

SOUND INTERACTION WORKSHOP

Sensing the world

Imanol Gómez
imanolgomez.net
5th May 2017

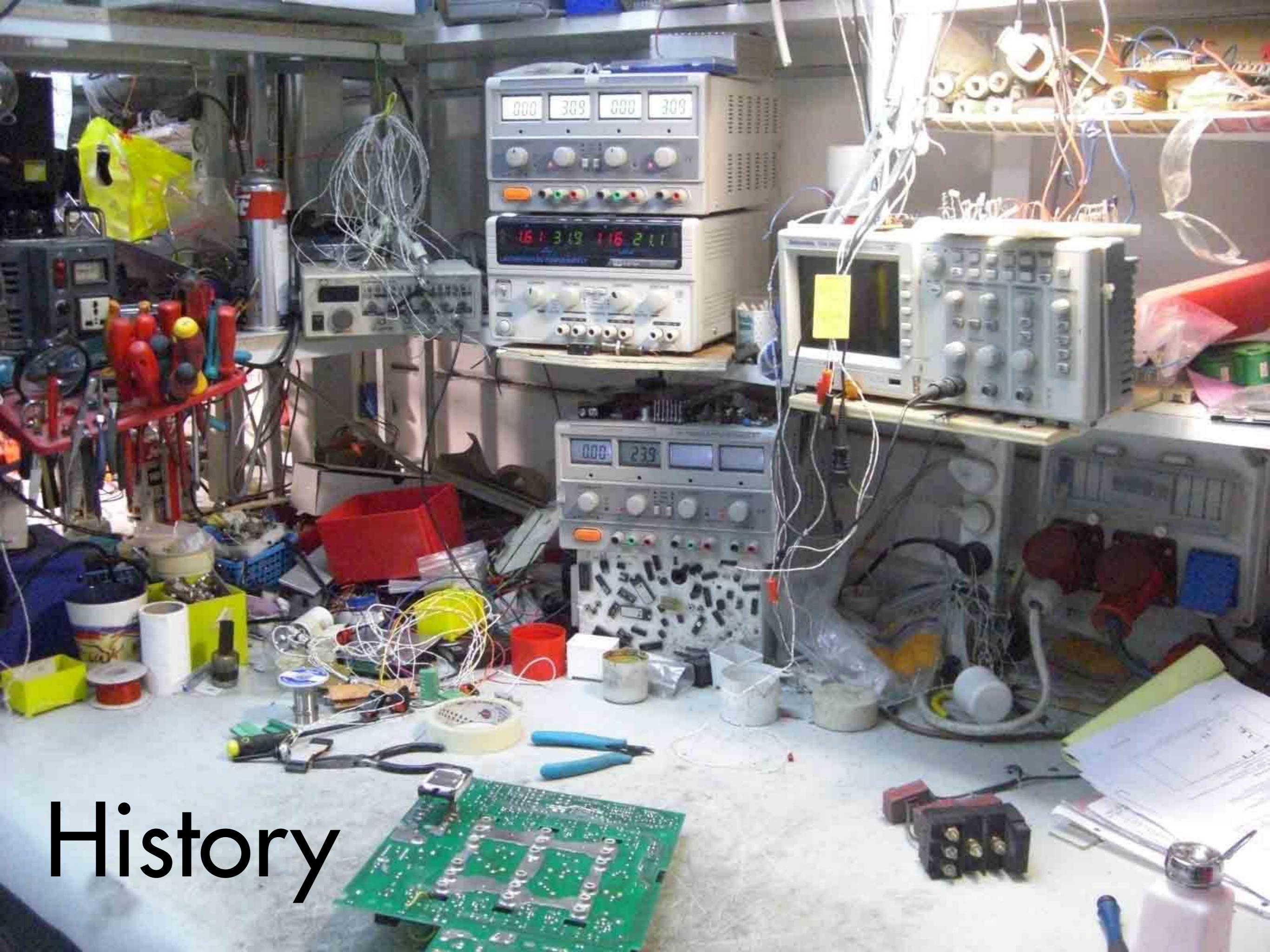
INTRODUCTION

Repository

[https://github.com/ImanolGo/
SoundInteractionWorkshop](https://github.com/ImanolGo/SoundInteractionWorkshop)



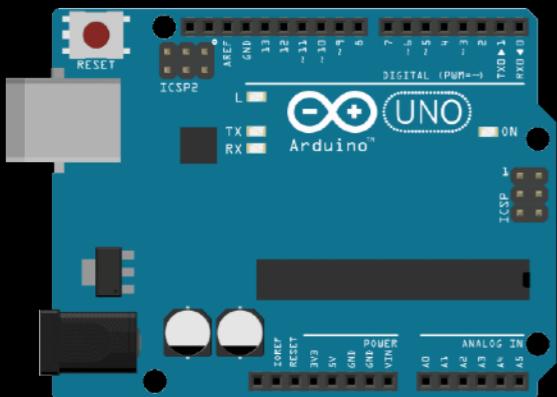
History



Hardware

Software

Community



A screenshot of the Arduino IDE interface. The title bar says "sketch_may03a | Arduino 1.8.1". The main window shows a code editor with the following code:

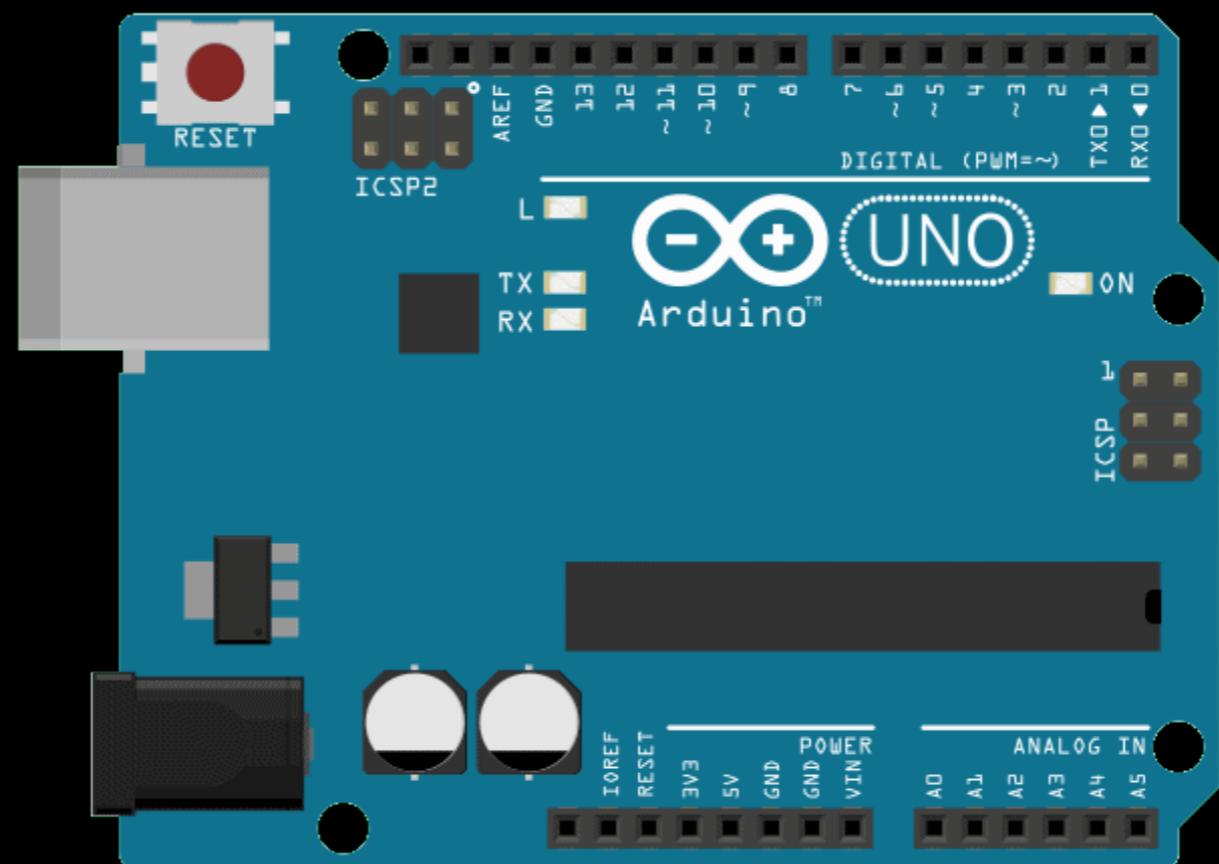
```
sketch_may03a
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8 }
```

A screenshot of the Arduino Forum website. The header includes the Arduino logo and navigation links for HOME, BUY, SOFTWARE, PRODUCTS, LEARNING, FORUM, SUPPORT, and BLOG. The main content area is titled "Using Arduino" and lists several categories with their respective posts and topics:

- Installation & Troubleshooting**: 79,722 Posts, 18,708 Topics
- Project Guidance**: 393,657 Posts, 54,214 Topics
- Programming Questions**: 540,850 Posts, 66,431 Topics
- General Electronics**: 186,704 Posts, 18,501 Topics
- Microcontrollers**: 75,895 Posts, 8,861 Topics
- LEDs and Multiplexing**: 53,742 Posts, 6,714 Topics
- Displays**: 59,772 Posts, 8,078 Topics
- Audio**: 79,066 Posts, 3,024 Topics

Adafruit Feather 32u4 on /dev/cu.usbmodem96

Arduino UNO



Boards



Arduino Uno



Arduino Leonardo



Arduino Due



Arduino Yún



Arduino Tre



Arduino Micro



Arduino Robot



Arduino Esplora



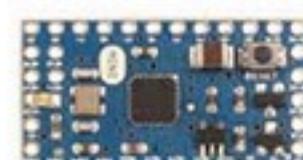
Arduino Mega ADK



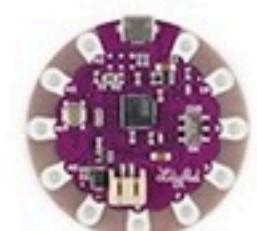
Arduino Ethernet



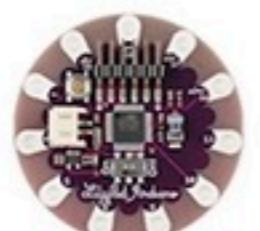
Arduino Mega 2560



Arduino Mini



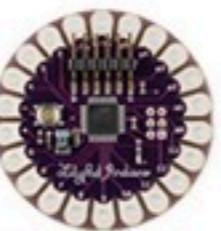
LilyPad Arduino USB



LilyPad Arduino
Simple



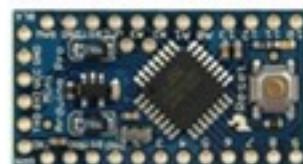
LilyPad Arduino
SimpleSnap



LilyPad Arduino

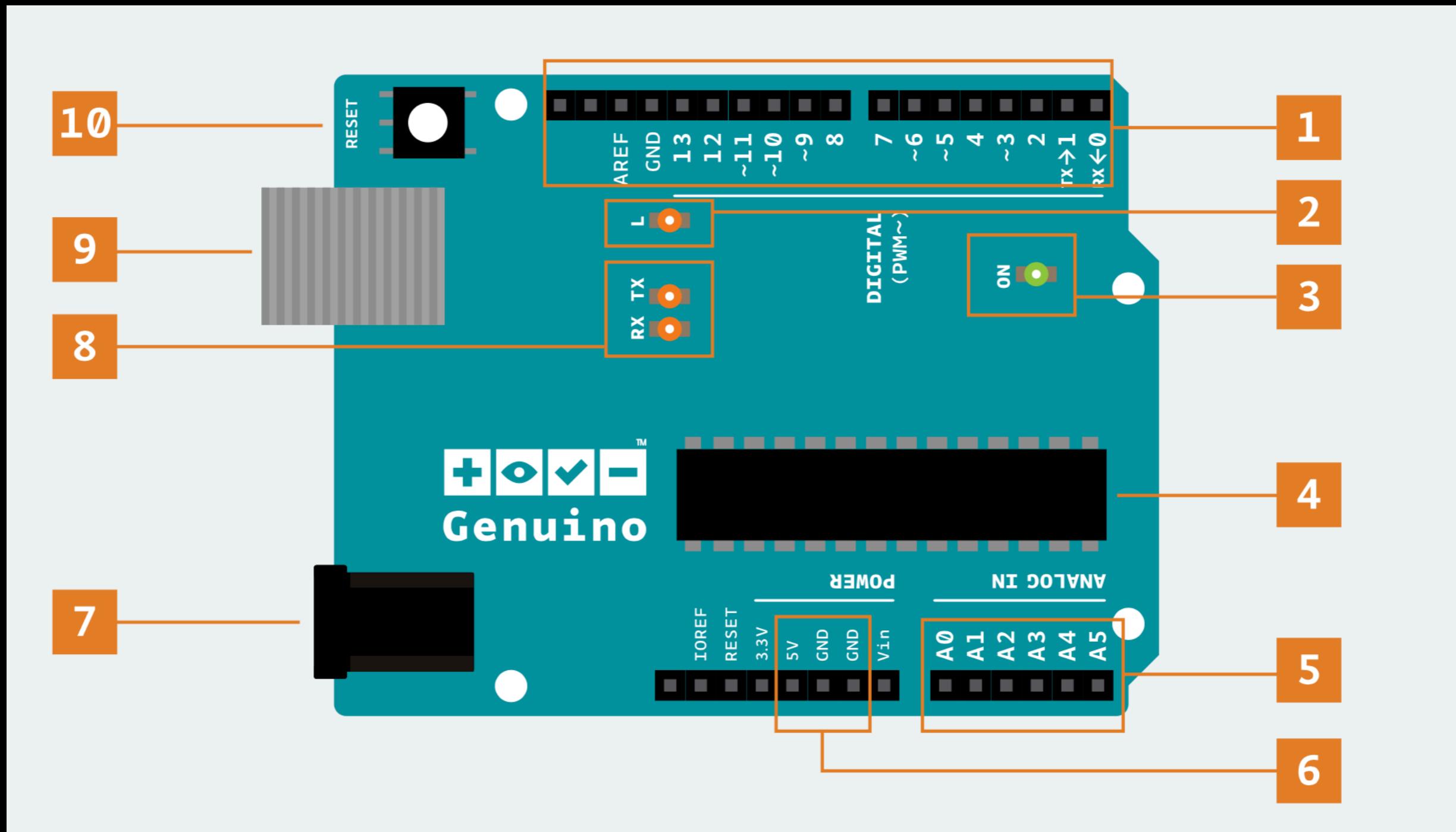


Arduino Nano

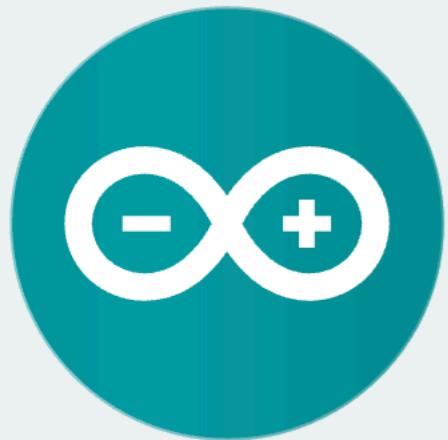


Arduino Pro Mini

What's on the board?



