably will include more functions but these two are the only ones needed for arduino to execute.

```
//this is where you declare 'global' variables
int variable = 0;

void setup() {
  //this is where you setup your program
}

void loop() {
  //this is the looping code that will go on playing forever!
}
```

If you have had any experience of processing then arduino is pretty much the same. You start by declaring variables then you create the `setup()` function. In this function you set the board up, this might mean telling arduino to listen to a certain input pin, you can also tell the arduino to create a 'serial' connection to talk to a computer etc. Then you have the `loop()` function which like it says will loop over the code inside it forever. Thats it.

So when an arduino runs your code, first it declares the variables then does the `setup()` function then the `loop()` function.

Coding an arduino is easy, but the better your code will work will be dependent on the elegance and simplicity of your code. Arduino only has limited processing power so if your code is too demanding your code will slow the board down and even crash it. So the more elegant you are the better, which generally means the shorter your code is the better. Ultimately though it is down to how you read the inputs and how you handle that information.

## An Example: Linking Arduino to Processing

In this example we will get arduino to read from one analog sensor and send out that data to processing over the serial connection. We have to write two programs, one for the arduino and one for processing.

Talking over serial is a relatively simple task especially for only one sensor connection, the more sensors used the trickier it becomes, the other main problem is in working out what your program does when data is not sent through, it takes a bit of practice but generally solutions are pretty straightforward.

Here is the Arduino code (you can copy and paste it into arduino):

```
int sensorPin = 0;
int val = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = analogRead(sensorPin);
  sendData(val);
  delay(10);
}

void sendData(int data){
  Serial.println(data);
}
```

The above code has the setup() function, the loop() function and another function called sendData(). In the setup() function there is just one line of code which sets up a serial connection.

The first line in the loop() function reads the sensor on pin 0 and stores it in the val variable. In the second line the sendData(val) function is called. The third line is a delay which stops the program for 10 milliseconds before looping around again.

The sendData() function simply sends out the val variable by doing Serial.println().

## The Processing code

The processing code is a little more complex, mainly because it takes the sensor data from arduino and then uses the data to draw a circle whose size changes with the sensor signal.

To make the flow simple the next example uses line numbers so the important lines can be explained.

```

1 /**
2 Code for reading in data from one sensor on an arduino
3 Pete Rogers 19/12/2008 12.15
4 **/
5
6 import processing.serial.*;
7
8 int lf = 10;           // Linefeed in ASCII
9 int myWidth = 1024;   // stage width
10 int myHeight = 1024; //stage height
11 String sensorString = null; //string holding raw data from arduino
12 int sensorInt = 0;    //number from sensor
13 Serial myPort;       // The serial port
14
15 void setup() {
16     size(myWidth, myHeight); //set up the stage window
17     frameRate(30);           //set frame rate playback
18     smooth();                //make the visual quality high
19 //lines below open and configure the serial port
20     println(Serial.list());
21     myPort = new Serial(this, Serial.list()[1], 9600);
22     myPort.clear();
23     sensorString = myPort.readStringUntil(lf);
24     sensorString = null;
25 }
26
27 void draw() {
28     background(0); //repaints background
29     ellipse(myWidth/2,myHeight/2, sensorInt, sensorInt); //draws circle
30 }
31
32 //serialEvent function is triggered whenever there is data from the arduino
33 void serialEvent(Serial myPort){
34     sensorString = myPort.readStringUntil(lf);
35     if ( sensorString != null) {
36         sensorString = trim( sensorString);
37         sensorInt = int( sensorString);
38     }
39 }

```

If you have ever programmed in processing before then the above code is straightforward, the only bit you may not be used to is the serialEvent() function. Lets take a look...

line 6: we import processing.serial library which provides methods to read serial.

line 8-13: some variables are declared including the serial object myPort

line 15-18: is the setup() function this is where the window is setup and framerate etc

line 20: prints out the various serial ports so you can see which one the arduino is on

line 21: actually starts the serial connection, change the number in list()[1] to get the right serial

line 22-24: clears any rough data out of the serial port before beginning

line 27-29 is the draw function which will just loop and redraw the circle

line 33: is the serialEvent function this is called when something is received in the serial port

line 34: reads all the data in the serial port until a new line is created and puts it in a string

line 35-37: checks that there is something in the string, if yes, it converts it to an integer: sensorInt

The serialEvent() function is really useful, it is an in-built ifunction that means processing only reads

the serial port when it receives data. It is not the only way to read from the serial port but it is the most efficient.

Below is the actual code (copy and paste to processing)

```
/**
Code for reading in data from one sensor on an arduino
Pete Rogers 19/12/2008 12.15
**/

import processing.serial.*;

int lf = 10;           // Linefeed in ASCII
int myWidth = 1024;   // stage width
int myHeight = 1024; //stage height
String sensorString = null; //string holding raw data from arduino
int sensorInt = 0;    //number from sensor
Serial myPort;       // The serial port

void setup() {
  size(myWidth, myHeight); //set up the stage window
  frameRate(30);           //set frame rate playback
  smooth();                //make the visual quality high
  //lines below open and configure the serial port
  println(Serial.list());
  myPort = new Serial(this, Serial.list()[1], 9600);
  myPort.clear();
  sensorString = myPort.readStringUntil(lf);
  sensorString = null;
}

void draw() {
  background(0); //repaints background
  ellipse(myWidth/2,myHeight/2, sensorInt, sensorInt); //draws circle
}

//serialEvent function is triggered whenever there is data from the arduino
void serialEvent(Serial myPort){
  sensorString = myPort.readStringUntil(lf);
  if ( sensorString != null) {
    sensorString = trim( sensorString);
    sensorInt = int( sensorString);
  }
}
```

## Smoothing the signal from the arduino

Dependent on your sensor you might find that the signal is 'jumpy', this can be common for quite a few different reasons, it is called noise and a common way to deal with it is to smooth it out.

the Arduino code below is one way to smooth out a signal. The code is based on the previous code with changes in green

Copy and paste code to arduino

```
int sensorPin = 0;
int val = 0;
int smoothed;
int alpha = 200;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = analogRead(sensorPin);
  sendData(val);
  delay(10);
}

void sendData(int data){
  if (data > smoothed){
    smoothed = smoothed + (data - smoothed)/alpha;
  }else{
    smoothed = smoothed - (smoothed - data)/alpha;
  }
  if (smoothed < alpha){
    smoothed = smoothed - 1;
    if (smoothed < 0){
      smoothed = 0;
    }
  }
  data = smoothed;
  Serial.println(data);
}
```

Changing the variable alpha will change the level of smoothness. The higher the number the smoother it will be.