

SENSING FOR VISUALIZATION



+



A Wiring and Processing
workshop to read and
visualize data



SENSING FOR VISUALIZATION

A Wiring and Processing workshop
to read and visualize data

Sensing for Visualization is an introduction to analog sensors and interactive graphics visualization. It includes circuit designs and programs to get started capturing data from the outside world. Reading data from our environment can be useful to create sensible interfaces that respond to our presence.

The analogue sensors covered in this workshop will be: photocell, potentiometer, switch and accelerometer. These sensors respond to light variations, touch, movement and orientation. Using inexpensive microcontrollers (Wiring/Arduino), electronic components and open software, you can prototype circuits to read meaningful data from the environment.



1.Functions

Before we start using the Wiring analog inputs, lets review some of the programming elements. In Processing, the syntax is almost identical to Java, but processing adds special features related to **graphics** and **interaction**.

1. Download the Processing IDE visiting

www.processing.org/download

At the bottom of the page, choose to download

Processing 3



2. Install program and open it

You can type commands to easily draw images on the computer screen by using predefined graphic functions such as `ellipse()` and `rect()`. Each function is conformed by its name and parameters. This processing function below draws an ellipse on the screen using the points (50,50) as center and a dimension of 80 by 80.

name of function



```
ellipse (50, 50, 80, 80);
```



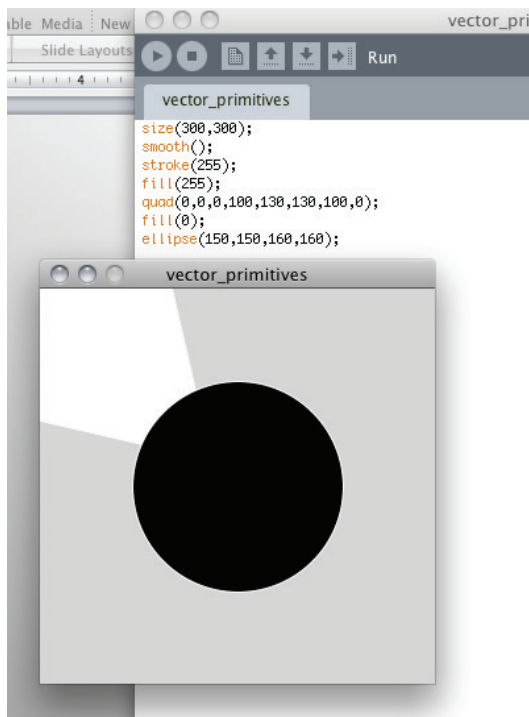
parameters

You don't need to memorize all the processing functions, but you will become familiar as you use them. The most helpful part is that you can find the meaning of the specific syntax in the Processing's reference (<http://processing.org/reference/>). From the reference we learn that:

ellipse(x, y, width, height);

2. Variables

The parameters from our functions can be changing numbers instead of fixed ones. These results can be used to create interactive systems that respond to a user's input. Taking in consideration that each parameter can be a variable that changes in time, we can alter graphics characteristics such as the size, the position, the color among many other properties.



Color variables (values range from 0-255)

tint(R,G,B,Alpha);
fill(R,G,B,Alpha);

3. Wiring/Arduino + Processing

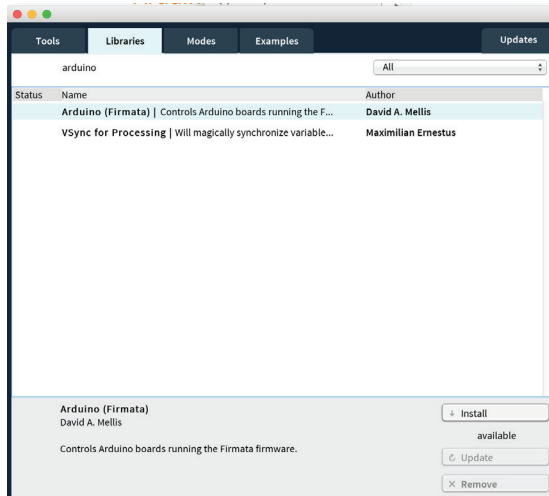
You will need to install the software that allows the computer to communicate with your Arduino/wiring board. To allow your Arduino Mega board read variable data in processing, you must first complete the following steps:

1. Download the Arduino IDE visiting www.arduino.cc/download
2. Install the Arduino.exe file. This will include the drivers and software
3. Run the Arduino IDE for the first time and plug in your board to the USB port
4. In the tool bar go to: file / examples/ Firmata/ AllinputsFirmata
5. Chose your board (Arduino Mega 2560)and Serial port (should be the one starting with /dev/.../usb...)
6. Click on the “Upload” button

A screenshot of the Arduino IDE interface. The title bar reads 'AllInputsFirmata | Arduino 1.6.5'. The toolbar at the top includes icons for a checkmark, a plus sign, a document, an upload arrow, a download arrow, and a 'Verify' button. The code editor displays the 'AllInputsFirmata' sketch. The code includes variable declarations for analog values, port status, and timer variables, along with comments explaining the code's purpose. The status bar at the bottom indicates the selected board is 'Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560)' and the serial port is '/dev/cu.usbmodem1411'.

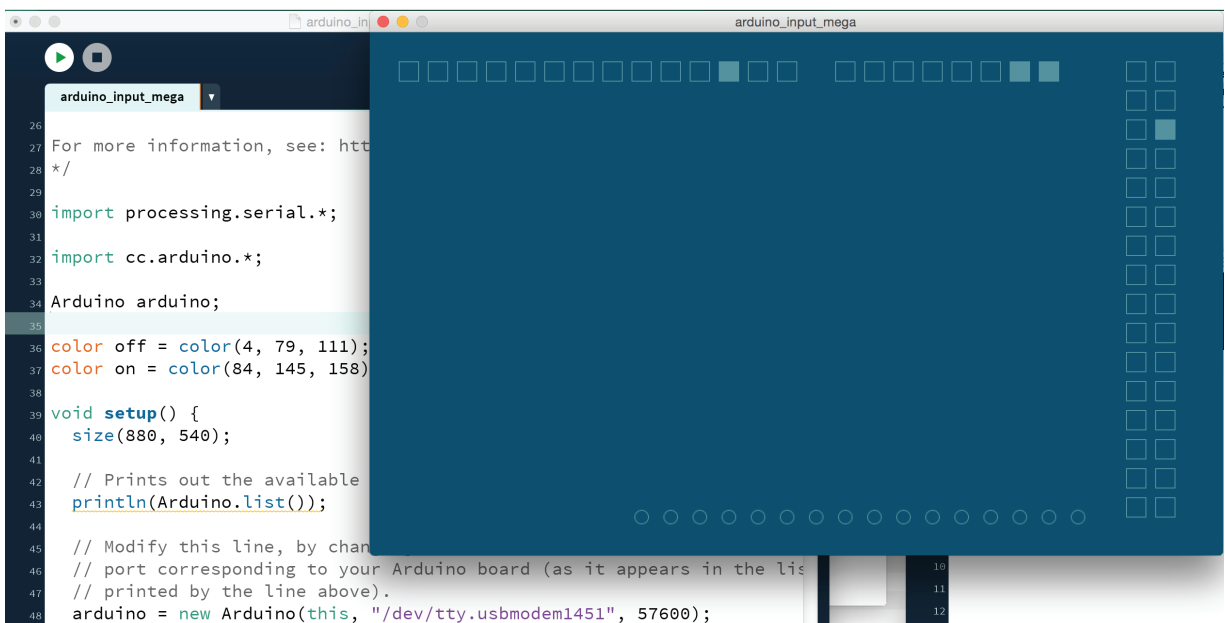
