

## Arduino/Processing Communication

First, we need to load Firmata on the Arduino Board. Firmata is a special Firmware for the Arduino board which allows us to fully control the board through Processing.

In Arduino open:

File > Examples > Firmata > Standard Firmata

Load the code on the Arduino board. If successfully uploaded you won't need to use the Arduino software anymore!

## Processing

Processing was developed in 2001 by Casey Reas and Ben Fry at MIT as a computational sketching environment. In the meantime it has developed into a powerful software development tool for a variety of multimedia and interactive content. More information at: <http://processing.org/>

Let's look at the anatomy of a simple Processing sketch (code written in Processing), open:

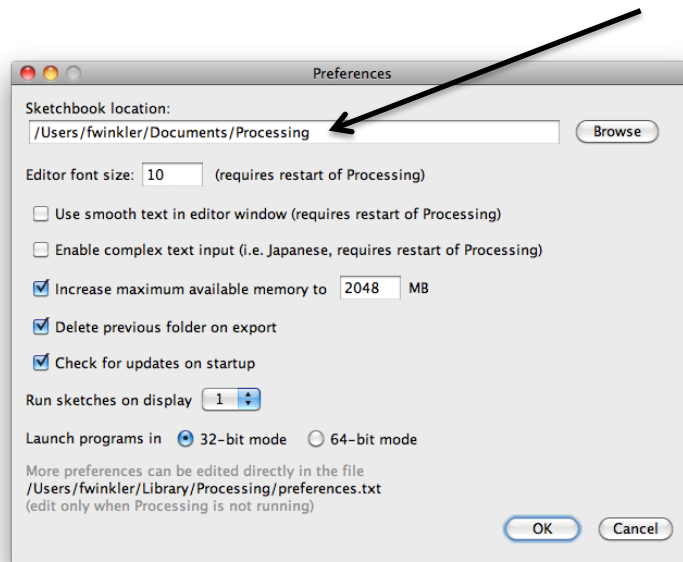
File > Examples > Topics > Drawing > Continuous Lines

## Installing the Library for Processing/Arduino Communication

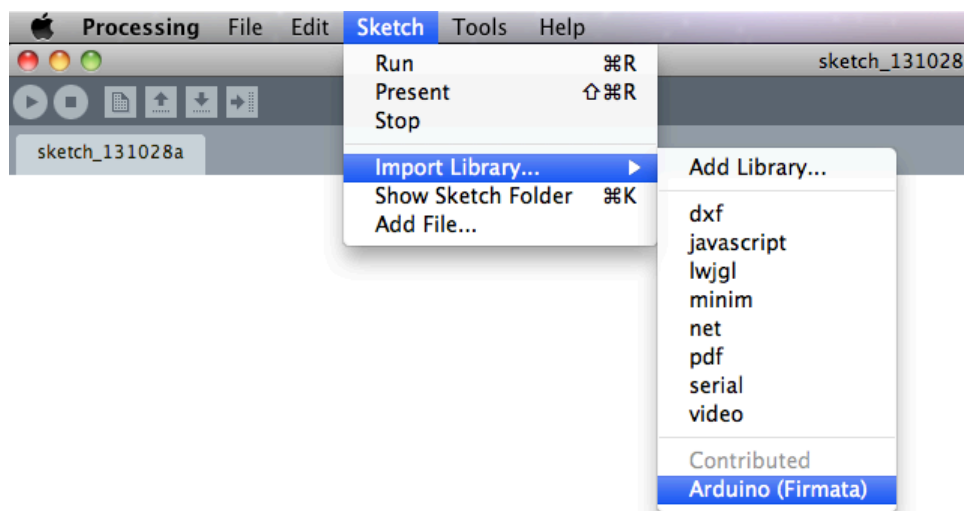
Download the Arduino library for Processing:

<http://playground.arduino.cc/interfacing/processing>

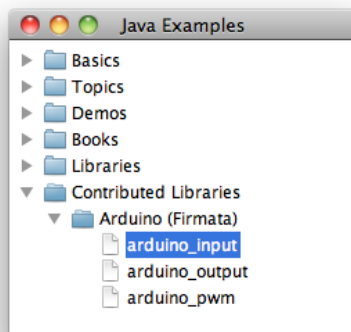
Unzip the library and copy the "arduino" folder into the "libraries" sub-folder of your Processing Sketchbook. (You can find the location of your Sketchbook by opening the Processing Preferences. If you haven't made a "libraries" sub-folder in your Sketchbook folder, create one.)