Download Software



ARDUINO 1.8.2

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer

Windows ZIP file for non admin install

Windows app [Get](#)

Mac OS X 10.7 Lion or newer

Linux 32 bits

Linux 64 bits

Linux ARM

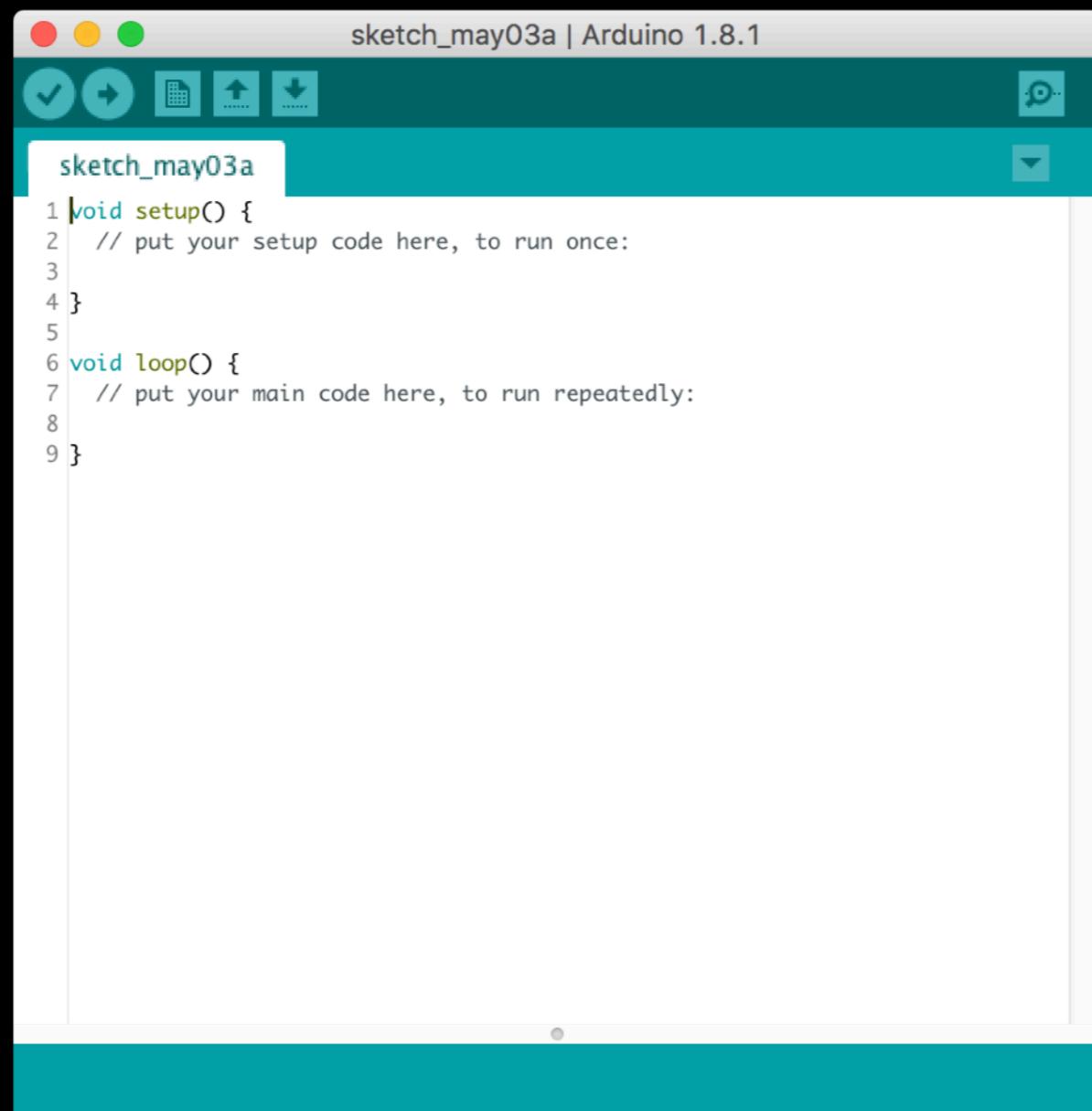
[Release Notes](#)

[Source Code](#)

[Checksums \(sha512\)](#)

<https://www.arduino.cc/en/Main/Software>

Arduino Software



```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }  
4  
5 void loop() {  
6     // put your main code here, to run repeatedly:  
7 }  
8  
9 }
```

Arduino Software

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See [uploading](#) below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the **File | Sketchbook** menu instead.

Save

Saves your sketch.

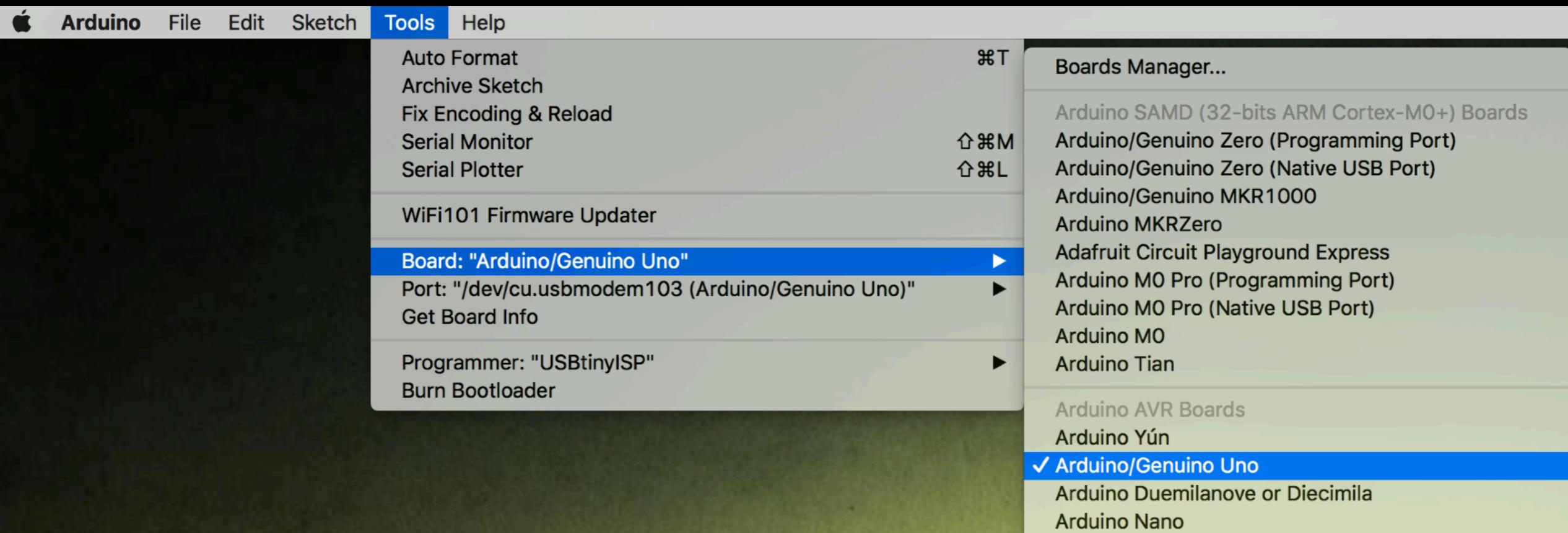
Serial Monitor

Opens the [serial monitor](#).