Now let's run Processing 3

1. Go to the menu Sketch/import Library/Add Library and choose “Arduino”

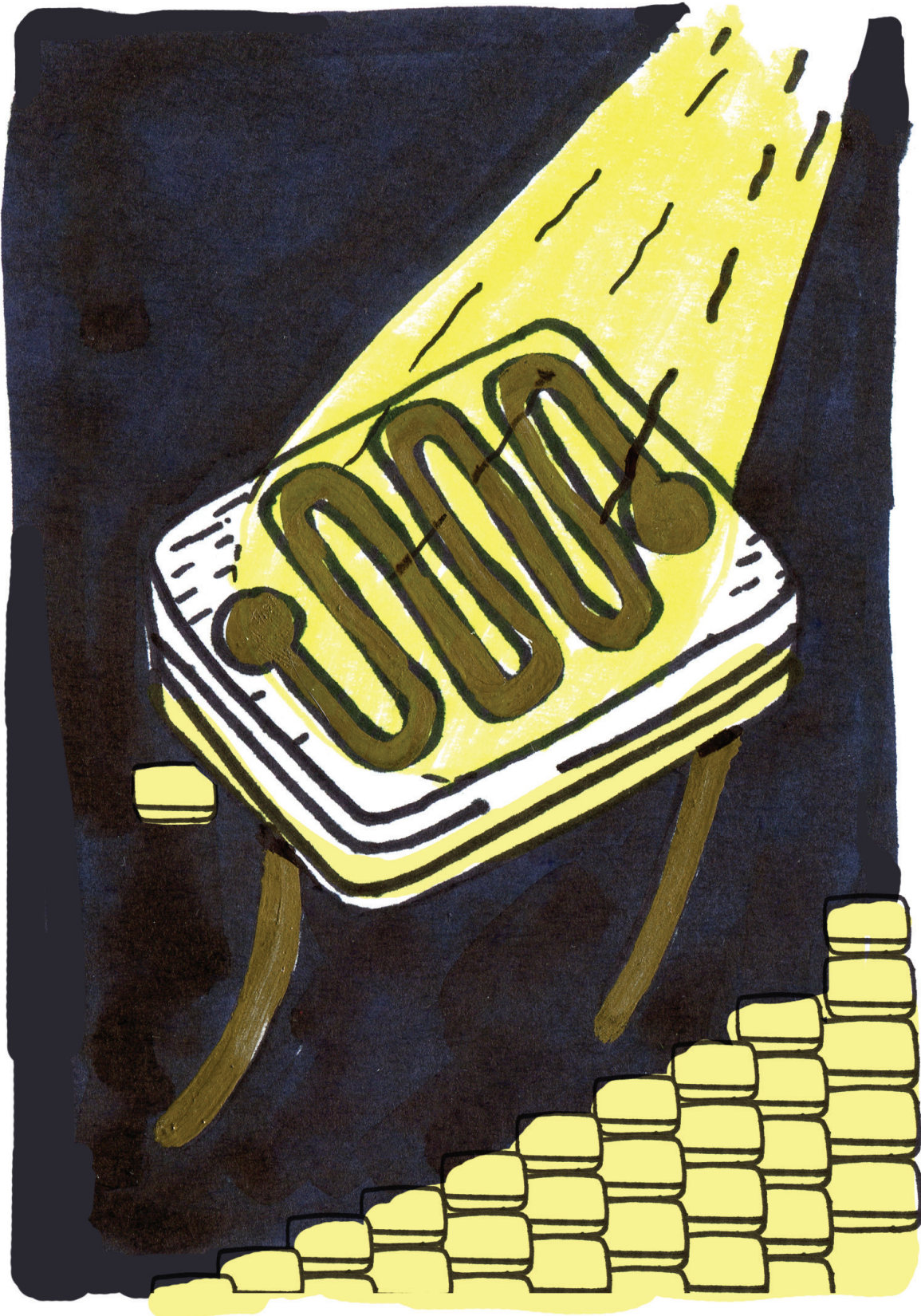


← click on “Install”

2. Run the application by pressing the “Play Button”



4. Reading Analog Data



Photocell

Analog Sensors

Depending on your choice to interface with the environment, you can use different sensors. The photocell is sensitive to light, the potentiometer is a knob, and the accelerometer responds to its inclination. In all these cases, the user is manipulating data, and the data can be meaningful for creating user experiences. We assume that environment variables can be also computer variables or parameters, and therefore used to manipulate values in the screen.

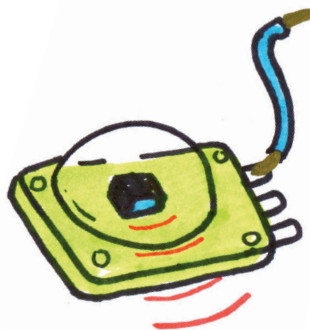
Every sensor has a different behavior, but when we read physical data in the computer we will have numbers. These numbers can be used as parameters of a computer program. The numeric values that the computer can read is an interval between 0 – 1023. We will use this interval as a range of variability.



