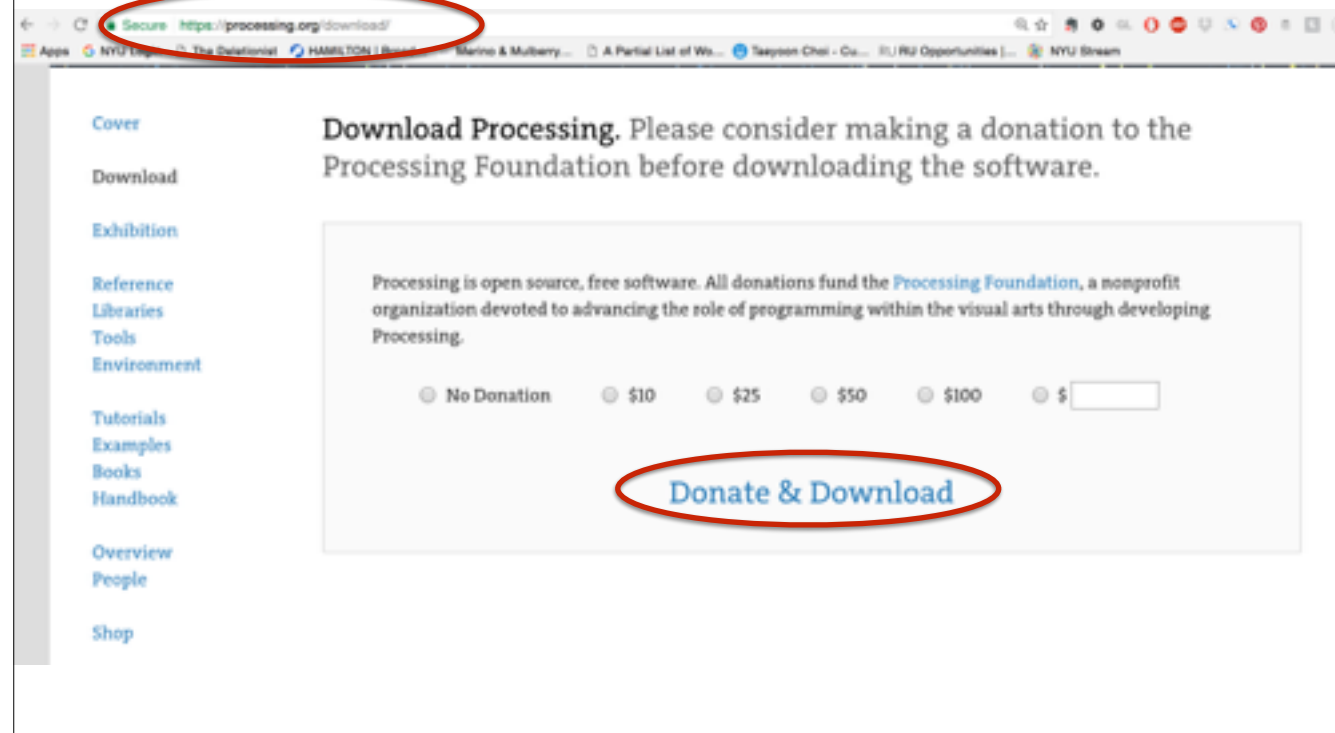


# Arduino: Processing & P5.js

# Arduino + Processing



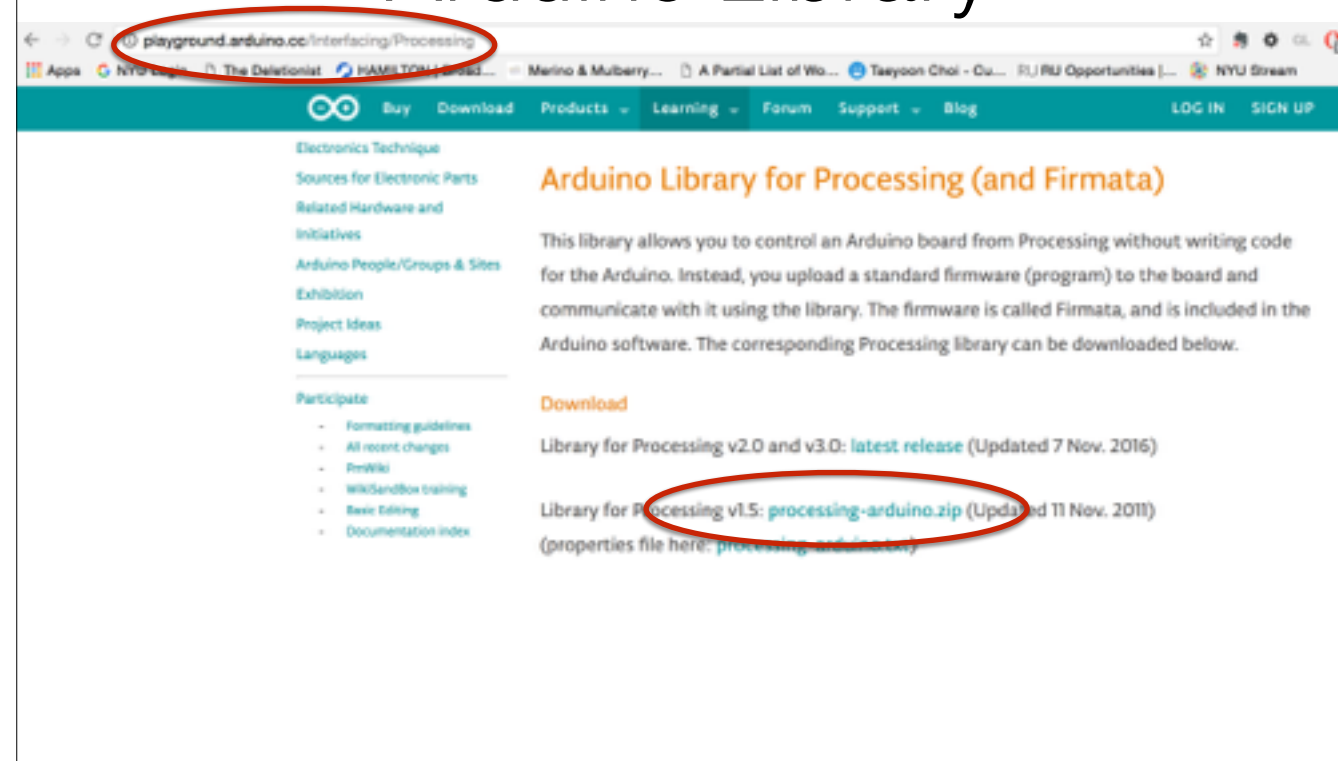
Download Processing for your laptop and Install it

# Using the Serial Library

- Import Serial library
- note port name in Serial object
- read and write to the port

Another method is to use the Processing  
Library for the Arduino within Processing

# Download the Processing-Arduino Library



The screenshot shows the Arduino Playground website at the URL [playground.arduino.cc/interfacing/Processing](https://playground.arduino.cc/interfacing/Processing). The page features a teal navigation bar with links for Buy, Download, Products, Learning, Forum, Support, and Blog. A left sidebar lists various resources like Electronics Technique, Sources for Electronic Parts, and Participate. The main content area is titled "Arduino Library for Processing (and Firmata)" and explains that the library allows controlling an Arduino board from Processing without writing code. It provides two download options: "Library for Processing v2.0 and v3.0: latest release (Updated 7 Nov. 2016)" and "Library for Processing v1.5: [processing-arduino.zip](#) (Updated 11 Nov. 2011)". The link to the v1.5 zip file is circled in red.

playground.arduino.cc/interfacing/Processing

Buy Download Products Learning Forum Support Blog LOG IN SIGN UP

Electronics Technique  
Sources for Electronic Parts  
Related Hardware and Initiatives  
Arduino People/Groups & Sites  
Exhibition  
Project Ideas  
Languages

Participate

- Formatting guidelines
- All recent changes
- PinWiki
- Wiki Sandbox training
- Basic Editing
- Documentation index