Restart Processing, after a successful installation you should now see the library in your libraries menu:

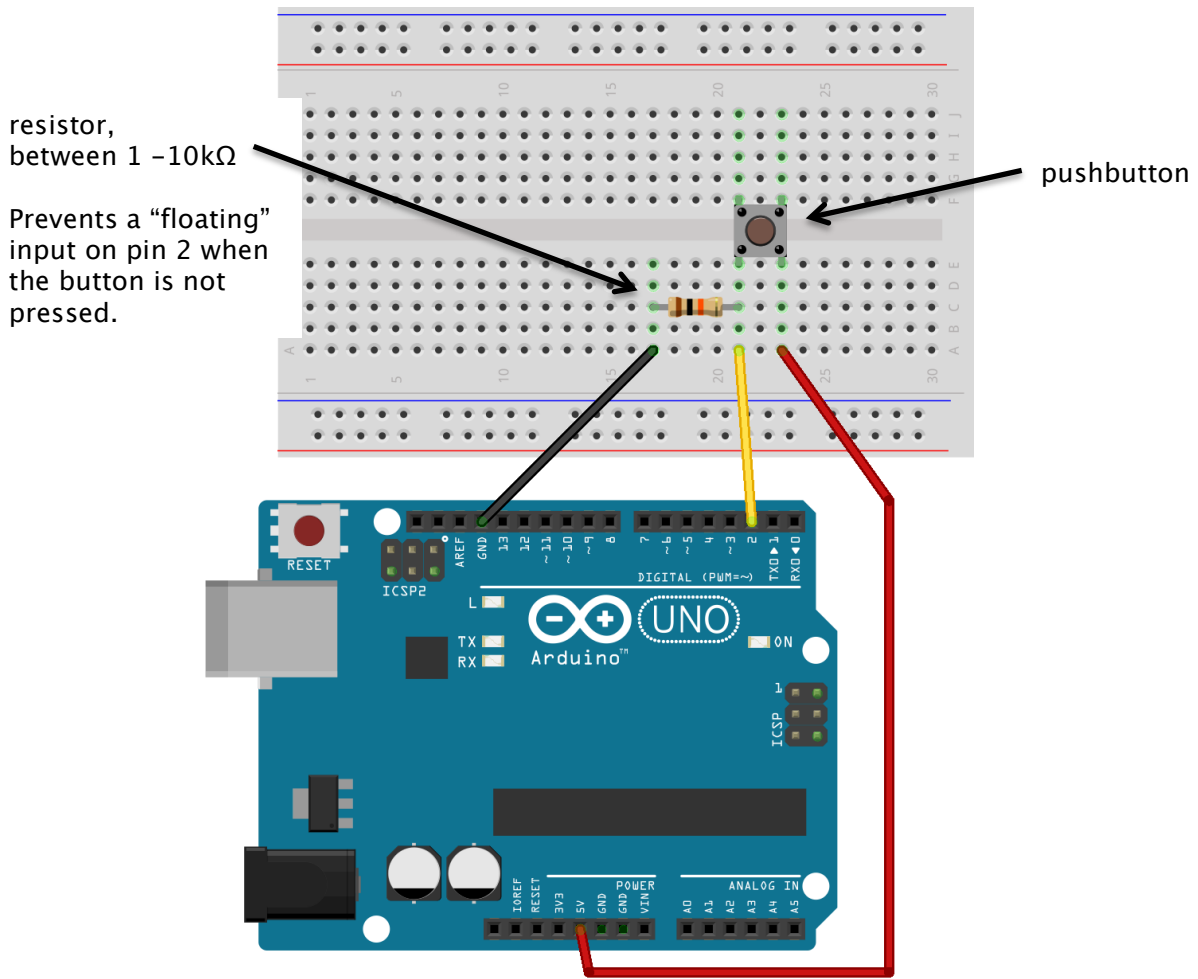


Now open the "arduino\_input" example from File > Examples > Contributed Libraries > Arduino (Firmata) > arduino\_input.