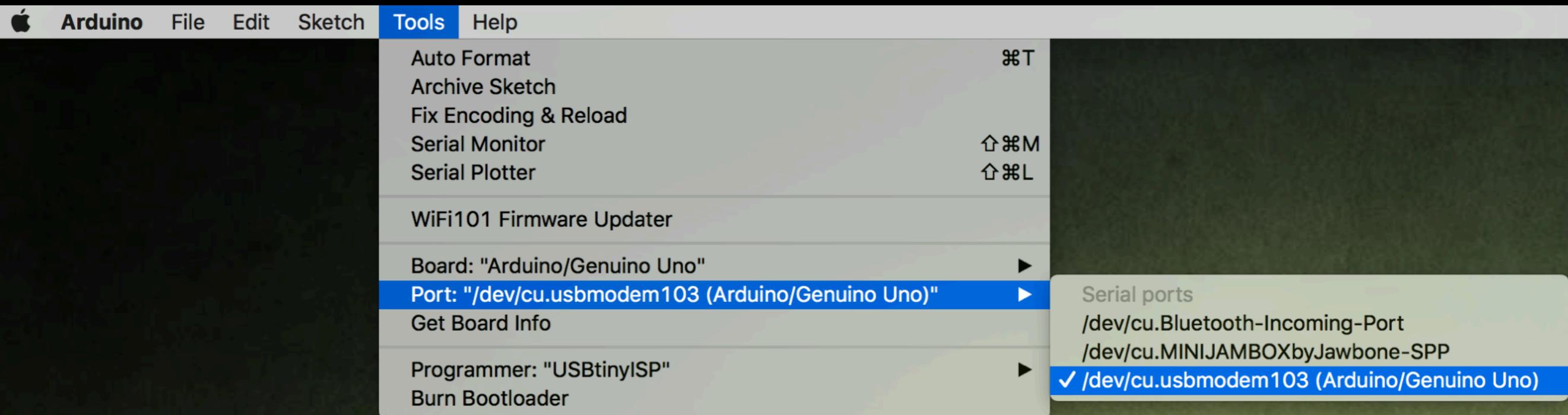
Connect the board



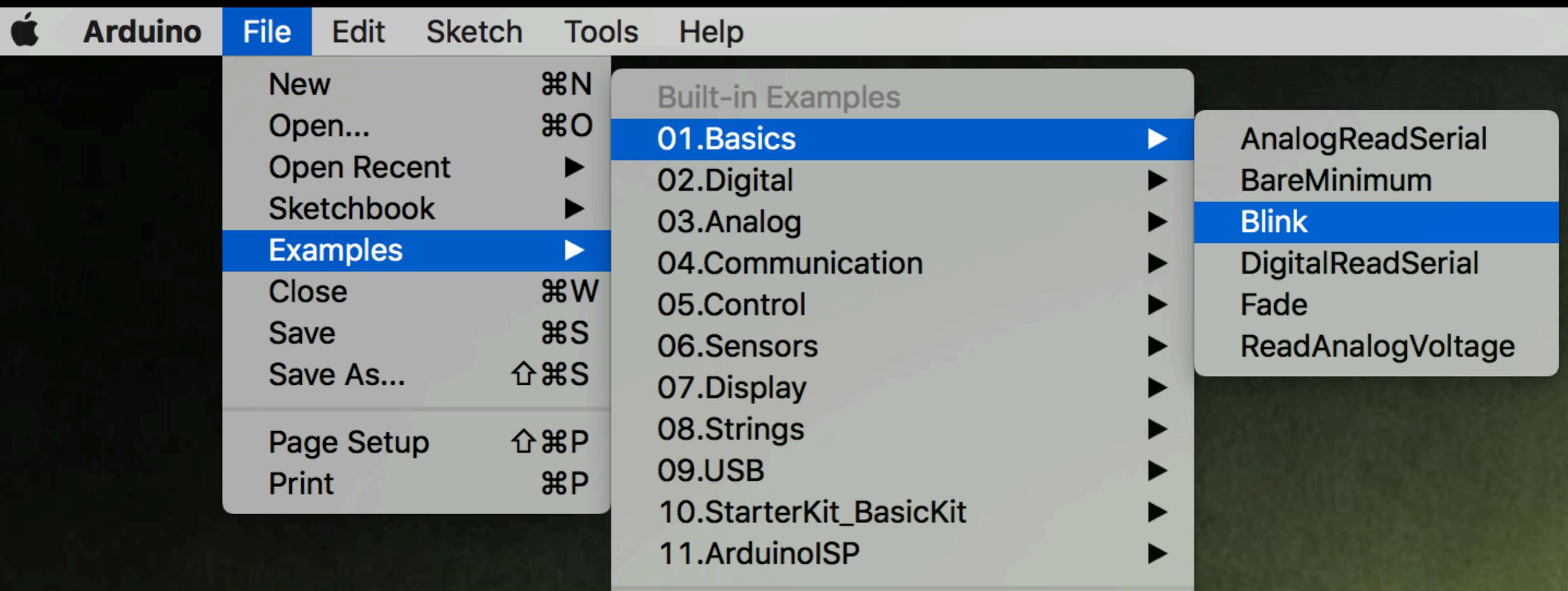
Select Board Type



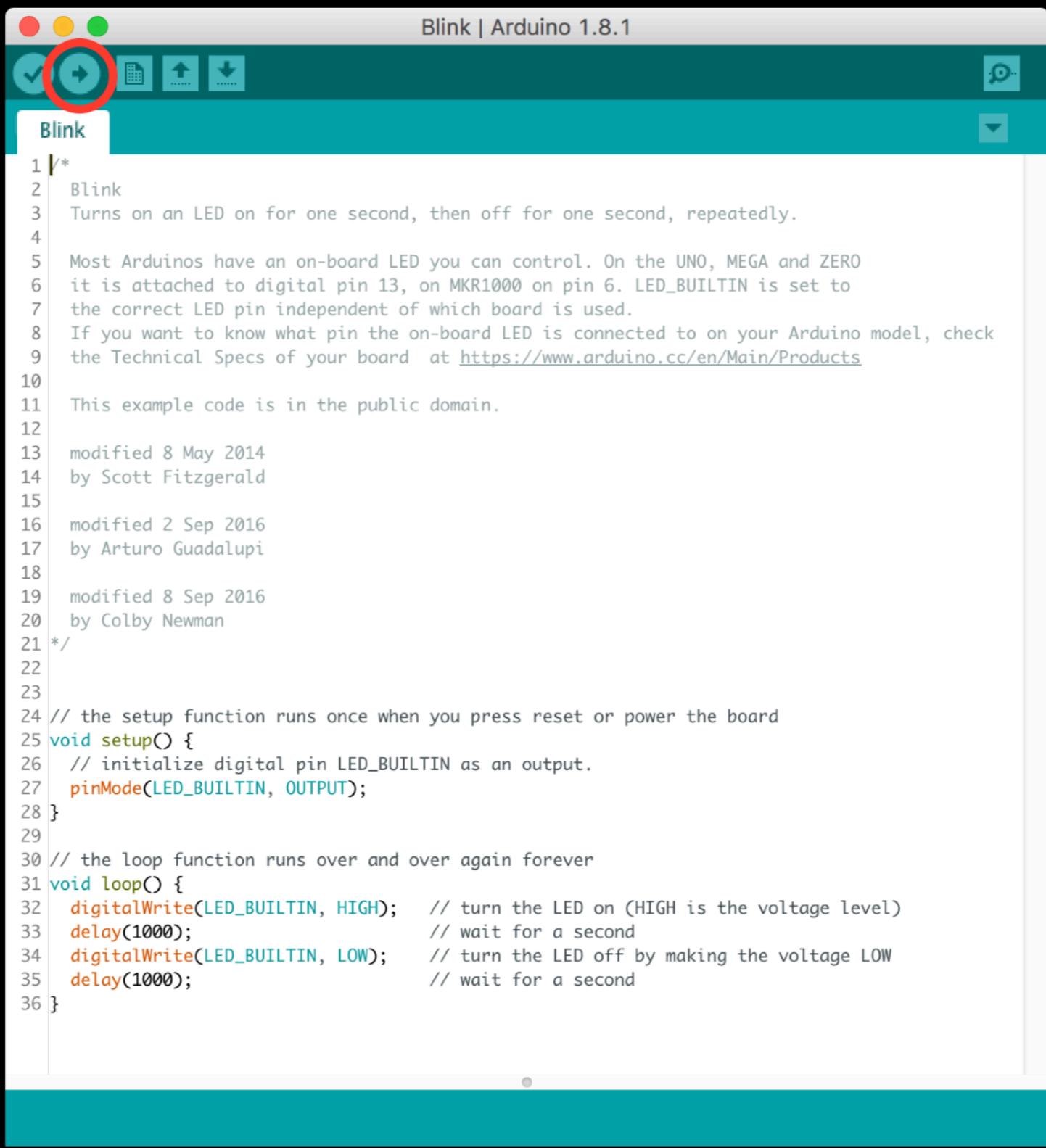
Select Correct Serial Port



Open Blink Sketch



Upload

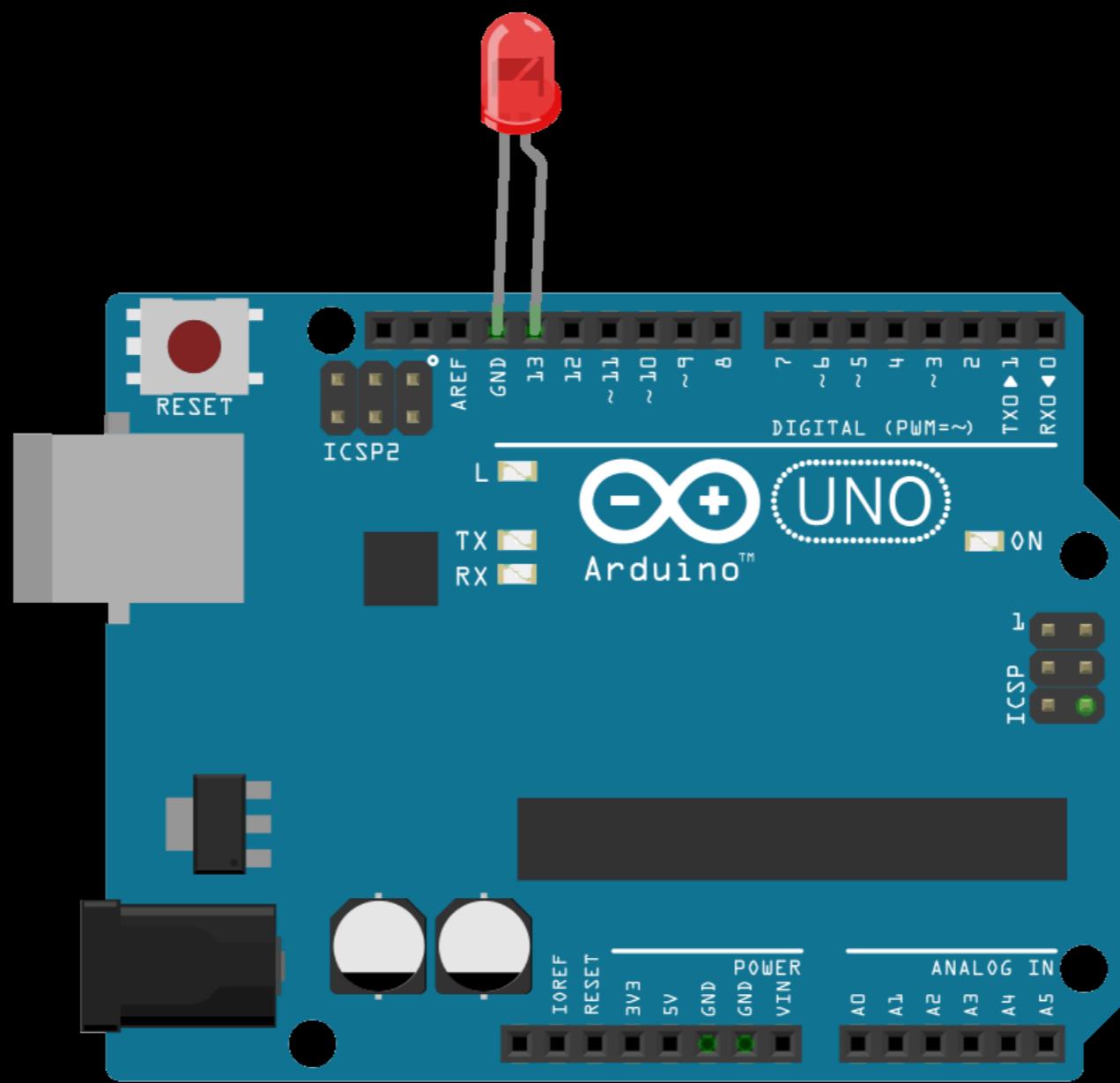


Watch!

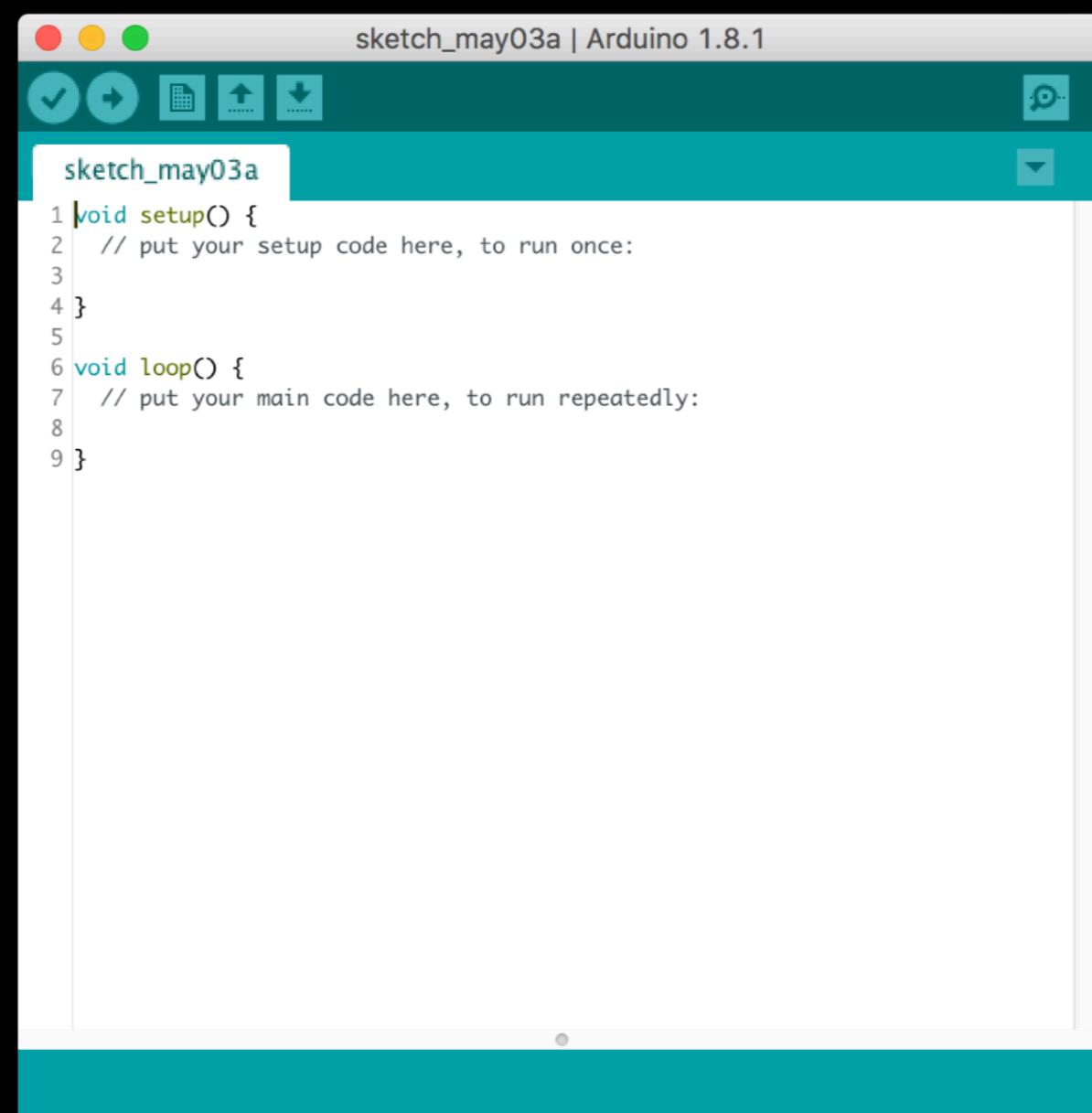


How long does Blink run for?

Experiment



Program Structure



The screenshot shows the Arduino IDE interface with a sketch titled "sketch_may03a". The code editor displays the following structure:

```
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8 }
```

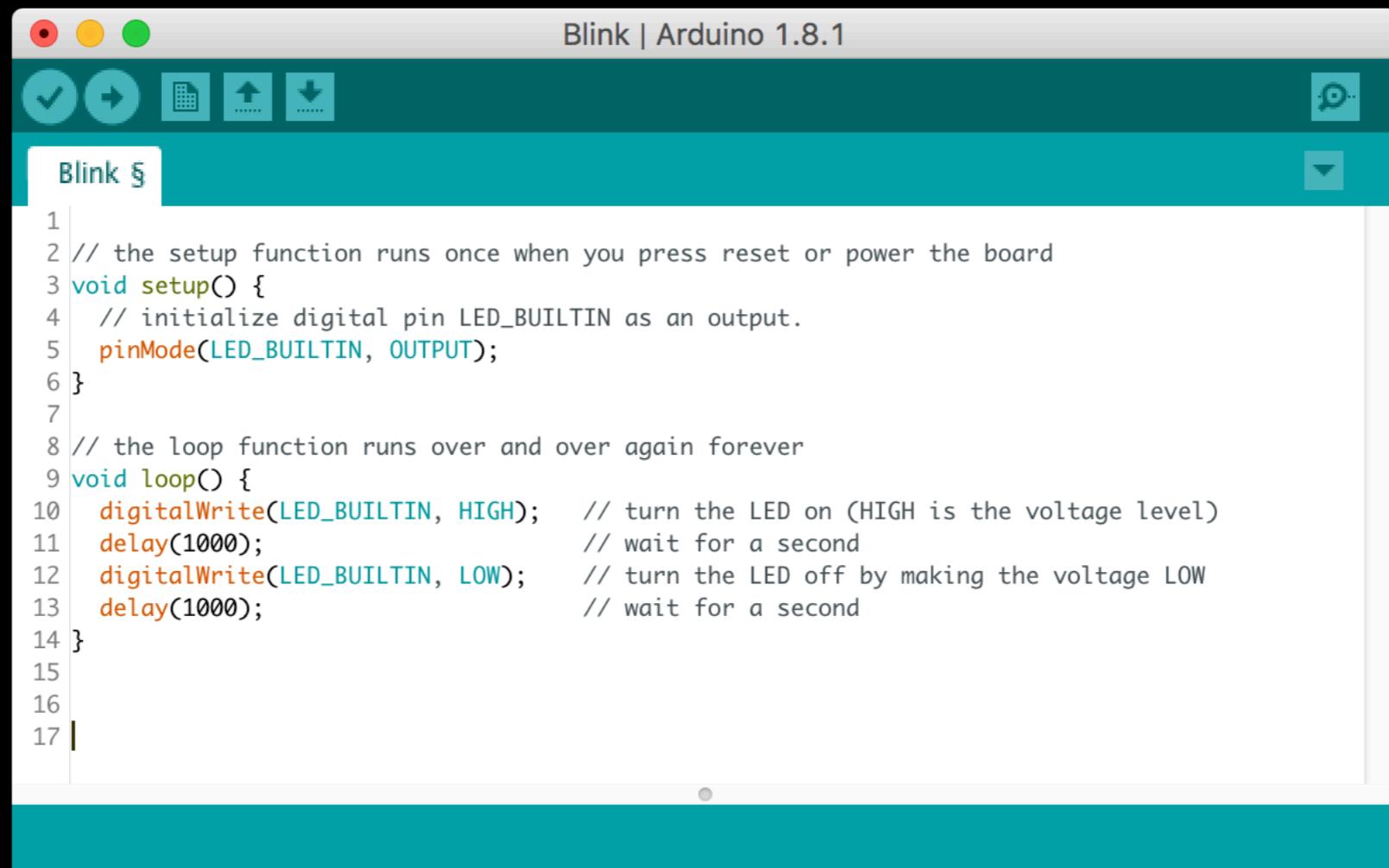
Statement

```
// initialize digital pin 13 as an output.  
pinMode(13, OUTPUT);
```

Function

Function name	(Inputs & Details Separated by commas)	;
pinMode	(13, OUTPUT)	;

Inside the code



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.8.1". Below the title bar is a toolbar with icons for file operations (checkmark, arrow, file, up, down) and a serial monitor icon. The main area displays the "Blink" sketch:

```
1 // the setup function runs once when you press reset or power the board
2 void setup() {
3     // initialize digital pin LED_BUILTIN as an output.
4     pinMode(LED_BUILTIN, OUTPUT);
5 }
6
7
8 // the loop function runs over and over again forever
9 void loop() {
10    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
11    delay(1000);                      // wait for a second
12    digitalWrite(LED_BUILTIN, LOW);       // turn the LED off by making the voltage LOW
13    delay(1000);                      // wait for a second
14 }
15
16
17 |
```

Changing the delay

```
delay(500);
```

Exercises!

- Exercise 1: Modify the code so that the light is on for 100 milliseconds and off for 900 milliseconds. This makes for a nice once-per-second second timer.
- Exercise 2: Modify the code so that the light is on for 50 milliseconds and off for 50 milliseconds. What happens?
- Exercise 3: Modify the code so that the light is on for 10 ms (the shorthand for milliseconds) and off for 10 ms. What happens?