## Arduino Library for Processing (and Firmata)

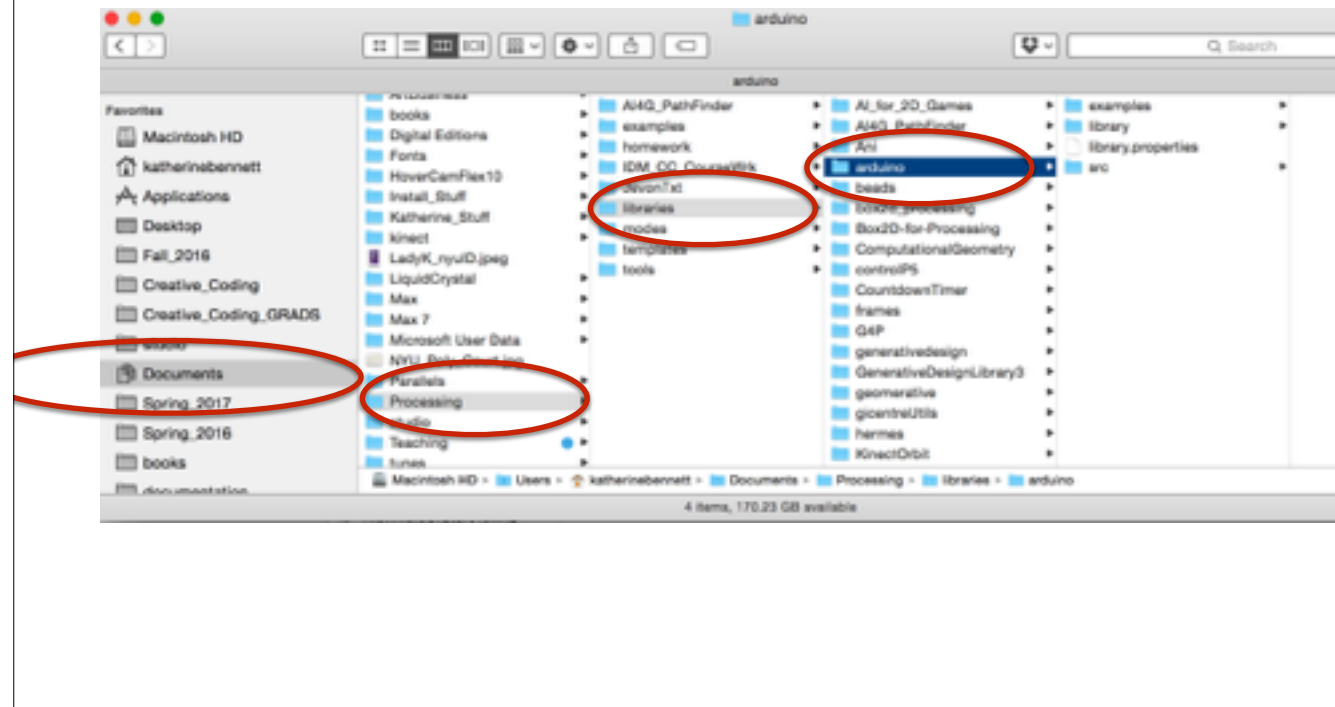
This library allows you to control an Arduino board from Processing without writing code for the Arduino. Instead, you upload a standard firmware (program) to the board and communicate with it using the library. The firmware is called Firmata, and is included in the Arduino software. The corresponding Processing library can be downloaded below.

### Download

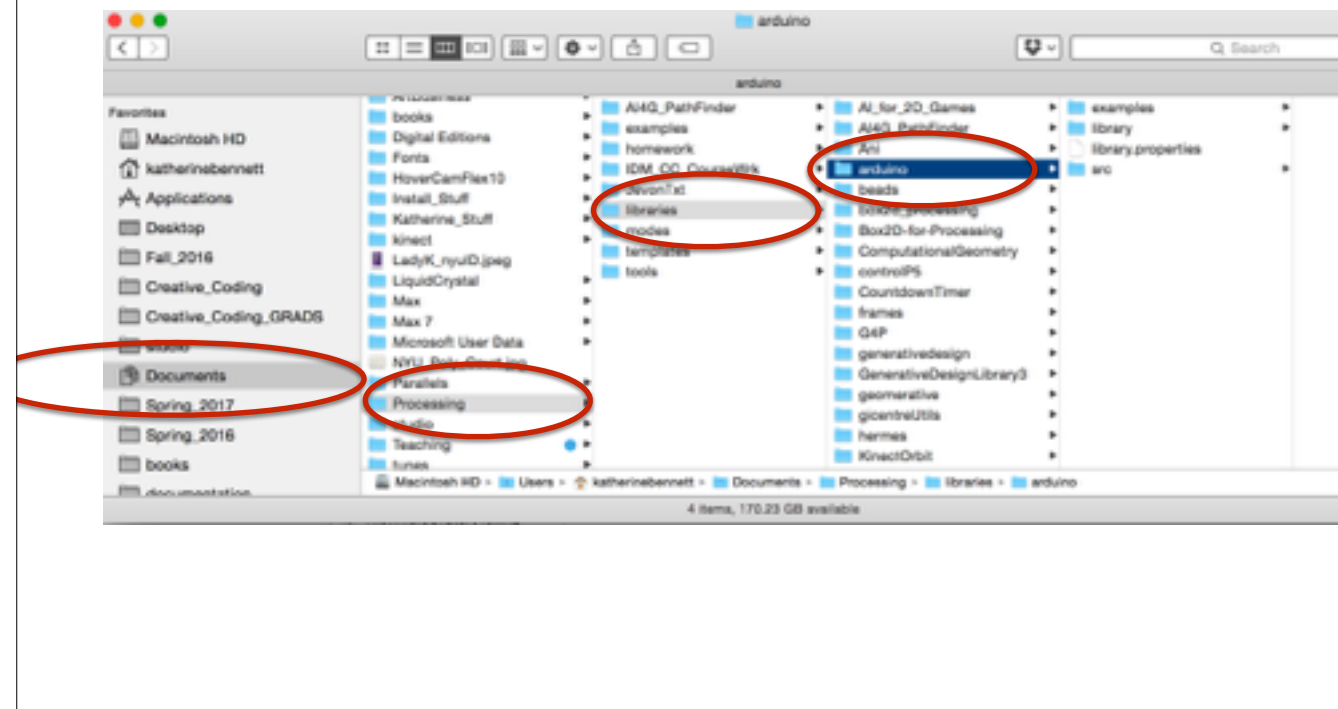
Library for Processing v2.0 and v3.0: [latest release](#) (Updated 7 Nov. 2016)

Library for Processing v1.5: [processing-arduino.zip](#) (Updated 11 Nov. 2011)  
(properties file here: [processing-arduino.txt](#))

Now that you've downloaded Processing + installed it, look in your Documents folder or in your "c" drive for a folder called "Processing"



If you don't have a libraries folder, make one and drop the “arduino” file from the Processing-Arduino download inside of it.