Connect your Arduino board (with Firmata installed) and the pushbutton circuitry attached and run your “arduino\_input” Processing sketch.

This is the hardware setup for the Arduino:



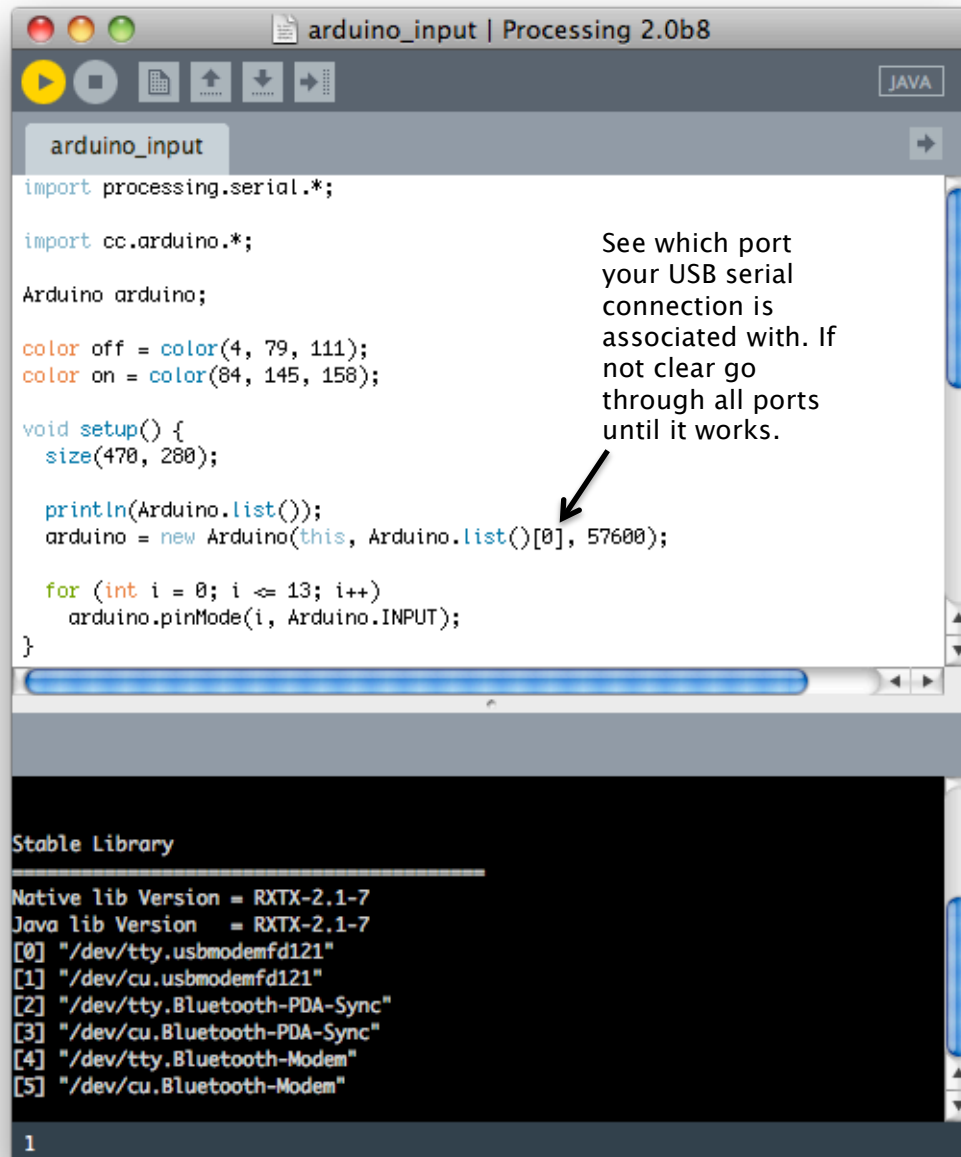
The example might not work initially – check your serial port settings in the “arduino\_input” Processing sketch. On my computer, I got lucky and default serial port “[0]” matched with my usb port (“[1]” would have worked as well) but this might be vastly different on your computer. Check the debugging window in Processing to see all your ports and their associated number.