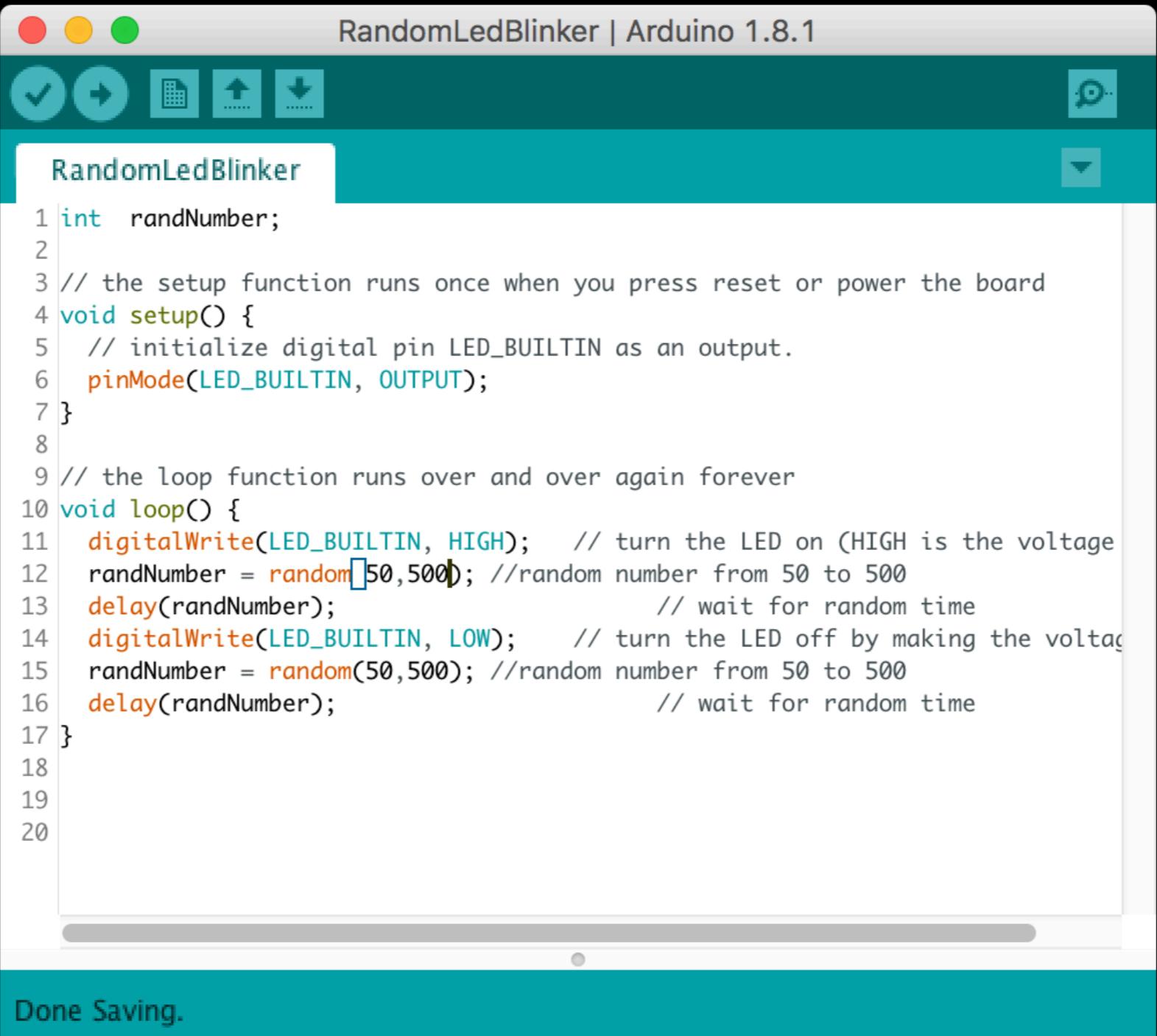
Project: Dubstep LED Blinker



Variable

```
int inputVariable1;  
int inputVariable2 = 0;      // both are correct
```

Random Blink



The screenshot shows the Arduino IDE interface with the title bar "RandomLedBlinker | Arduino 1.8.1". The main window displays the code for "RandomLedBlinker". The code uses the random number generator to control the built-in LED on an Arduino board, causing it to blink at random intervals between 50 and 500 milliseconds.

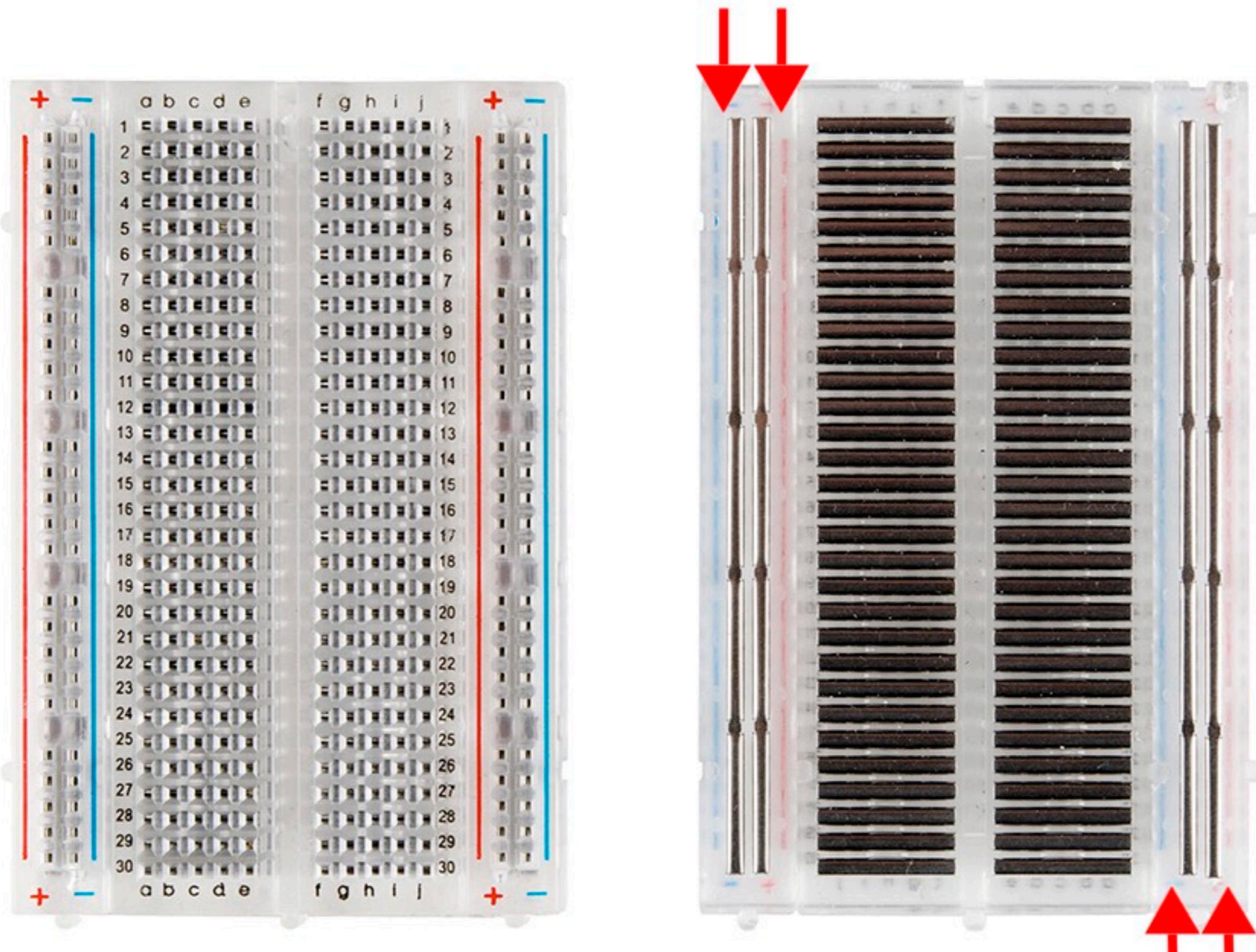
```
1 int randNumber;
2
3 // the setup function runs once when you press reset or power the board
4 void setup() {
5   // initialize digital pin LED_BUILTIN as an output.
6   pinMode(LED_BUILTIN, OUTPUT);
7 }
8
9 // the loop function runs over and over again forever
10 void loop() {
11   digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage
12   randNumber = random(50,500); //random number from 50 to 500
13   delay(randNumber);                // wait for random time
14   digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage
15   randNumber = random(50,500); //random number from 50 to 500
16   delay(randNumber);              // wait for random time
17 }
18
19
20
```

Done Saving.