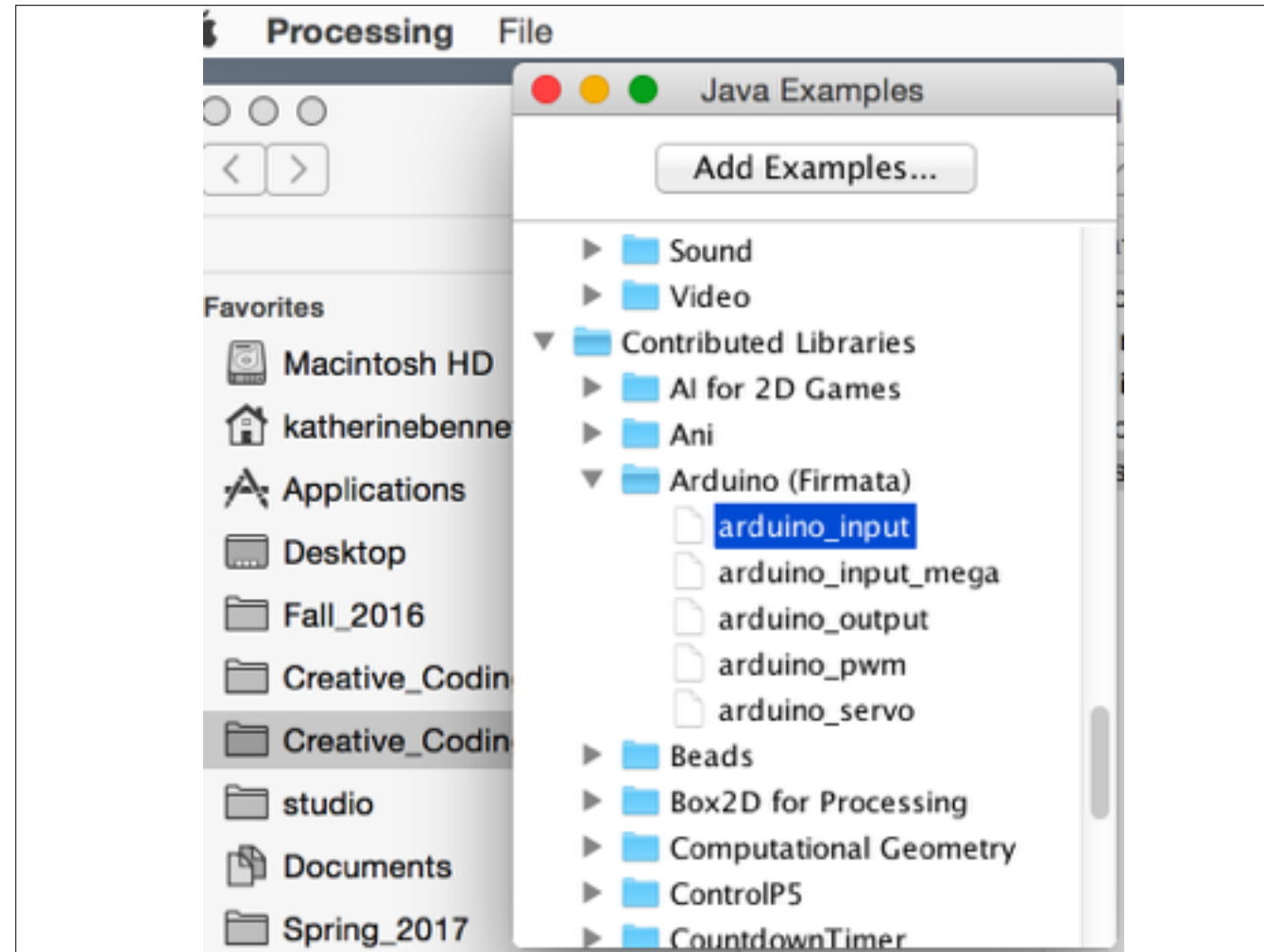


- With the Firmata sketch already loaded onto the Arduino board, we are ready to switch to Processing.
- If the Firmata sketch is not on your Arduino, load it on there from the File/Examples/Firmata/StandardFirmata
- Close the Arduino Application!!!!



Open up the examples that came with the  
arduino library for the Processing  
(Application)





\

- Edit the example code to select the serial port used by Arduino. Specifically, change the [0] in this line
- `arduino = new Arduino(this, Arduino.list()[0], 57600);`
- To find the correct item in the array, run this code in Processing:
- `import processing.serial.*;`
- `import cc.arduino.*;`
- `println(Arduino.list());`
- The output window will enumerate your serial ports. Select the number corresponding to the serial port in your Arduino environment found under Tools > Serial Port.

Arduino:  
P5.js

# Download the p5 libraries

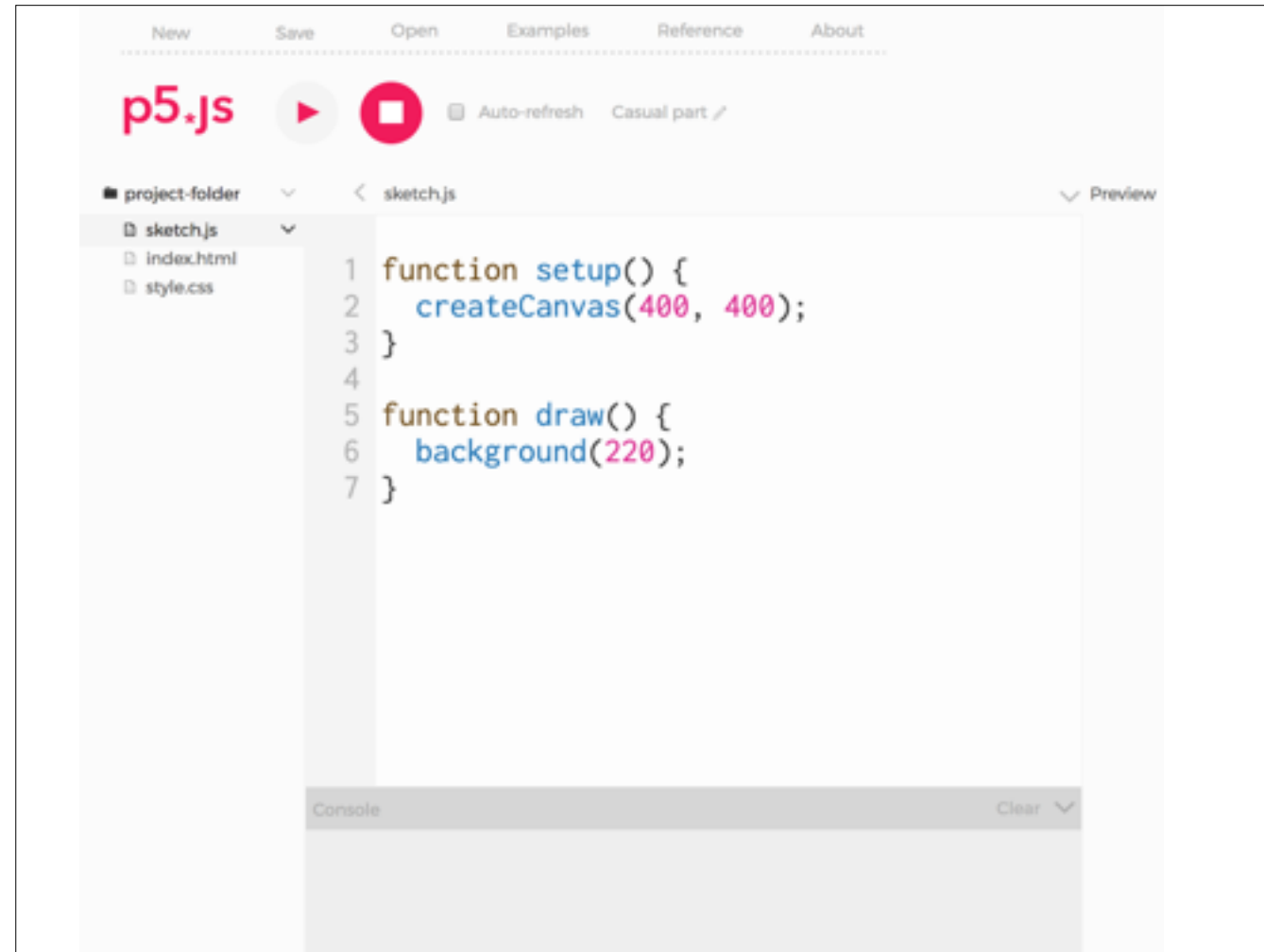
- <https://github.com/vanevery/p5.serialport>
- <https://github.com/vanevery/p5.serialcontrol/releases>