Button Switch



Potentiometer



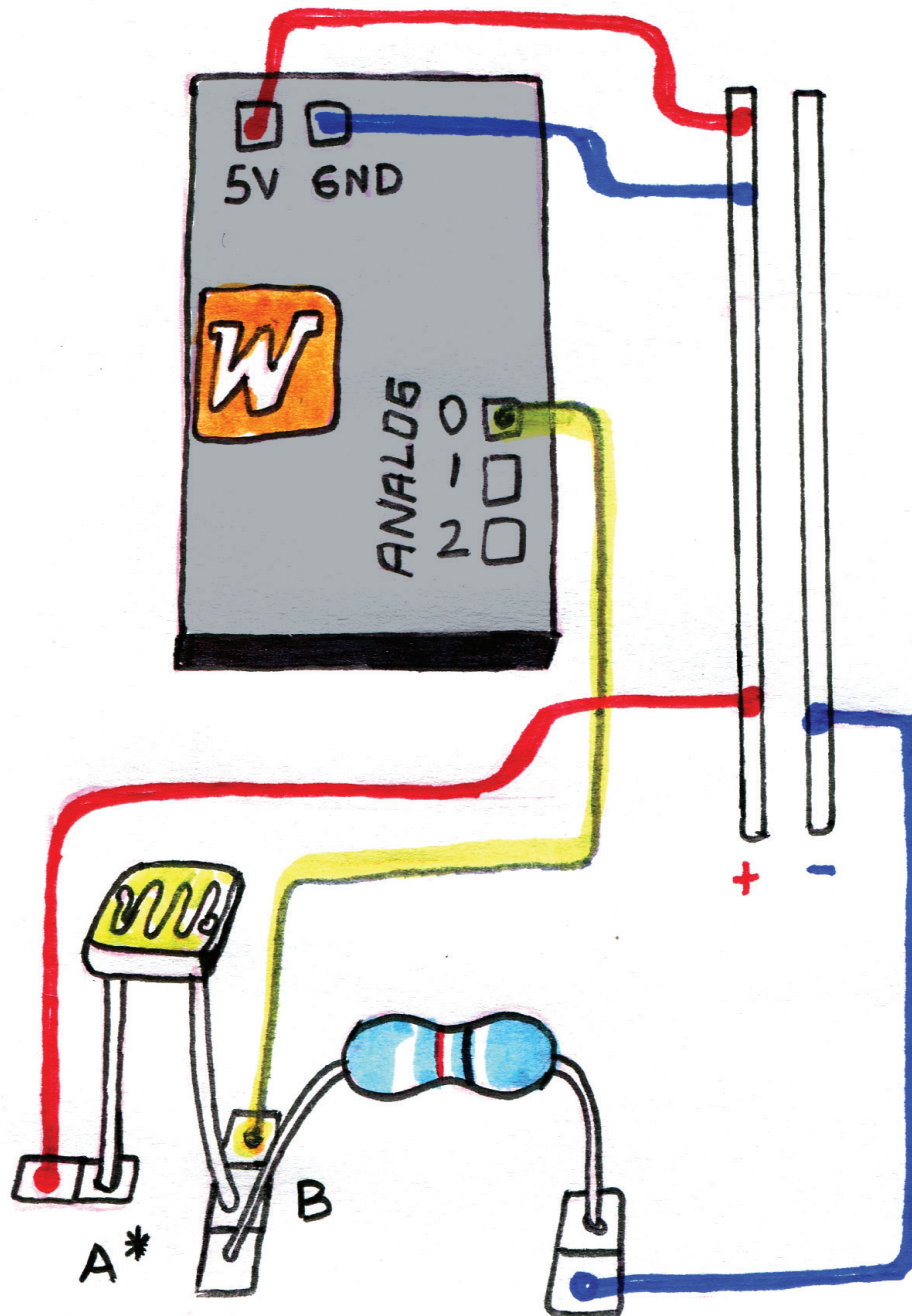
PIR motion sensor



Jumper wires

5. Visualizing one variable

Build the circuit below:

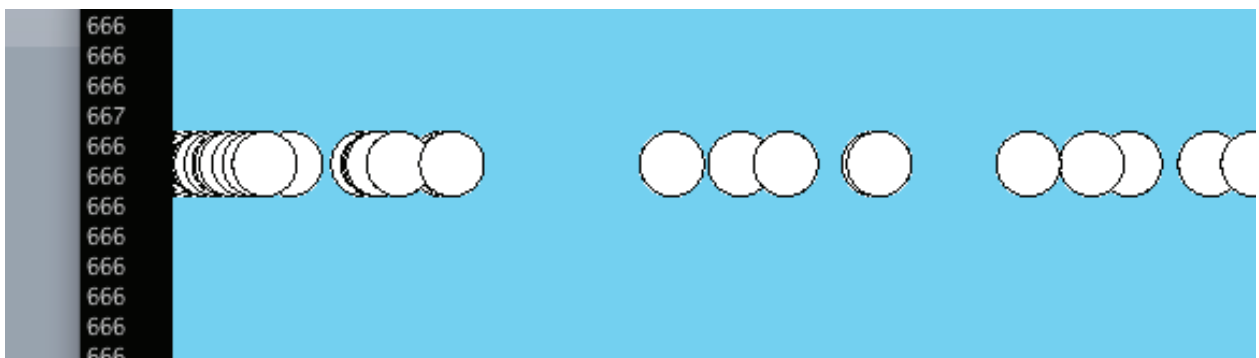


* You may replace the connection A-B with a potentiometer or a switch

Copy and paste the following code in processing:

```
1 import processing.serial.*;
2 import cc.arduino.*;
3
4 Arduino arduino;
5 int val;
6
7 void setup() {
8
9     size(720,500);
10    frameRate(10);
11
12    println(Arduino.list());
13    arduino = new Arduino ( this, "/dev/tty.usbmodem1451", 57600);
14    arduino.pinMode(4, arduino.INPUT);
15 }
16
17 void draw(){
18     background(0);
19     val=arduino.analogRead(4);
20     ellipse(val,250, 10,10);
21 }
```

The variable “val” is the numeric value captured by the photocell. The function *wiring.analogRead(4)* means that wiring reads the analog port #4 on your board. The function *println()*; allows us to see the true numeric value of “val.” Now the ellipse moves along the x-axis of the screen because of *ellipse(val,250, 10,10)*;



6. Calibration

Using the function `map()`; we can adjust the range of the maximum and minimum values of our sensor. Depending on the context it won't always be 0-1023.

Output range desired

$value = map(val, 0, 1023, 0, width);$

Input range from analog read

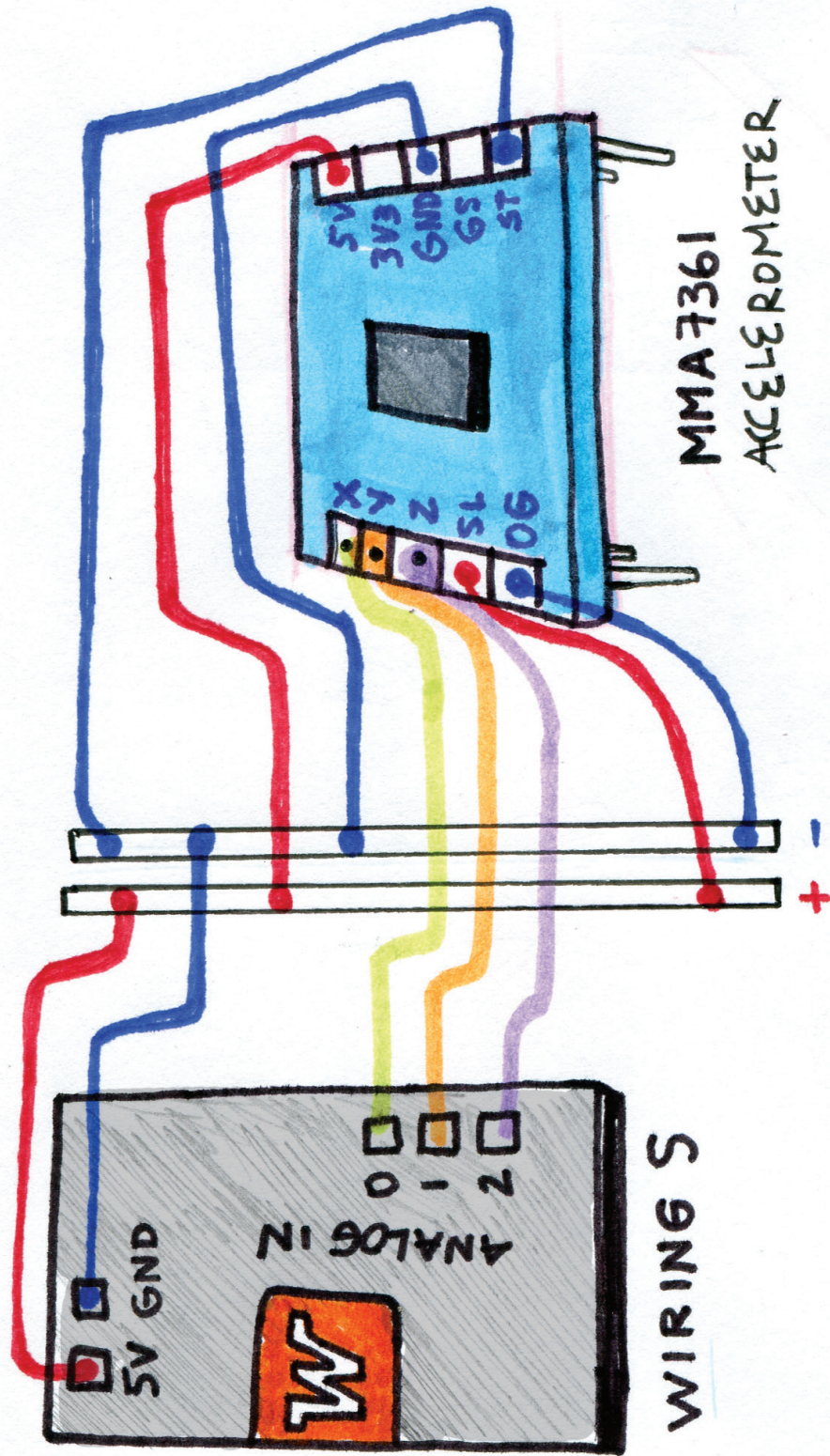
The desired range can be the size of the screen for example, therefore we use the variable `width`. The new code with the `map()` function added will be:

```
1 import processing.serial.*;
2 import cc.arduino.*;
3
4 Arduino arduino;
5 int val;
6 float value;
7
8 void setup() {
9
10     size(720,500);
11     frameRate(10);
12
13     println(Arduino.list());
14     arduino = new Arduino ( this, "/dev/tty.usbmodem1451", 57600);
15     arduino.pinMode(0, arduino.INPUT);
16 }
17
18
19 void draw(){
20     background(val*0.2);
21     val=arduino.analogRead(4);
22     value=map(val, 0, 1023, 0, width);
23     ellipse(value,250, 10,10);
24     textSize(32);
25     fill(255);
26     text(val,100,100);
27
28 }
```