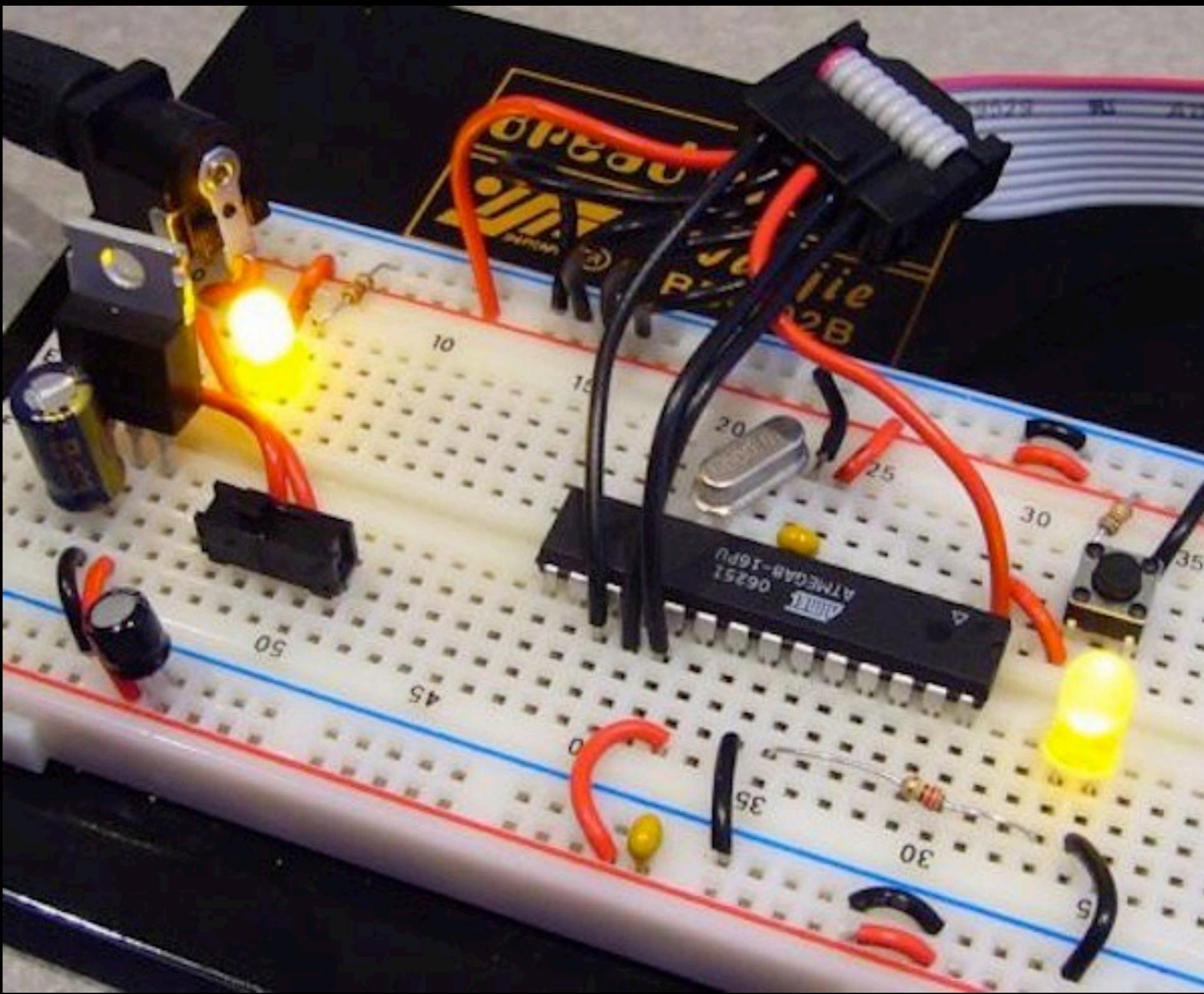
Breadboard



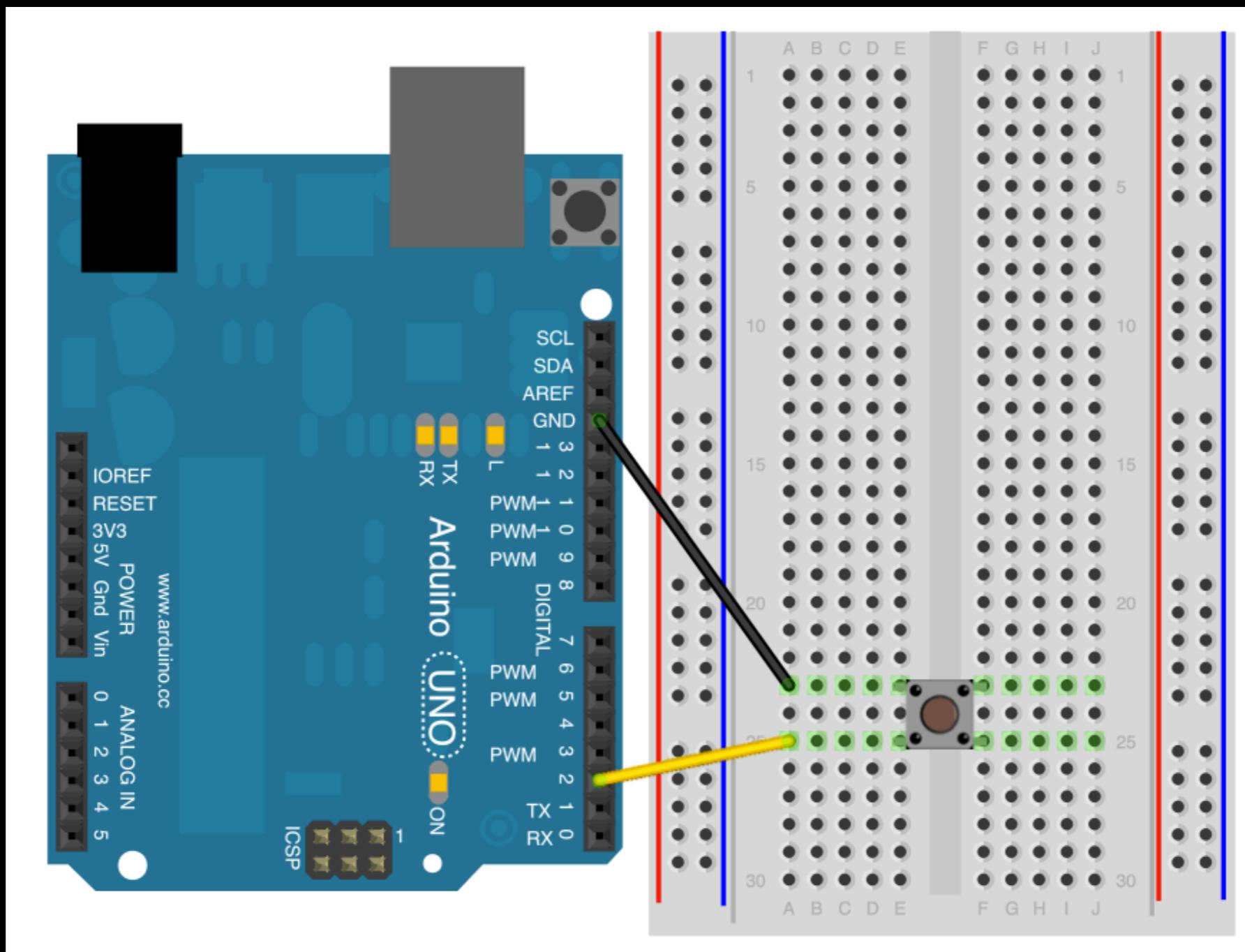
How to use a breadboard



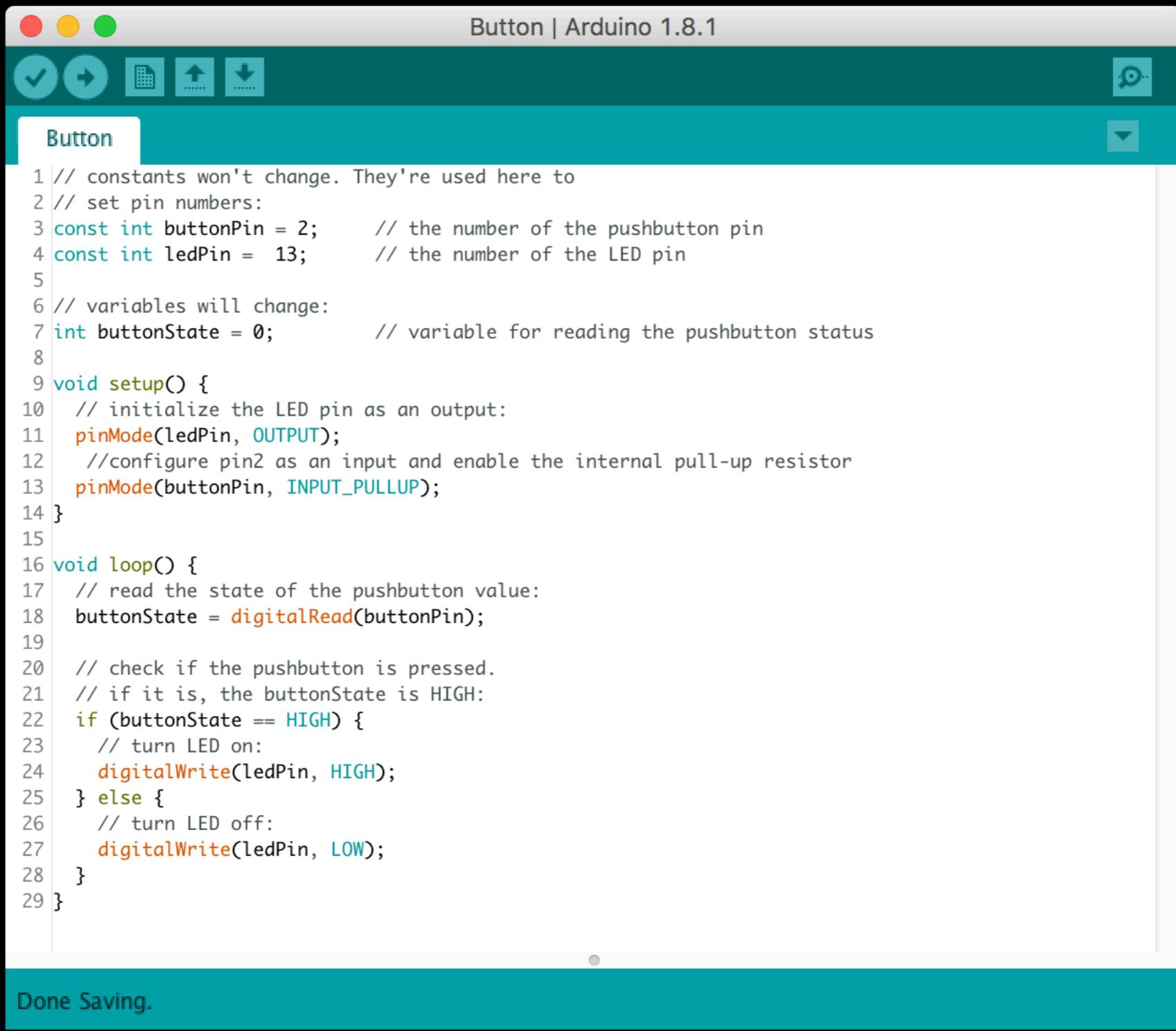
Circuit Example



Input Button



Input Button



The screenshot shows the Arduino IDE interface with the title bar "Button | Arduino 1.8.1". The main area displays the following C++ code for a button input example:

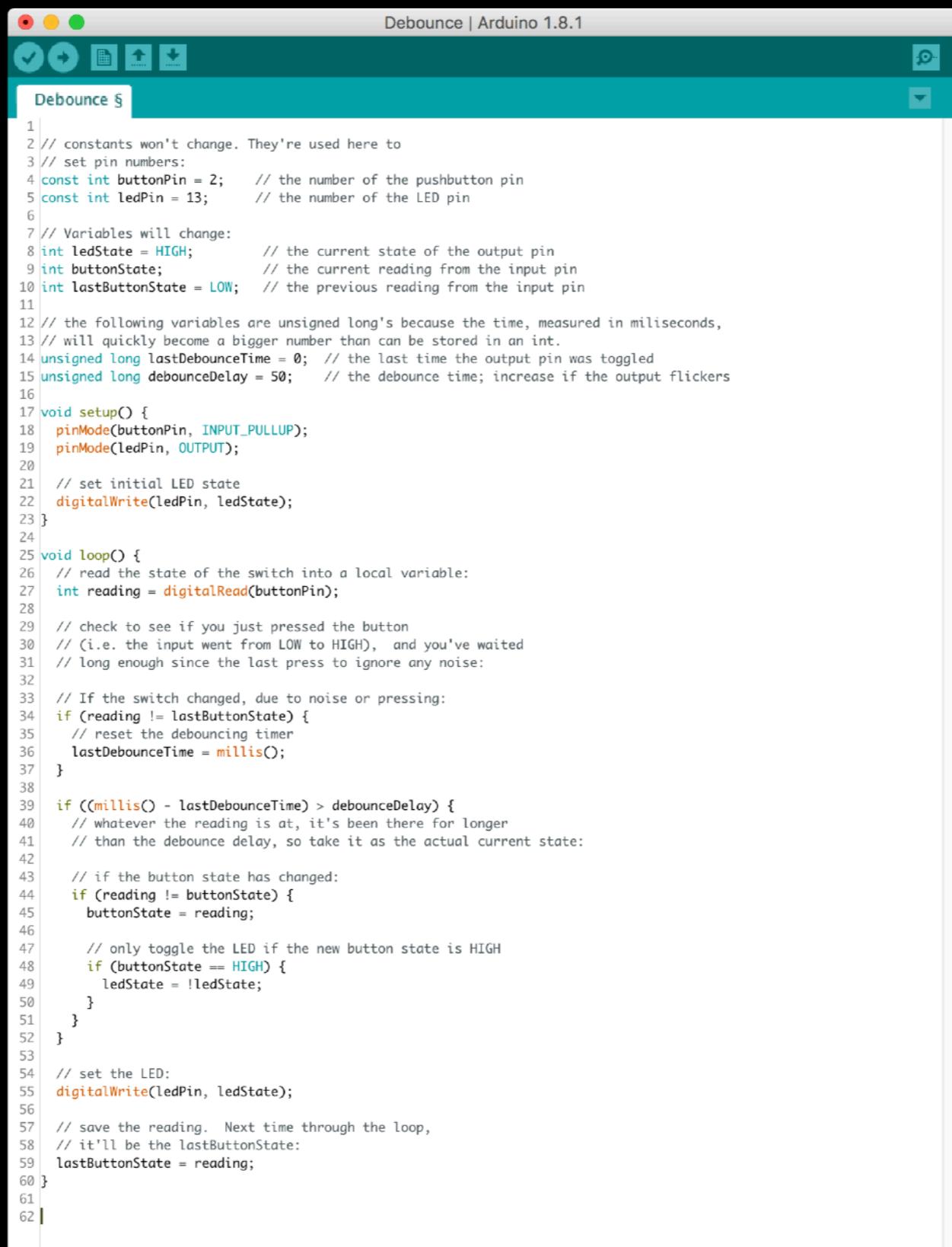
```
1 // constants won't change. They're used here to
2 // set pin numbers:
3 const int buttonPin = 2;      // the number of the pushbutton pin
4 const int ledPin = 13;        // the number of the LED pin
5
6 // variables will change:
7 int buttonState = 0;          // variable for reading the pushbutton status
8
9 void setup() {
10    // initialize the LED pin as an output:
11    pinMode(ledPin, OUTPUT);
12    //configure pin2 as an input and enable the internal pull-up resistor
13    pinMode(buttonPin, INPUT_PULLUP);
14 }
15
16 void loop() {
17    // read the state of the pushbutton value:
18    buttonState = digitalRead(buttonPin);
19
20    // check if the pushbutton is pressed.
21    // if it is, the buttonState is HIGH:
22    if (buttonState == HIGH) {
23        // turn LED on:
24        digitalWrite(ledPin, HIGH);
25    } else {
26        // turn LED off:
27        digitalWrite(ledPin, LOW);
28    }
29 }
```

At the bottom of the code editor, a blue bar displays the message "Done Saving."

if / else

```
if (pinFiveInput < 500)
{
    // action A
}
else
{
    // action B
}
```

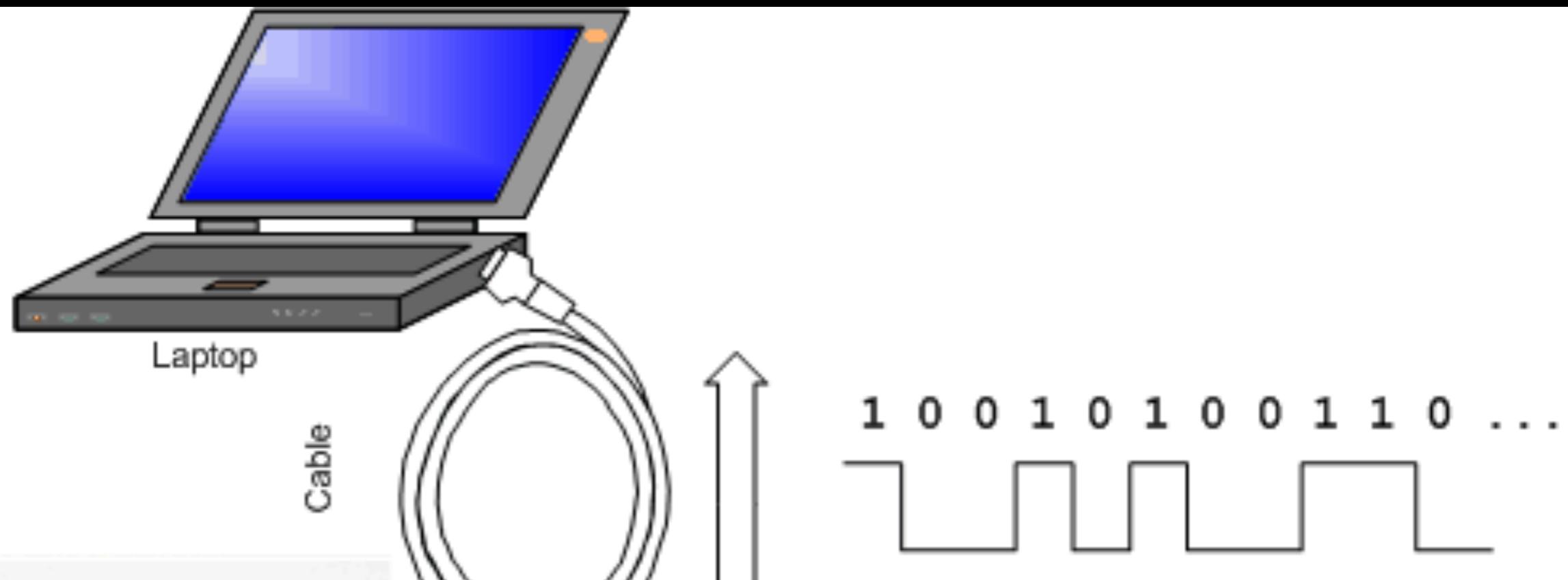
Switch Button



The image shows a screenshot of the Arduino IDE interface. The title bar reads "Debounce | Arduino 1.8.1". Below the title bar is a toolbar with various icons: a checkmark, a circular arrow, a file icon, an upload icon, and a download icon. The main area is a code editor with the following content:

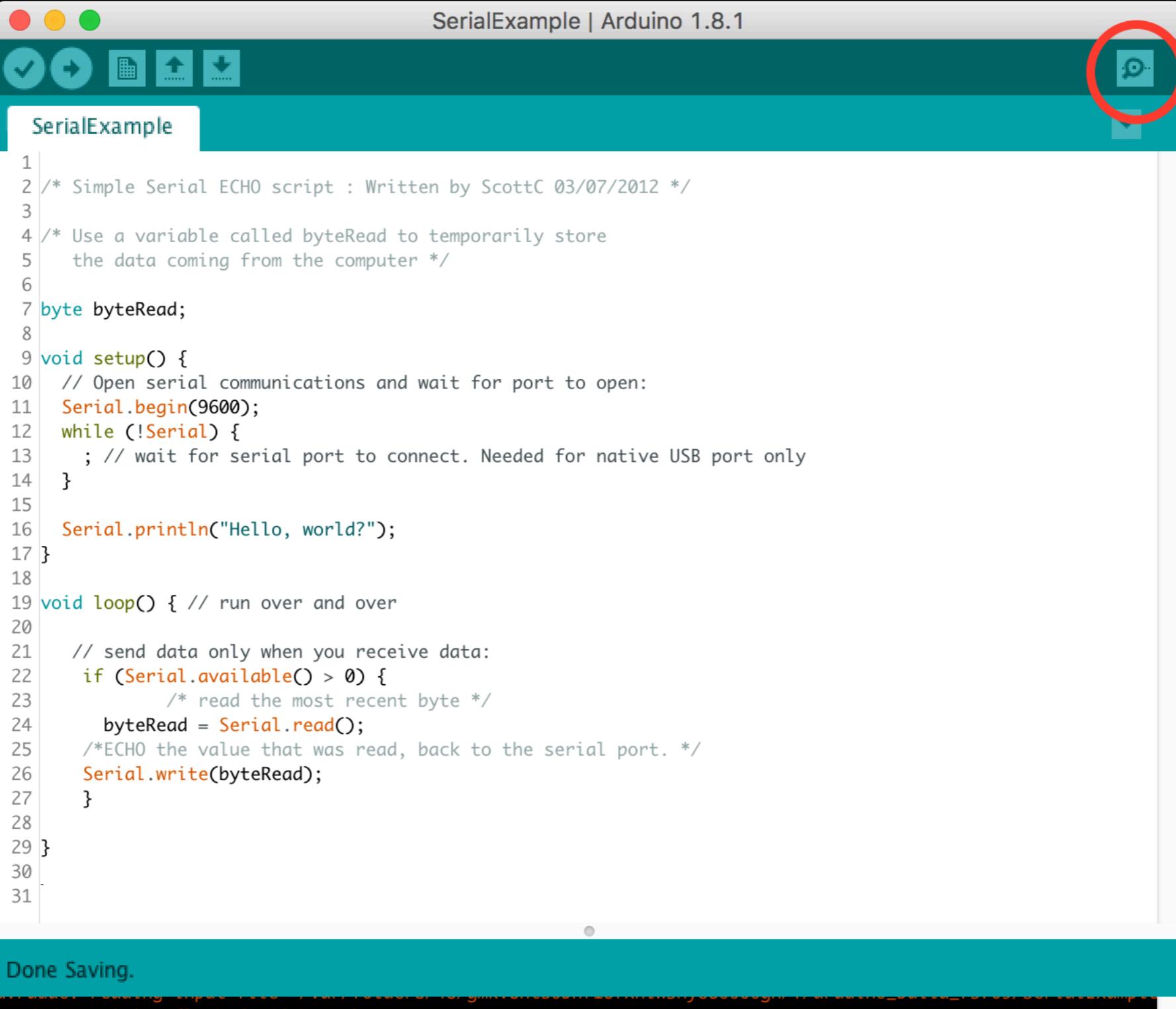
```
Debounce §
1 // constants won't change. They're used here to
2 // set pin numbers:
3 const int buttonPin = 2;      // the number of the pushbutton pin
4 const int ledPin = 13;        // the number of the LED pin
5
6 // Variables will change:
7 int ledState = HIGH;          // the current state of the output pin
8 int buttonState;              // the current reading from the input pin
9 int lastButtonState = LOW;    // the previous reading from the input pin
10
11 // the following variables are unsigned long's because the time, measured in milliseconds,
12 // will quickly become a bigger number than can be stored in an int.
13 unsigned long lastDebounceTime = 0; // the last time the output pin was toggled
14 unsigned long debounceDelay = 50;   // the debounce time; increase if the output flickers
15
16 void setup() {
17     pinMode(buttonPin, INPUT_PULLUP);
18     pinMode(ledPin, OUTPUT);
19
20     // set initial LED state
21     digitalWrite(ledPin, ledState);
22 }
23
24 void loop() {
25     // read the state of the switch into a local variable:
26     int reading = digitalRead(buttonPin);
27
28     // check to see if you just pressed the button
29     // (i.e. the input went from LOW to HIGH), and you've waited
30     // long enough since the last press to ignore any noise:
31
32     // If the switch changed, due to noise or pressing:
33     if (reading != lastButtonState) {
34         // reset the debouncing timer
35         lastDebounceTime = millis();
36     }
37
38     if ((millis() - lastDebounceTime) > debounceDelay) {
39         // whatever the reading is at, it's been there for longer
40         // than the debounce delay, so take it as the actual current state:
41
42         // if the button state has changed:
43         if (reading != buttonState) {
44             buttonState = reading;
45
46             // only toggle the LED if the new button state is HIGH
47             if (buttonState == HIGH) {
48                 ledState = !ledState;
49             }
50         }
51     }
52
53     // set the LED:
54     digitalWrite(ledPin, ledState);
55
56     // save the reading. Next time through the loop,
57     // it'll be the lastButtonState:
58     lastButtonState = reading;
59 }
60
61
62 }
```

Serial Communication



Information passes between the computer and Arduino through the USB cable. Information is transmitted as zeros ('0') and ones ('1')... also known as **bits!**

Serial Example

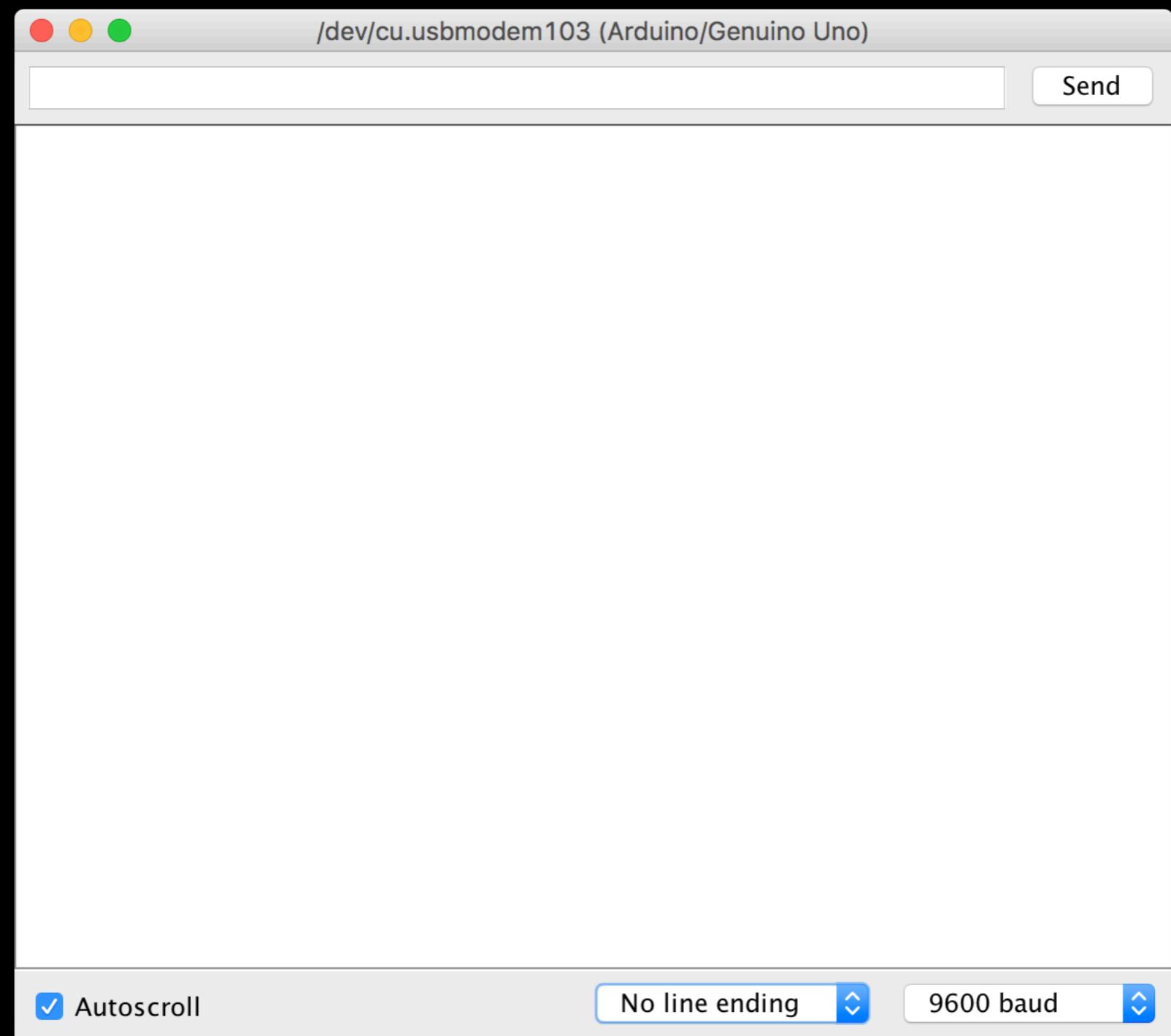


The screenshot shows the Arduino IDE interface with the title bar "SerialExample | Arduino 1.8.1". The toolbar at the top includes standard icons for file operations (New, Open, Save, Upload, Download) and a magnifying glass icon for search, which is circled in red. The main code editor window displays the "SerialExample" sketch:

```
1  /* Simple Serial ECHO script : Written by ScottC 03/07/2012 */
2
3  /* Use a variable called byteRead to temporarily store
4   the data coming from the computer */
5
6  byte byteRead;
7
8
9 void setup() {
10 // Open serial communications and wait for port to open:
11 Serial.begin(9600);
12 while (!Serial) {
13     ; // wait for serial port to connect. Needed for native USB port only
14 }
15
16 Serial.println("Hello, world?");
17 }
18
19 void loop() { // run over and over
20
21 // send data only when you receive data:
22 if (Serial.available() > 0) {
23     /* read the most recent byte */
24     byteRead = Serial.read();
25     /*ECHO the value that was read, back to the serial port. */
26     Serial.write(byteRead);
27 }
28
29 }
30
31 }
```

The status bar at the bottom of the IDE indicates "Done Saving." and shows memory usage information: "avr-gcc: writing flash (1476 bytes)".

Serial Monitor



Input Serial

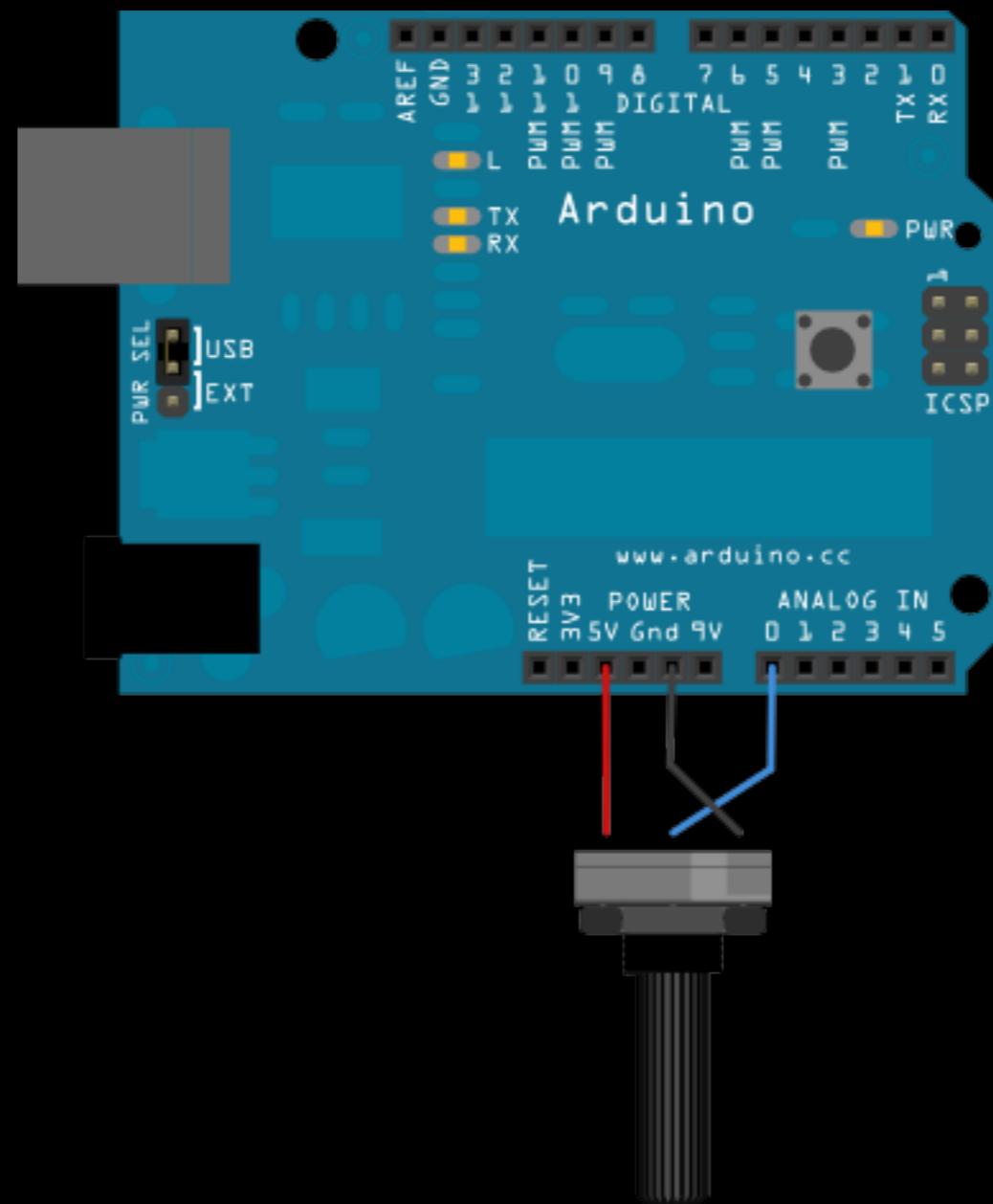


The screenshot shows the Arduino IDE interface with the title bar "InputSerial | Arduino 1.8.1". Below the title bar is a toolbar with icons for file operations (checkmark, arrow, file, upload, download) and a settings gear. The main area displays the code for the "InputSerial" sketch. The code uses the Serial library to read from a pushbutton connected to pin 2 and control an LED on pin 13. It includes comments explaining the logic for inverting the pushbutton's state due to its internal pull-up resistor.