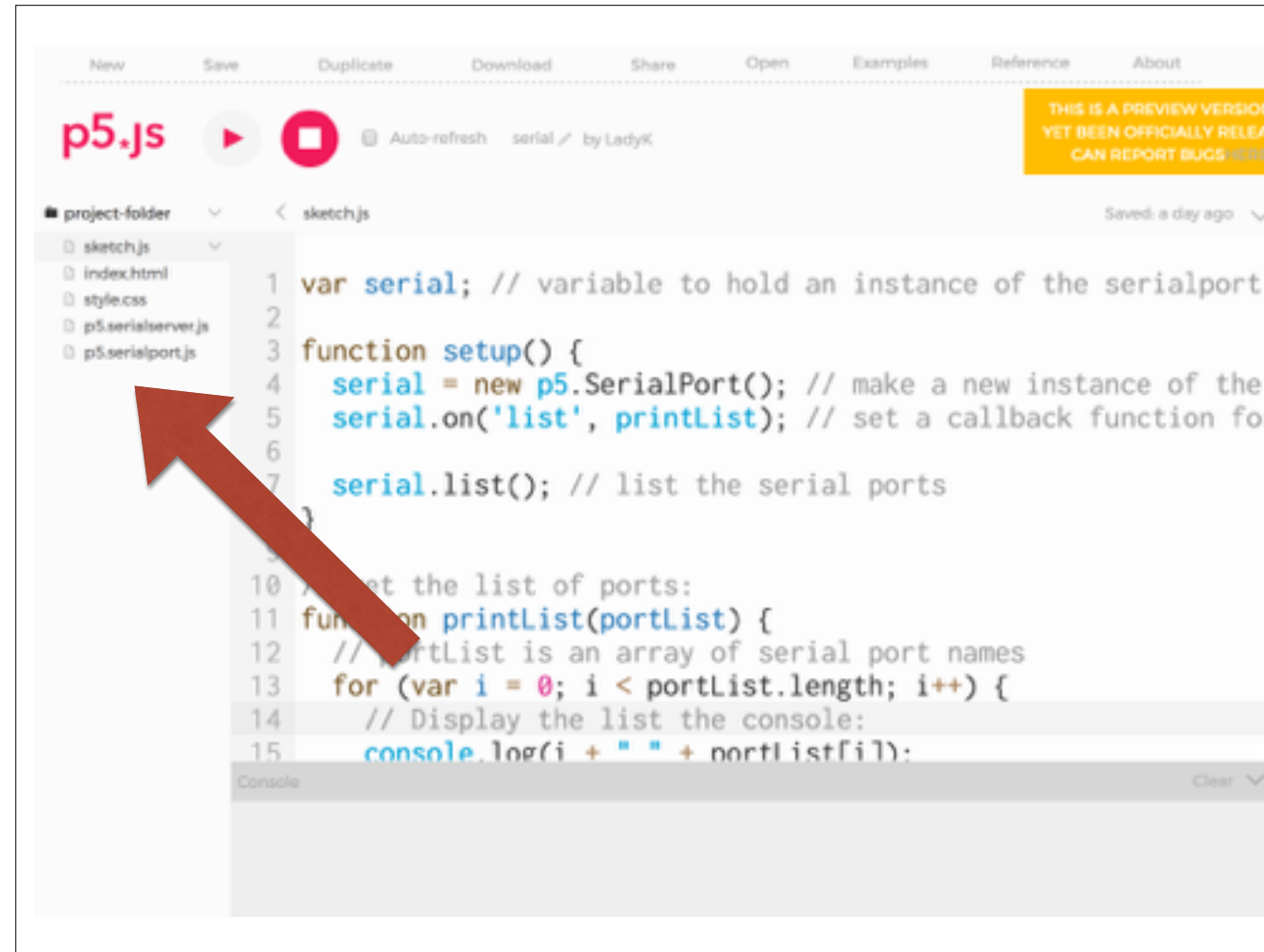


Need 2 things: the library in your p5 project folder + an internal server running (either with node.js installed + node startserver.js from the command line OR the p5.SerialControl (stand alone) application. We are going to use the p5.SerialControl app in conjunction with our p5 sketches

# Start the p5 Serial Control App





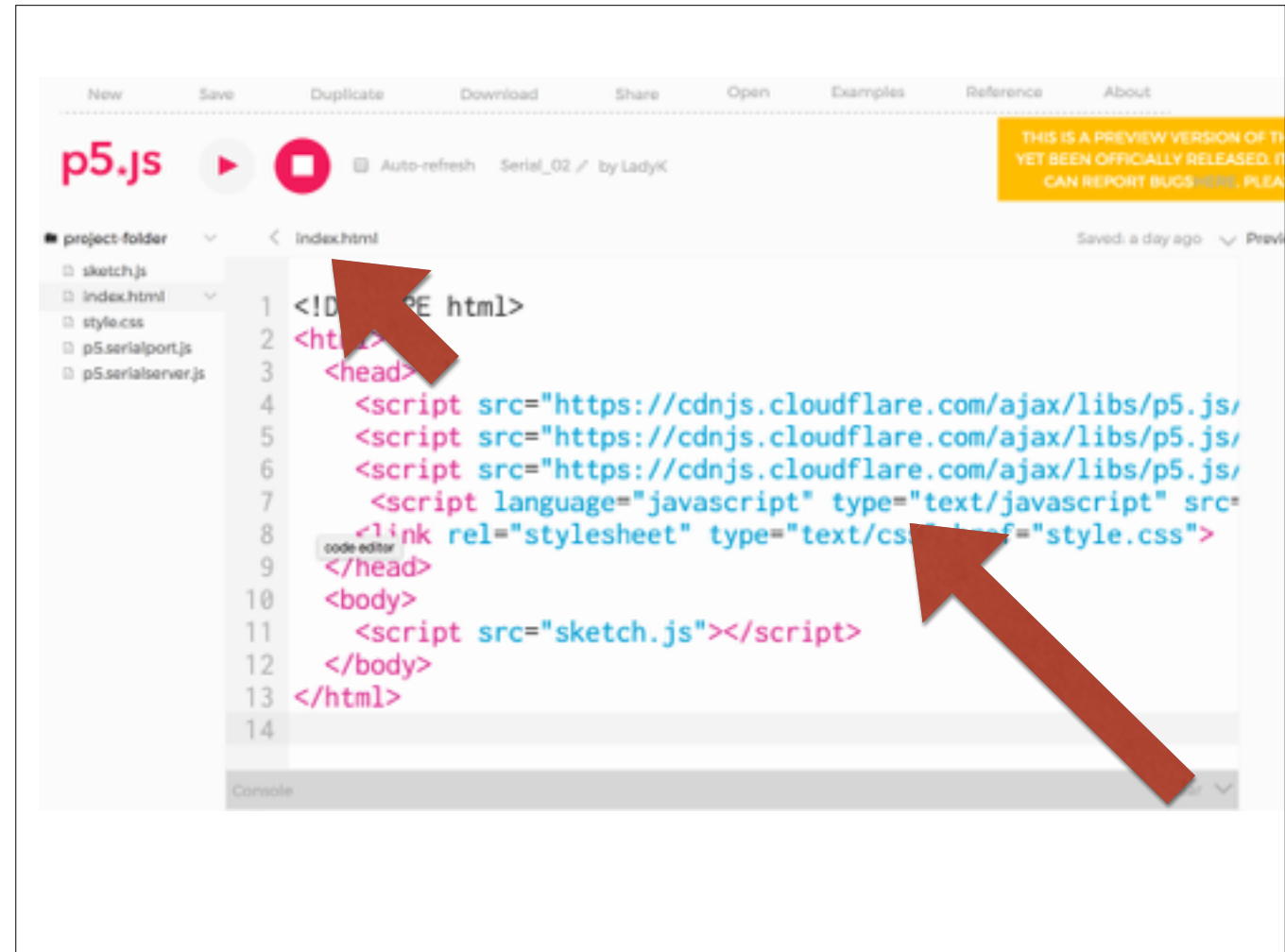


## Serial example

move the p5.serialport.js file and add it to your sketch, through the sidebar

you don't need the p5.serialserver.js file. ignore that file





Make a reference in your index.html for the p5.serialPort.js library in your index file

The screenshot shows the p5.js web editor interface. At the top, there are navigation buttons: New, Save, Duplicate, Download, Share, Open, Examples, Reference, and About. Below these, the p5.js logo is visible, along with a play button and a red square button. A notification banner on the right states: "THIS IS A PREVIEW VERSION OF YET BEEN OFFICIALLY RELEASED CAN REPORT BUGS HERE, PLI". The main editor area displays a file named "sketch.js" with the following code:

```
1 var serial; // variable to hold an instance of the serialport
2
3 function setup() {
4   serial = new p5.SerialPort(); // make a new instance of the
5   serial.on('list', printList); // set a callback function for
6
7   serial.list(); // list the serial ports
8
9
10 // get the list of ports:
11 function printList(portList) {
12   // portList is an array of serial port names
13   for (var i = 0; i < portList.length; i++) {
14     // Display the list the console:
15     console.log(i + " " + portList[i]);
```