7. Accelerometer

Can you create a program that reads 3 variables at the same time? How would you alter your visualization? Use the same code adding new variables.

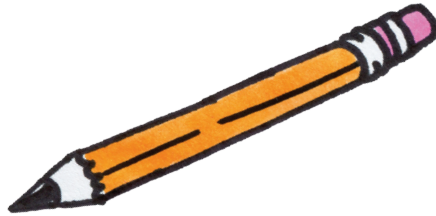


8. Programming Objects

Another of the element that may play importance while using graphics is the concept of a programming “object.” A programming object is related to real objects in the sense that they have specific characteristics.

Say a pencil...

- Is yellow
- It's 7 inches
- Has an eraser



In the same way, every “object” in programming has different properties. You can change the size, position, hue and many other properties by applying variables to each individual object.

In programming we can “call” the characteristics (or properties) of the objects and use them as variables. Objects can be movies, bit-maps vectors, sounds, etc. For raster graphics, we use the PImage object:

PImage b; //creates the object

b = loadImage(“mar.png”); // assigns a file to that image object

image(b, 0, 0); // the function image draws “b” in the screen point (0,0)

We can also explore more in the reference to learn about the properties of the image object. In the case of the image, the syntax explains:

image(img, x, y, width, height, opacity)

To access the object’s properties we use “.” To refer to that object for example the width of image b is noted as “*b.width*”

Experiment in processing bringing images and modifying their properties using the following code. Remember to save image in the “data” folder of the same sketch.

PImage a; //creates the object a

```
void setup() {  
  size(800, 600);  
  a = loadImage("name of your image.png");  
}
```

```
void draw() {  
  image(a, 400, 300, a.width*val, a.height*val);  
}
```

Can you change the size of an image using the potentiometer or accelerometer?

The expression *a.width*val* takes the original width of the image and multiplies the size by the value of the variable “val”



Sensing for Visualization:
A Wiring and Processing work-
shop to read and
visualize data

BPVIZ

BROADENING & PARTICIPATION IN VISUALIZATION

Created by Esteban García Bravo
Revised in 2016 for BPViz



Acknowledgements

Thanks to Hernando Barragán for his work developing Wiring. He revolutionized the practice of electronic art and made physical computing more approachable. Wiring-based hardware has become ubiquitous for hobbyists and engineers alike. To learn more about the history of Arduino and Wiring visit: <https://arduinohistory.github.io>

At school in Colombia, Hernando taught us that programming is a lot of fun. The Wiring website includes extensive documentation and examples of this project. For more information visit www.wiring.org.co

Wiring is © 2011 Hernando Barragán, Brett Hagman and Alexander Brevig.