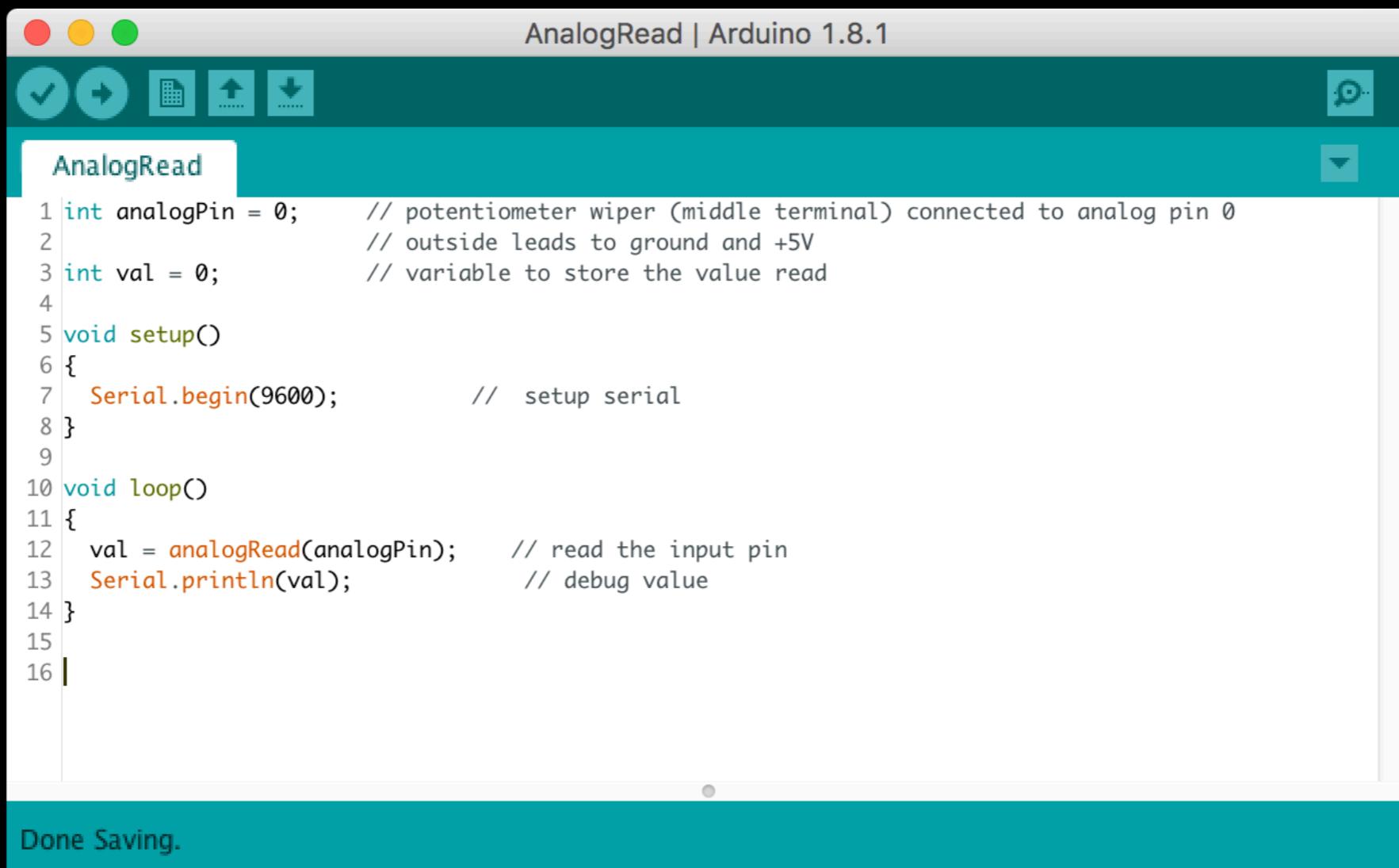
```
1
2 void setup() {
3     //start serial connection
4     Serial.begin(9600);
5     //configure pin2 as an input and enable the internal pull-up resistor
6     pinMode(2, INPUT_PULLUP);
7     pinMode(13, OUTPUT);
8
9 }
10
11 void loop() {
12     //read the pushbutton value into a variable
13     int sensorVal = digitalRead(2);
14     //print out the value of the pushbutton
15     Serial.println(sensorVal);
16
17     // Keep in mind the pullup means the pushbutton's
18     // logic is inverted. It goes HIGH when it's open,
19     // and LOW when it's pressed. Turn on pin 13 when the
20     // button's pressed, and off when it's not:
21     if (sensorVal == HIGH) {
22         digitalWrite(13, LOW);
23     } else {
24         digitalWrite(13, HIGH);
25     }
26 }
```

Done Saving.

Potentiometer



Analog Read



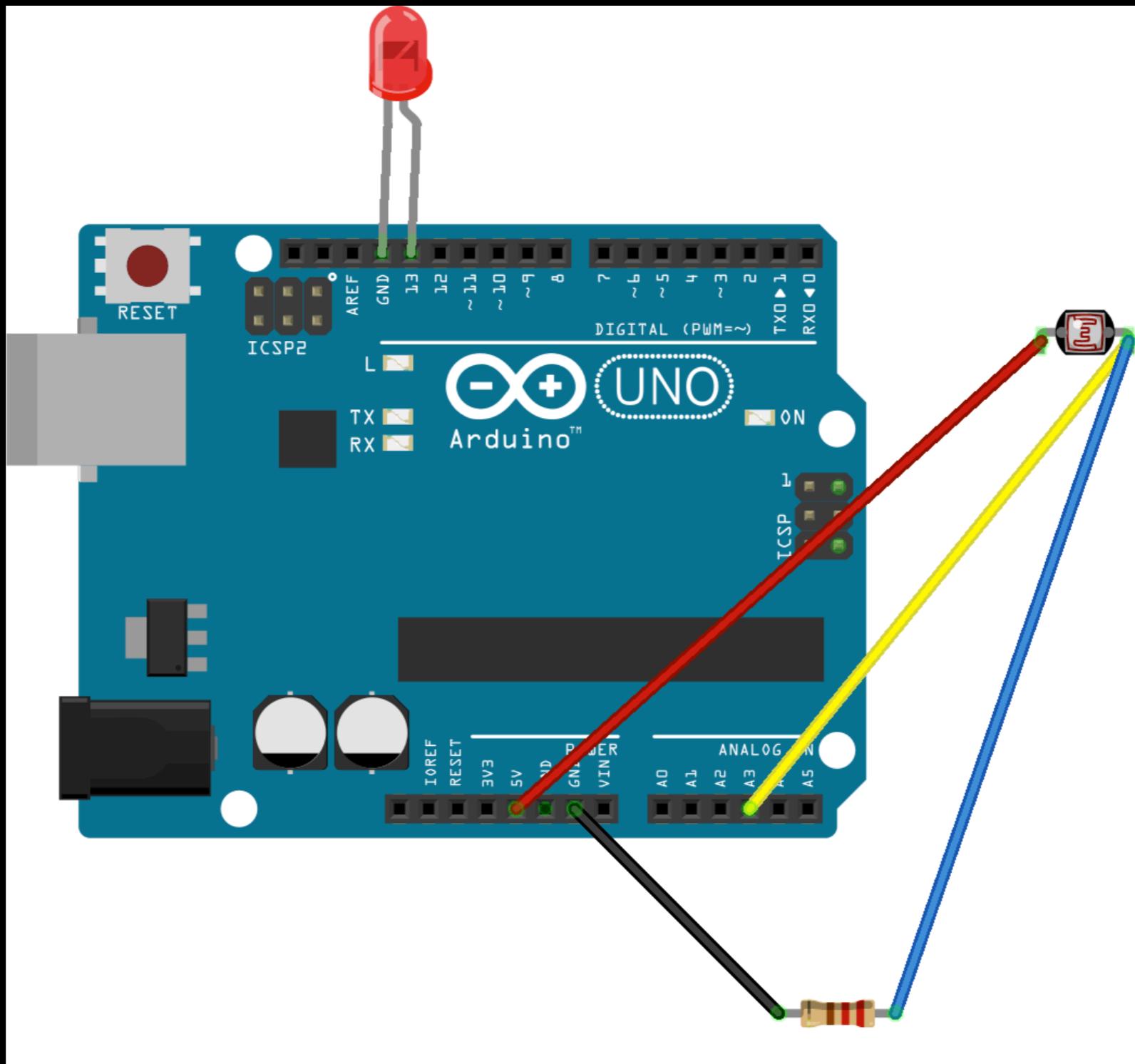
The screenshot shows the Arduino IDE interface with the title bar "AnalogRead | Arduino 1.8.1". The code editor contains the following sketch:

```
1 int analogPin = 0;      // potentiometer wiper (middle terminal) connected to analog pin 0
2                               // outside leads to ground and +5V
3 int val = 0;              // variable to store the value read
4
5 void setup()
6 {
7   Serial.begin(9600);      // setup serial
8 }
9
10 void loop()
11 {
12   val = analogRead(analogPin); // read the input pin
13   Serial.println(val);       // debug value
14 }
```

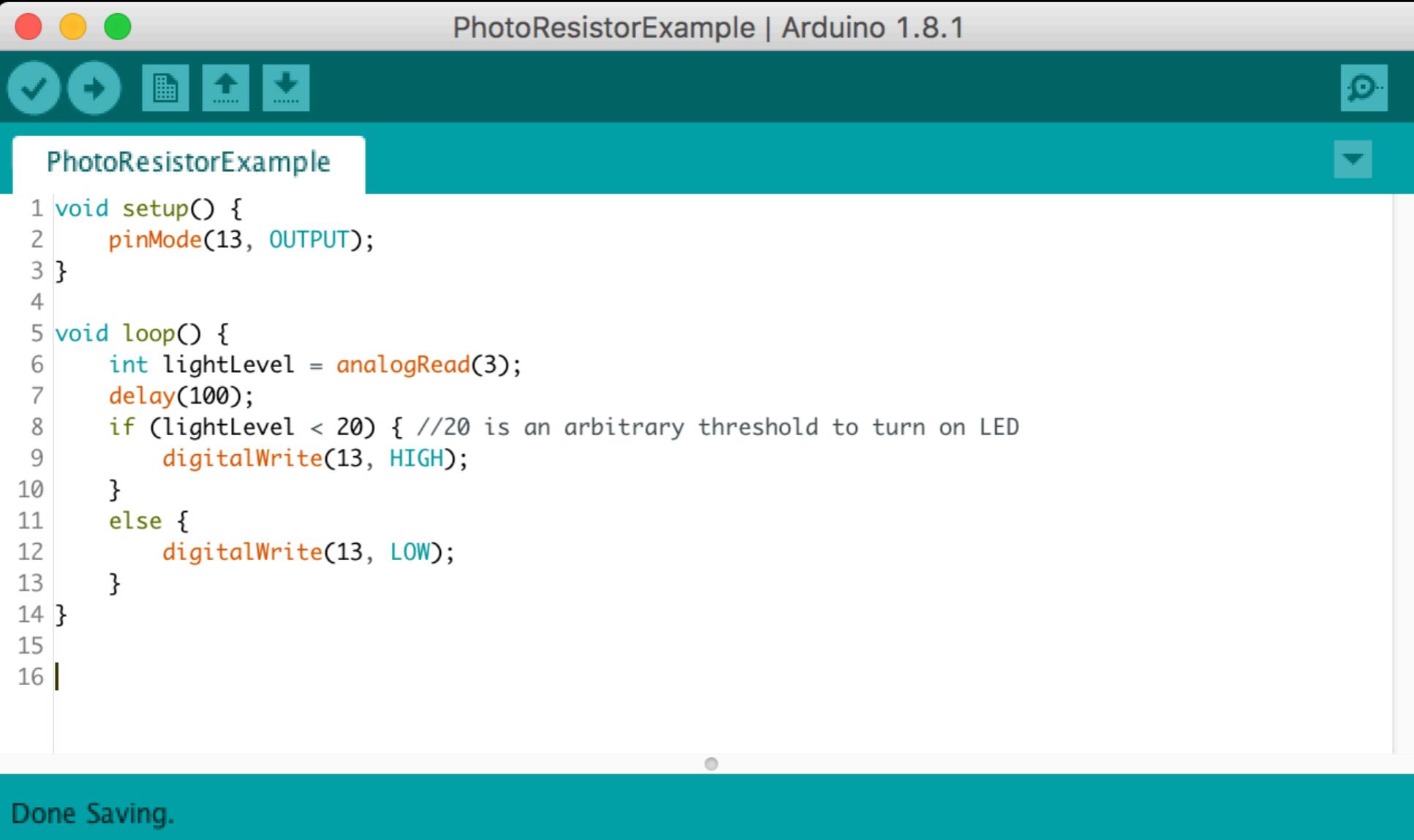
The status bar at the bottom of the IDE displays the message "Done Saving."

Components

Photoresistor



Photoresistor

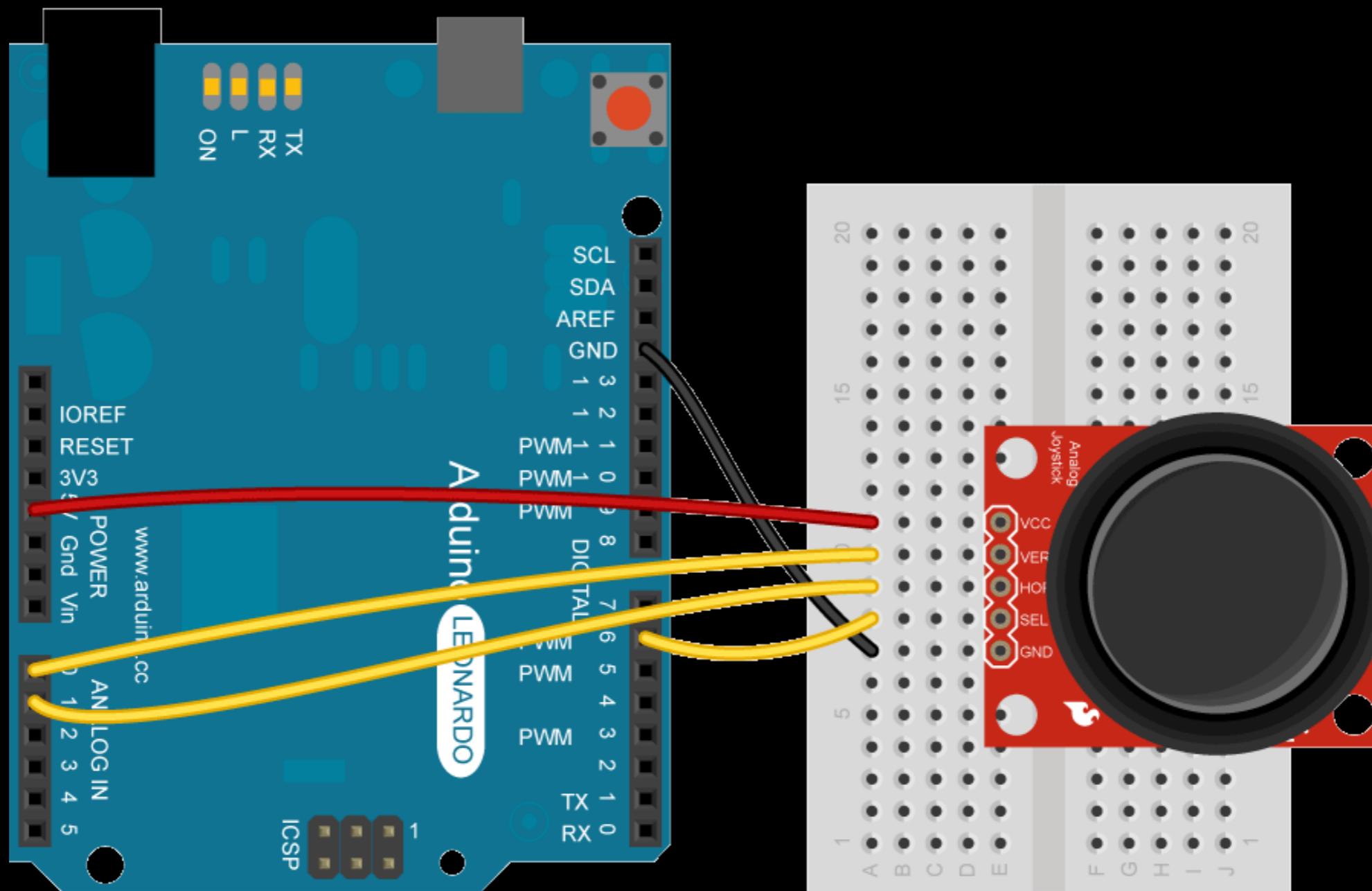


The screenshot shows the Arduino IDE interface with the title bar "PhotoResistorExample | Arduino 1.8.1". The central code editor window displays the following C++ code for a photoresistor example:

```
1 void setup() {
2     pinMode(13, OUTPUT);
3 }
4
5 void loop() {
6     int lightLevel = analogRead(3);
7     delay(100);
8     if (lightLevel < 20) { //20 is an arbitrary threshold to turn on LED
9         digitalWrite(13, HIGH);
10    }
11    else {
12        digitalWrite(13, LOW);
13    }
14 }
```

The status bar at the bottom of the IDE indicates "Done Saving.".