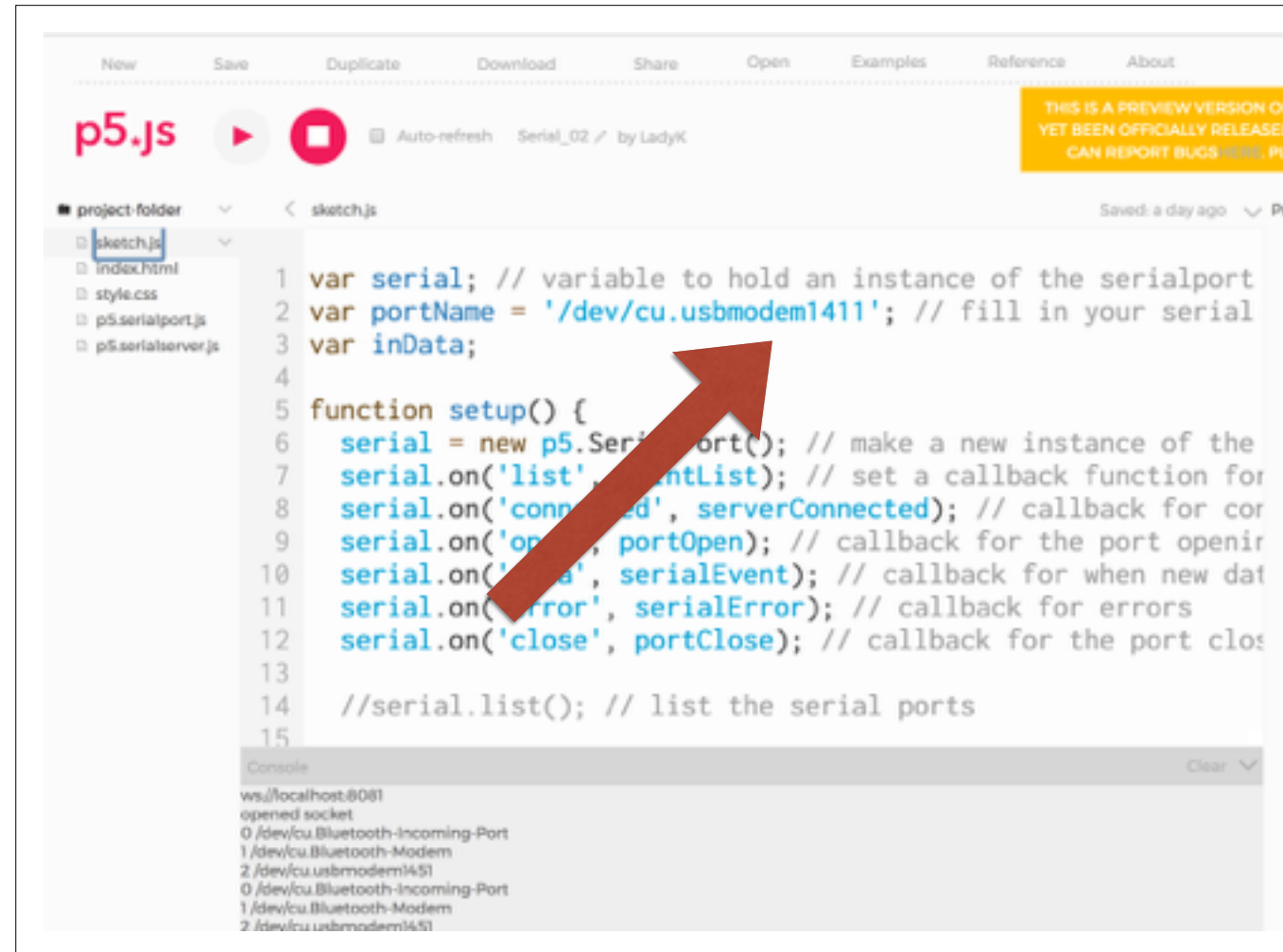
The console output at the bottom shows the following messages:

```
ws://localhost:
opened socket
0 /dev/cu.Bluetooth-Incoming-Port
1 /dev/cu.Bluetooth-Modem
2 /dev/cu.usbmodem1451
0 /dev/cu.Bluetooth-Incoming-Port
1 /dev/cu.Bluetooth-Modem
2 /dev/cu.usbmodem1451
```

A red arrow points from the console output to the code, specifically to the `serial.list()` function call in the `setup()` function.

Make sure you are referencing your port number that you are communicating on.

Make note of it in the console log



The screenshot shows the p5.js web editor interface. At the top, there's a navigation bar with links: New, Save, Duplicate, Download, Share, Open, Examples, Reference, and About. Below this, the p5.js logo is on the left, and a yellow banner on the right states: "THIS IS A PREVIEW VERSION OF p5.js. IT HAS NOT YET BEEN OFFICIALLY RELEASED. YOU CAN REPORT BUGS HERE. PLEASE".

The main workspace shows a file explorer on the left with a project folder containing sketch.js, index.html, style.css, p5.serialport.js, and p5.serialserver.js. The sketch.js file is open, showing the following code:

```
1 var serial; // variable to hold an instance of the serialport
2 var portName = '/dev/cu.usbmodem1411'; // fill in your serial
3 var inData;
4
5 function setup() {
6   serial = new p5.SerialPort(); // make a new instance of the
7   serial.on('list', serialList); // set a callback function for
8   serial.on('connected', serverConnected); // callback for con
9   serial.on('opened', portOpen); // callback for the port openi
10  serial.on('data', serialEvent); // callback for when new dat
11  serial.on('error', serialError); // callback for errors
12  serial.on('close', portClose); // callback for the port clos
13
14  //serial.list(); // list the serial ports
15
```

A red arrow points to the value of the 'portName' variable in line 2. Below the code editor is a console window showing the output of the serial list command:

```
ws://localhost:8081
opened socket
0 /dev/cu.Bluetooth-Incoming-Port
1 /dev/cu.Bluetooth-Modem
2 /dev/cu.usbmodem1451
0 /dev/cu.Bluetooth-Incoming-Port
1 /dev/cu.Bluetooth-Modem
2 /dev/cu.usbmodem1451
```

change your port name

# Input

- Digital switch

Or you could do analog in. Or output.

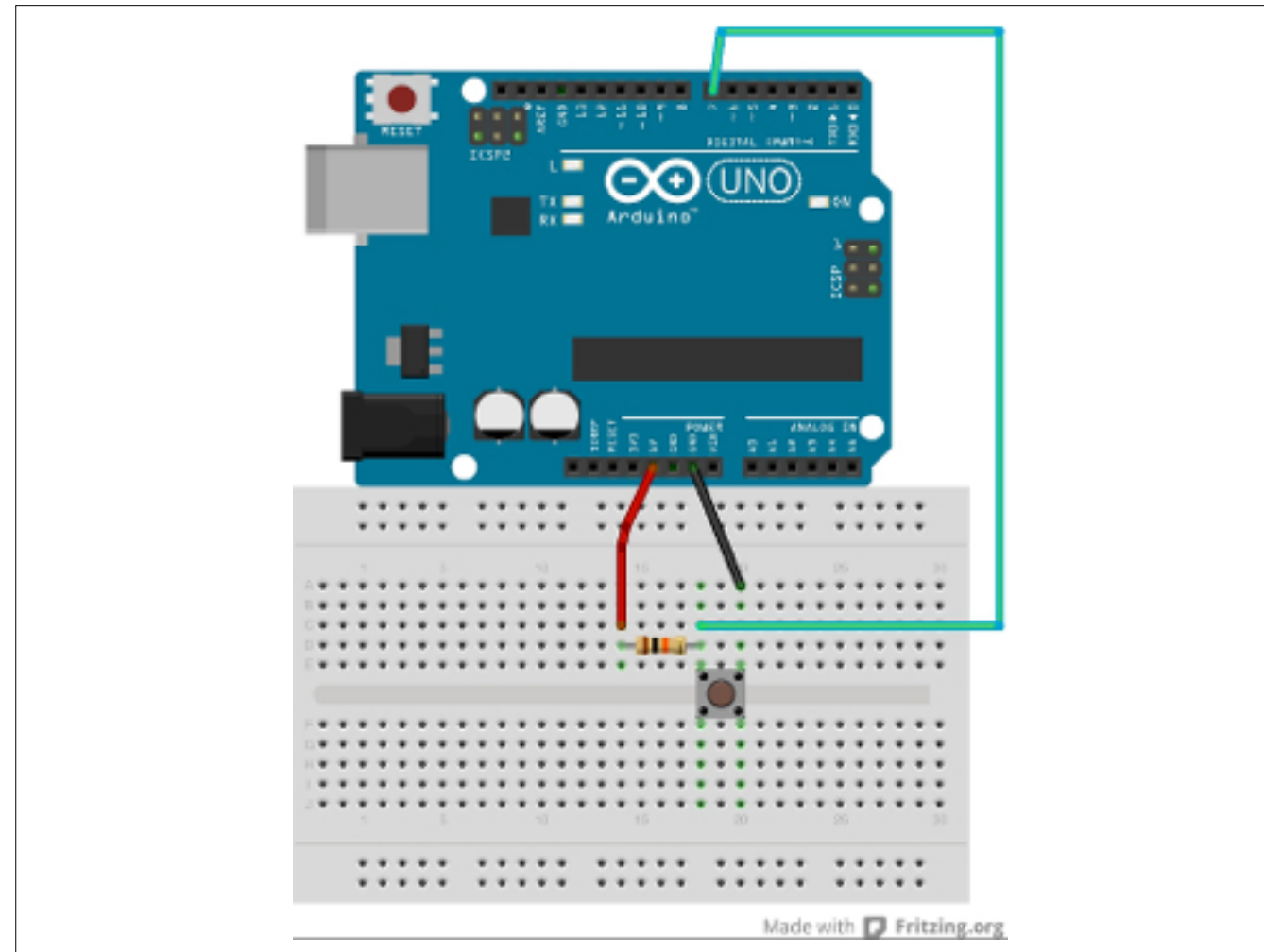
But serially write the data in/out to the arduino