[0] and [1] have valid USB serial port addresses.



```
Stable Library
Native lib Version = RXTX-2,1-7
Java lib Version = RXTX-2,1-7
[0] "/dev/tty.usbmodemfd121"
[1] "/dev/cu.usbmodemfd121"
[2] "/dev/tty.Bluetooth-PDA-Sync"
[3] "/dev/cu.Bluetooth-PDA-Sync"
[4] "/dev/tty.Bluetooth-Modem"
[5] "/dev/cu.Bluetooth-Modem"
```

This is where you might change the port number in your code:



```
arduino_input | Processing 2.0b8
arduino_input
import processing.serial.*;

import cc.arduino.*;

Arduino arduino;

color off = color(4, 79, 111);
color on = color(84, 145, 158);

void setup() {
  size(470, 280);

  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);

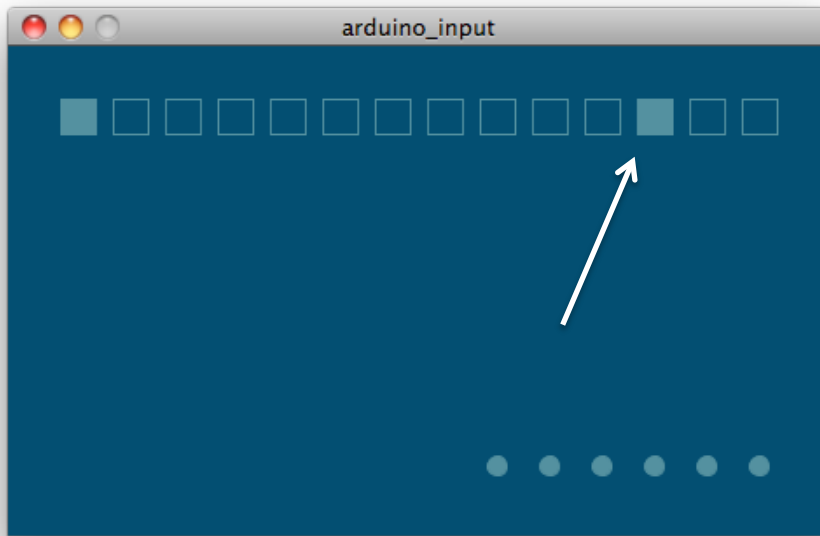
  for (int i = 0; i <= 13; i++)
    arduino.pinMode(i, Arduino.INPUT);
}

Stable Library
Native lib Version = RXTX-2.1-7
Java lib Version = RXTX-2.1-7
[0] "/dev/tty.usbmodemfd121"
[1] "/dev/cu.usbmodemfd121"
[2] "/dev/tty.Bluetooth-PDA-Sync"
[3] "/dev/cu.Bluetooth-PDA-Sync"
[4] "/dev/tty.Bluetooth-Modem"
[5] "/dev/cu.Bluetooth-Modem"

1
```