Burn the Digital Output code  
on the Arduino

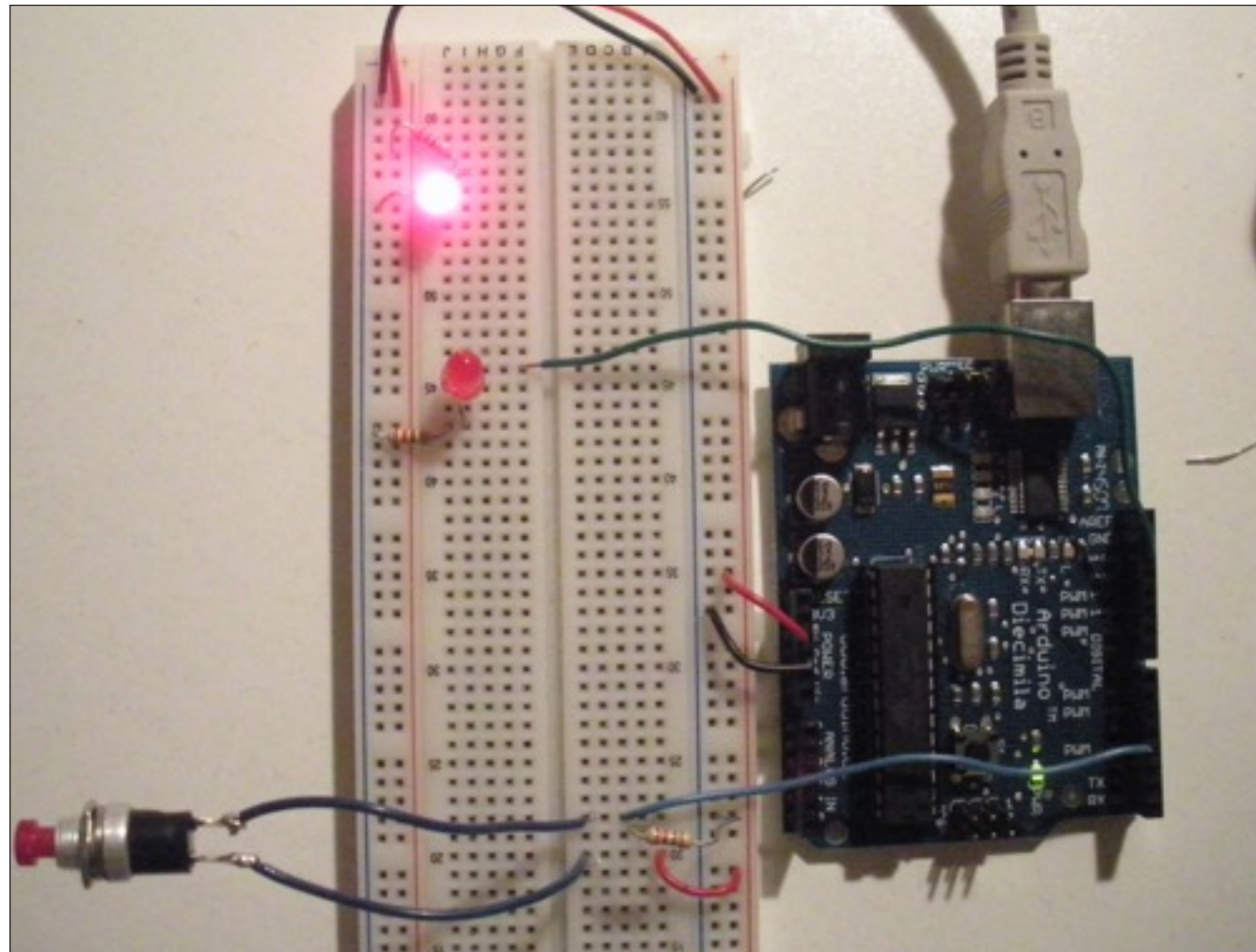
Burn the code on the arduino and close the arduino application

# Start the p5 Serial Control App



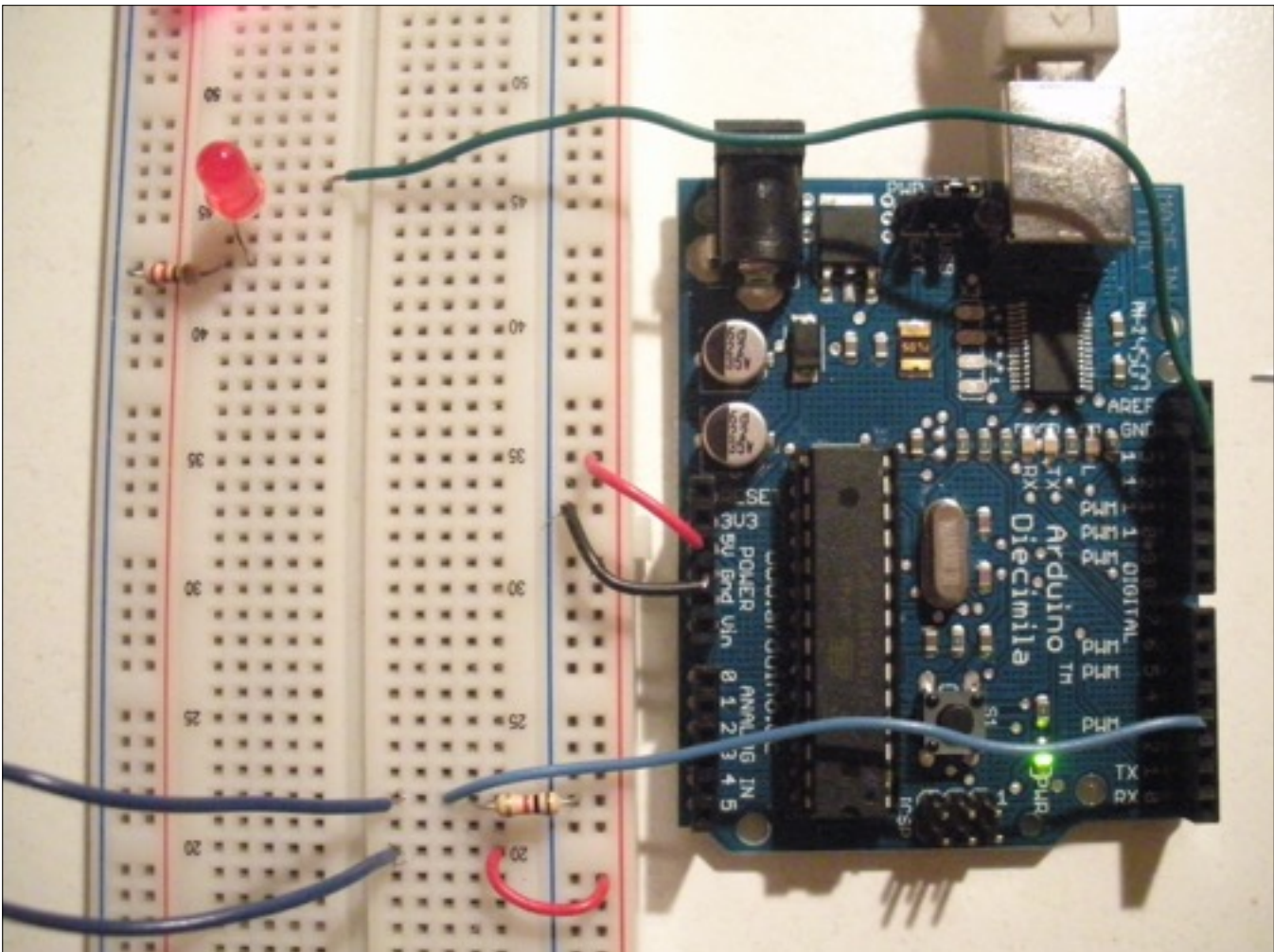


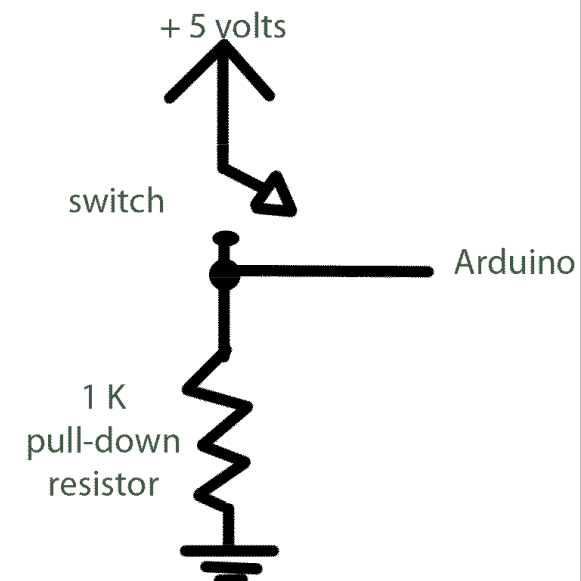
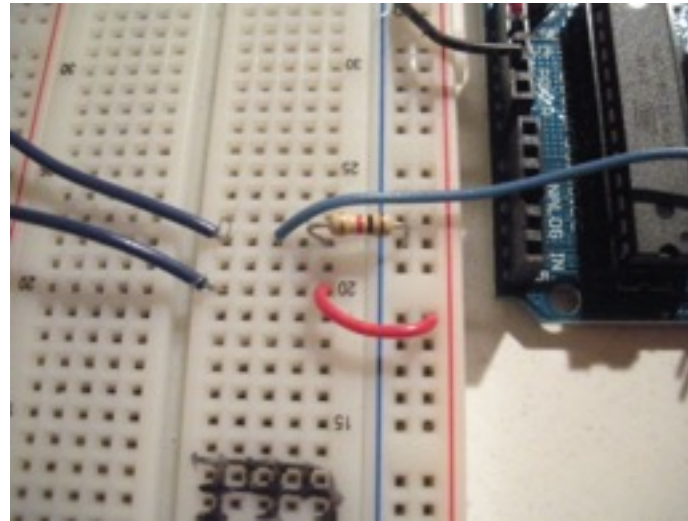
wire up your circuits accordingly



Basic set-up for the switch: switch, 1k resistor, wires.







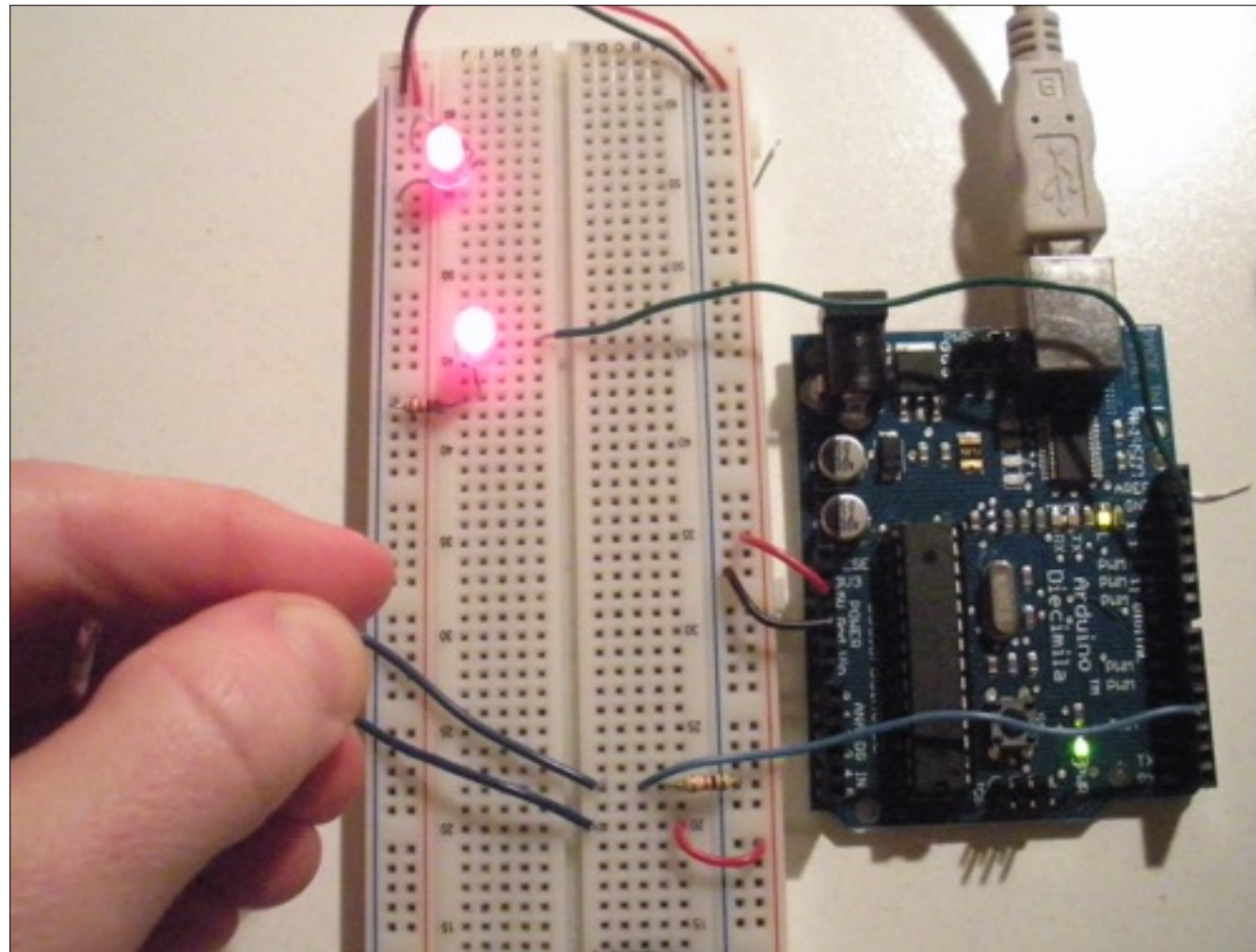
### A Switch

Switches ALWAYS need a pull-down resistor. One side of this resistor goes between ground. The other side goes to the “node” where one end of the switch also connects to the wire going into our Arduino Board.