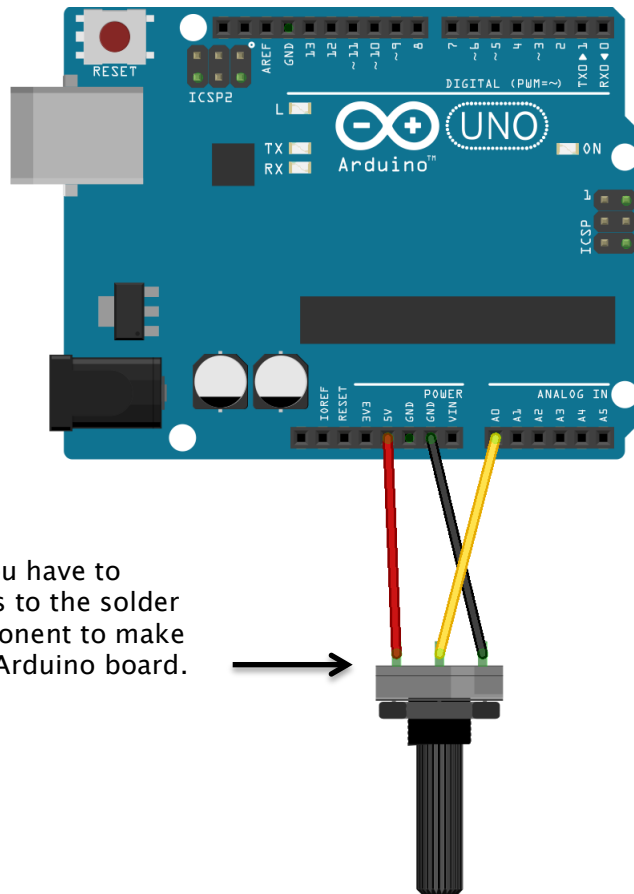
See which port your USB serial connection is associated with. If not clear go through all ports until it works.

Finally, you should see a window like the one below, when pressing the button connected to pin 2 of your board the third box from the right should appear filled with white. You might see some other boxes also change their background color but this is from all the other pins being “floating” pins (i.e. not tied to ground) so they are susceptible to subtle electric charges from your skin or objects close by. IN the next round of examples we won’t even activate those pins so we don’t have to worry about them.



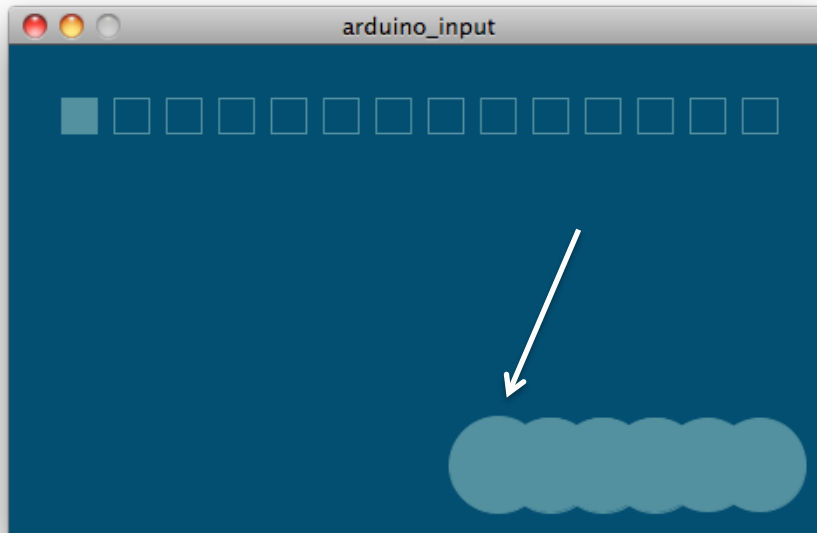
### Testing the Analog Sensor

Now connect the potentiometer to the Arduino pin A0, like this:



Potentiometer, you have to solder some wires to the solder lugs on the component to make it connect to the Arduino board.

Then run the “arduion\_input” sketch again and observe the changing size of the circle representing this pin (again some other circles might change their size as well but we’ll disable them later in our actual code):



OK, you are now all set to start working with Arduino and Processing!