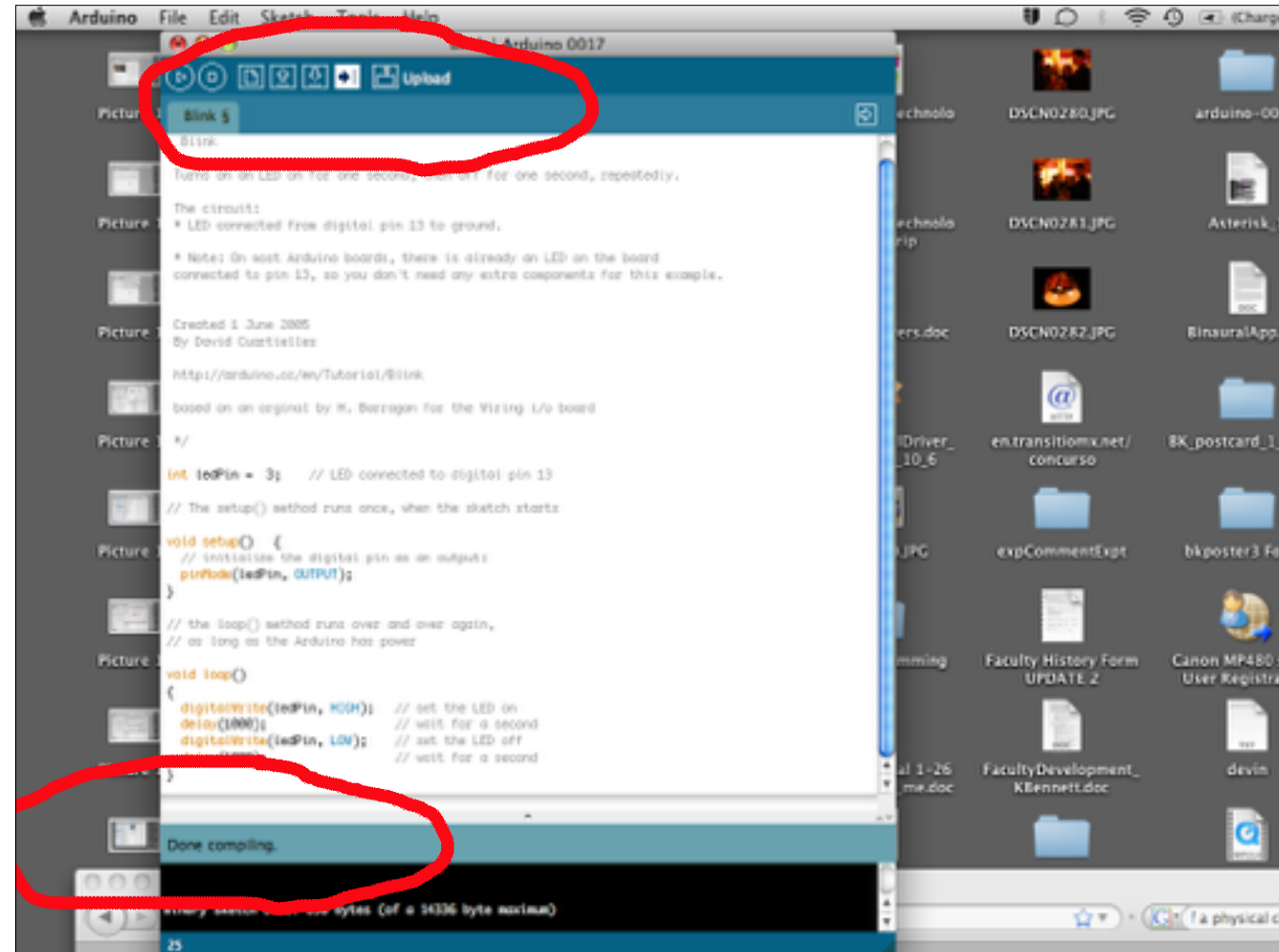
The other end of the switch is connected to the positive 5 volts.

The resistor here prevents the Arduino from reading low voltages in the atmosphere.

Electrical current always flows from the path of least resistance to ground. If you just had a wire connecting the switch and the pin to ground, then, when you close the switch, the path of least resistance would be through the wire, and you’d have a short circuit. The easiest path for the current is through the closed switch. When the switch is open, the resistor offers the only path and the current goes through it.

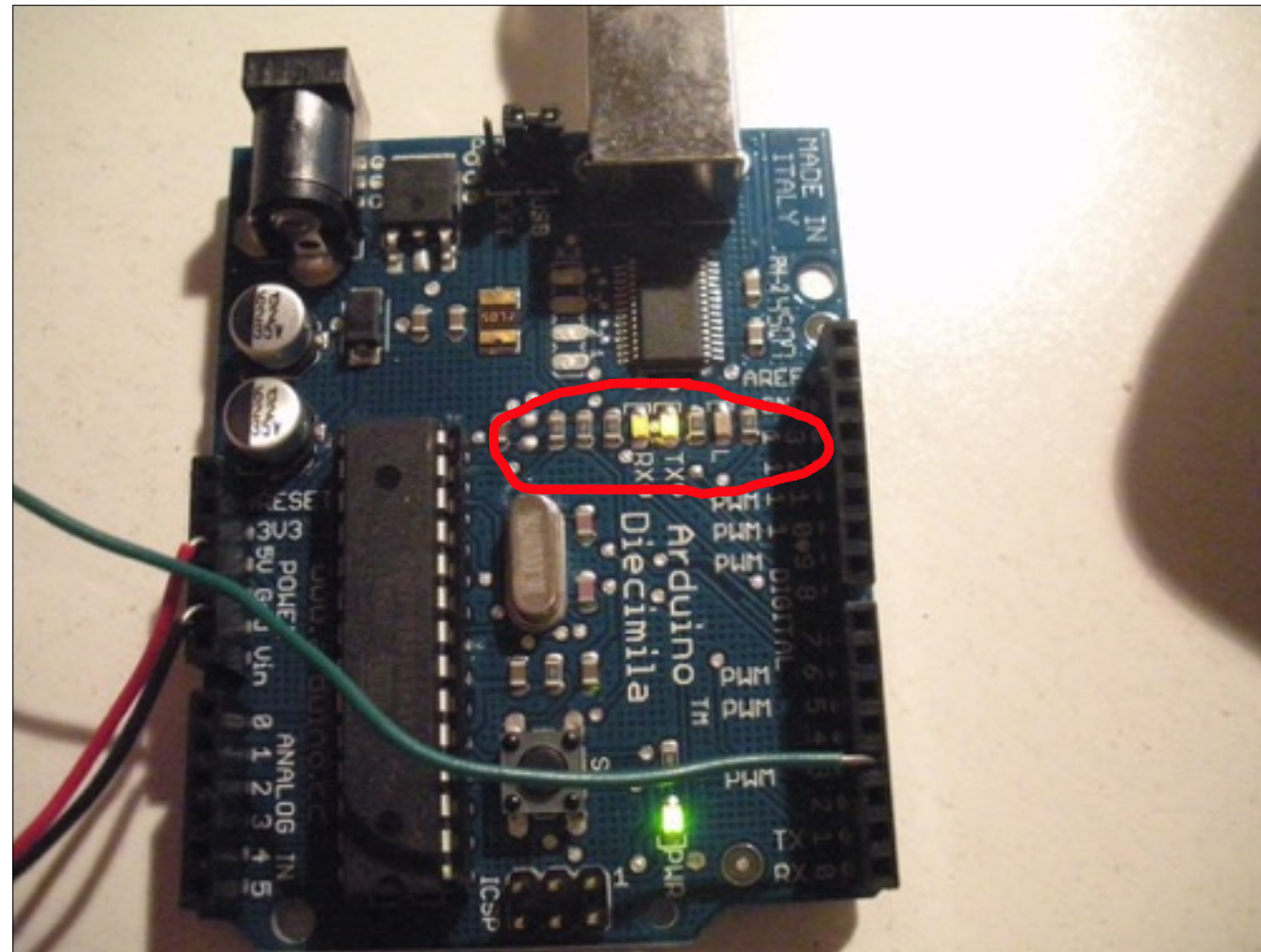


Make sure you have the correct ports and chip selected in the Arduino Program.  
Upload your button code. Make sure the code reflects the correct pins you've wired your button (input- listening) and led(output - talking) to.



Let's upload the code to the chip by pressing the upload button at the top of your Arduino sketch window, on your computer. This is now burning the code onto the chip. Notice the flashing TX and RX lights on the Arduino board while this is happening.





Notice the flashing TX and RX lights on the Arduino board while this is happening.