Joystick



Joystick



The screenshot shows the Arduino IDE interface with a sketch named "Joystick". The code uses analog pins 0 and 1 to read values from two variable resistors connected to digital pins 0 and 1 respectively. It then prints the treated values to the Serial monitor.

```
int ledPin = 13;
int joyPin1 = 0; // slider variable connected to analog pin 0
int joyPin2 = 1; // slider variable connected to analog pin 1
int value1 = 0; // variable to read the value from the analog pin 0
int value2 = 0; // variable to read the value from the analog pin 1

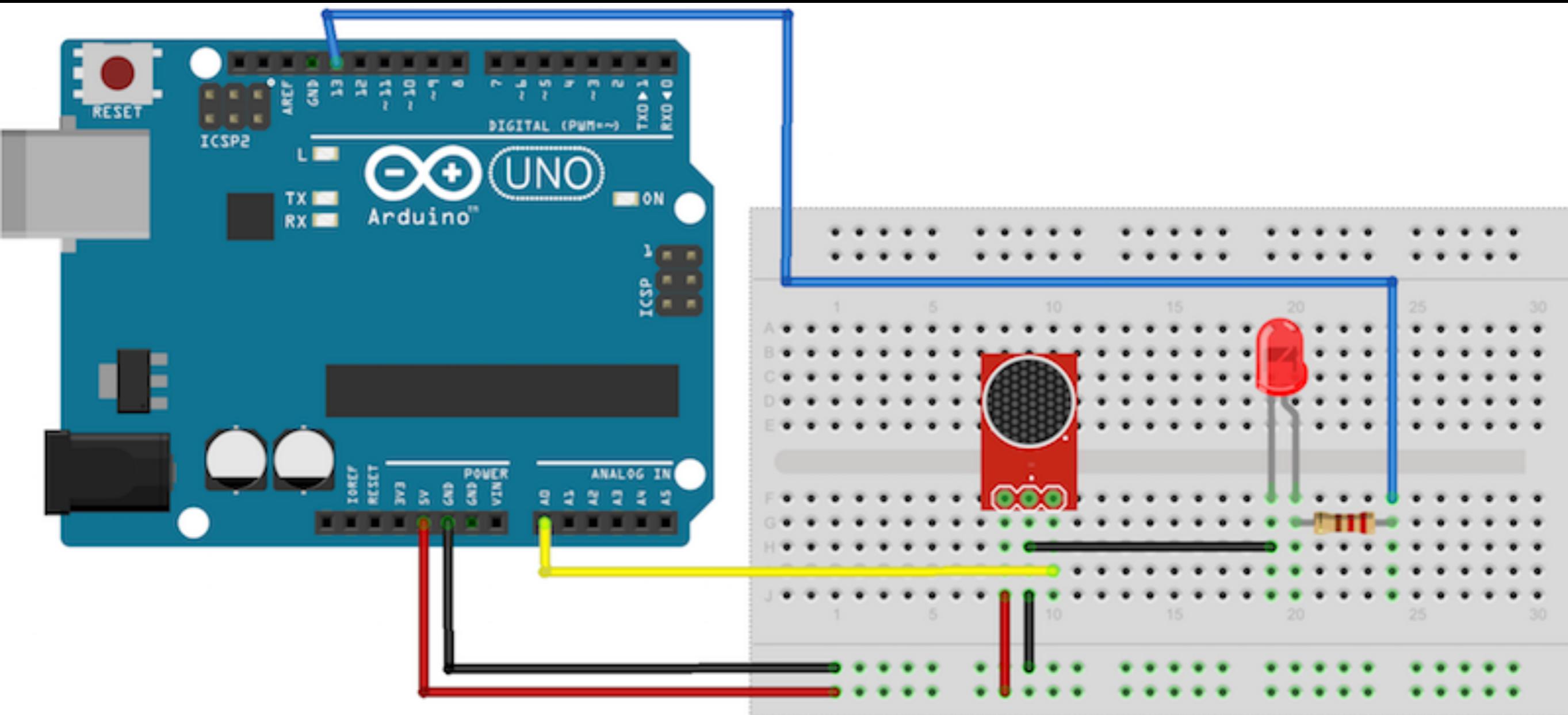
void setup() {
  pinMode(ledPin, OUTPUT); // initializes digital pins 0 to 7 as outputs
  Serial.begin(9600);
}

int treatValue(int data) {
  return (data * 9 / 1024) + 48;
}

void loop() {
  // reads the value of the variable resistor
  value1 = analogRead(joyPin1);
  // this small pause is needed between reading
  // analog pins, otherwise we get the same value twice
  delay(100);
  // reads the value of the variable resistor
  value2 = analogRead(joyPin2);

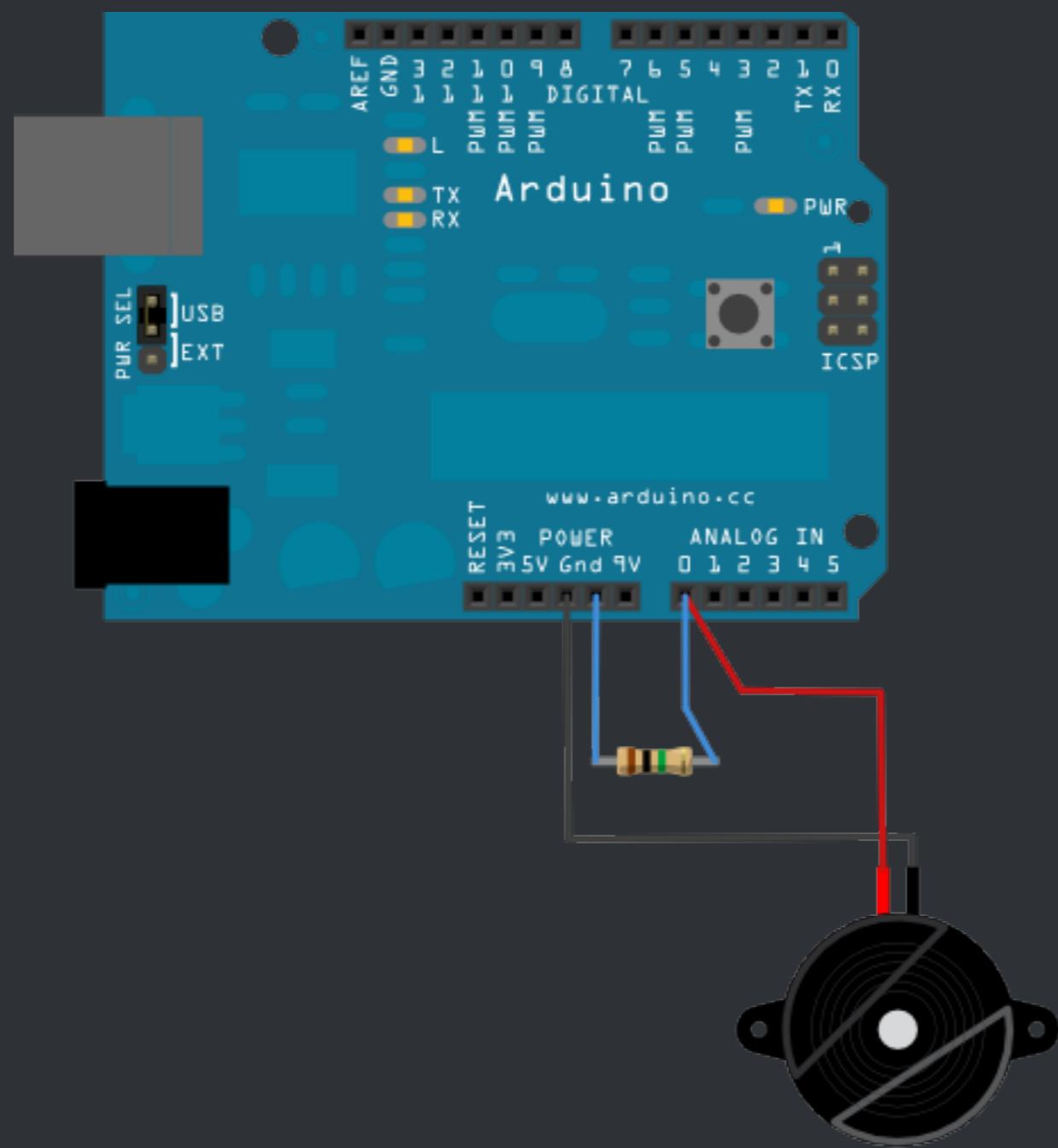
  digitalWrite(ledPin, HIGH);
  delay(value1);
  digitalWrite(ledPin, LOW);
  delay(value2);
  Serial.print('J');
  Serial.print(treatValue(value1));
  Serial.println(treatValue(value2));
}
```

Sound Sensor



fritzing

Piezo Element



Piezo Element



The screenshot shows the Arduino IDE interface with a sketch named "Piezo". The code implements a simple sound detection system using a piezo element connected to analog pin A0. It reads the sensor value, compares it to a threshold, and toggles an LED on digital pin 13 in response to detected sound.

```
1 // these constants won't change:
2 const int ledPin = 13;      // led connected to digital pin 13
3 const int knockSensor = A0; // the piezo is connected to analog pin 0
4 const int threshold = 100; // threshold value to decide when the detected sound is a knock or not
5
6
7 // these variables will change:
8 int sensorReading = 0;      // variable to store the value read from the sensor pin
9 int ledState = LOW;         // variable used to store the last LED status, to toggle the light
10
11 void setup() {
12   pinMode(ledPin, OUTPUT); // declare the ledPin as as OUTPUT
13   Serial.begin(9600);     // use the serial port
14 }
15
16
17 void loop() {
18   // read the sensor and store it in the variable sensorReading:
19   sensorReading = analogRead(knockSensor);
20
21   // if the sensor reading is greater than the threshold:
22   if (sensorReading >= threshold) {
23     // toggle the status of the ledPin:
24     ledState = !ledState;
25     // update the LED pin itself:
26     digitalWrite(ledPin, ledState);
27     // send the string "Knock!" back to the computer, followed by newline
28     Serial.println("Knock!");
29   }
30   delay(100); // delay to avoid overloading the serial port buffer
31 }
32
33
```

Done Saving.

Piezo Element

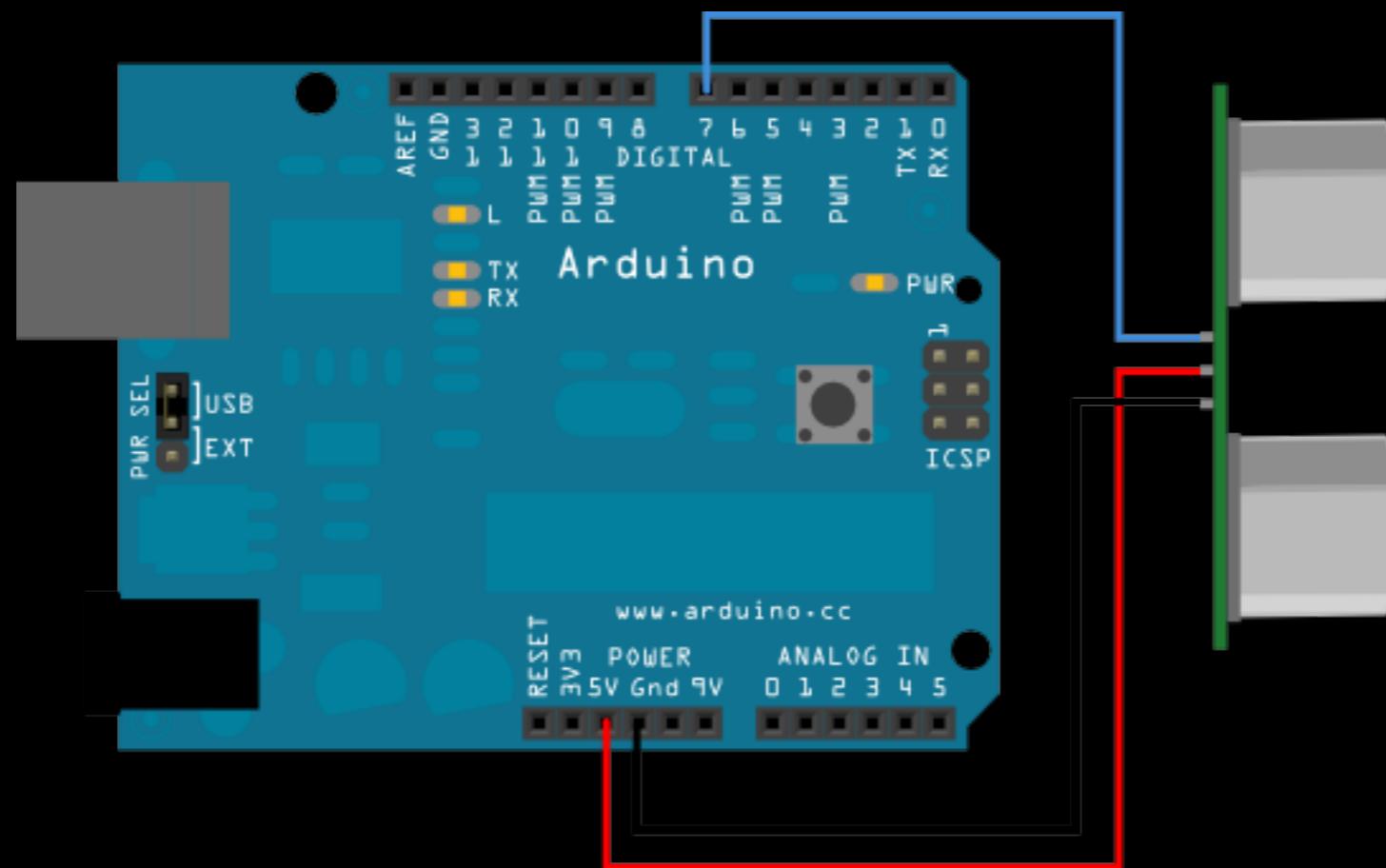


The screenshot shows the Arduino IDE interface with a sketch named "Piezo". The code is written in C++ and controls a piezo element connected to analog pin A0. It reads the sensor value, compares it to a threshold, and toggles an LED on digital pin 13 based on the reading. The sketch also prints "Knock!" to the serial port.

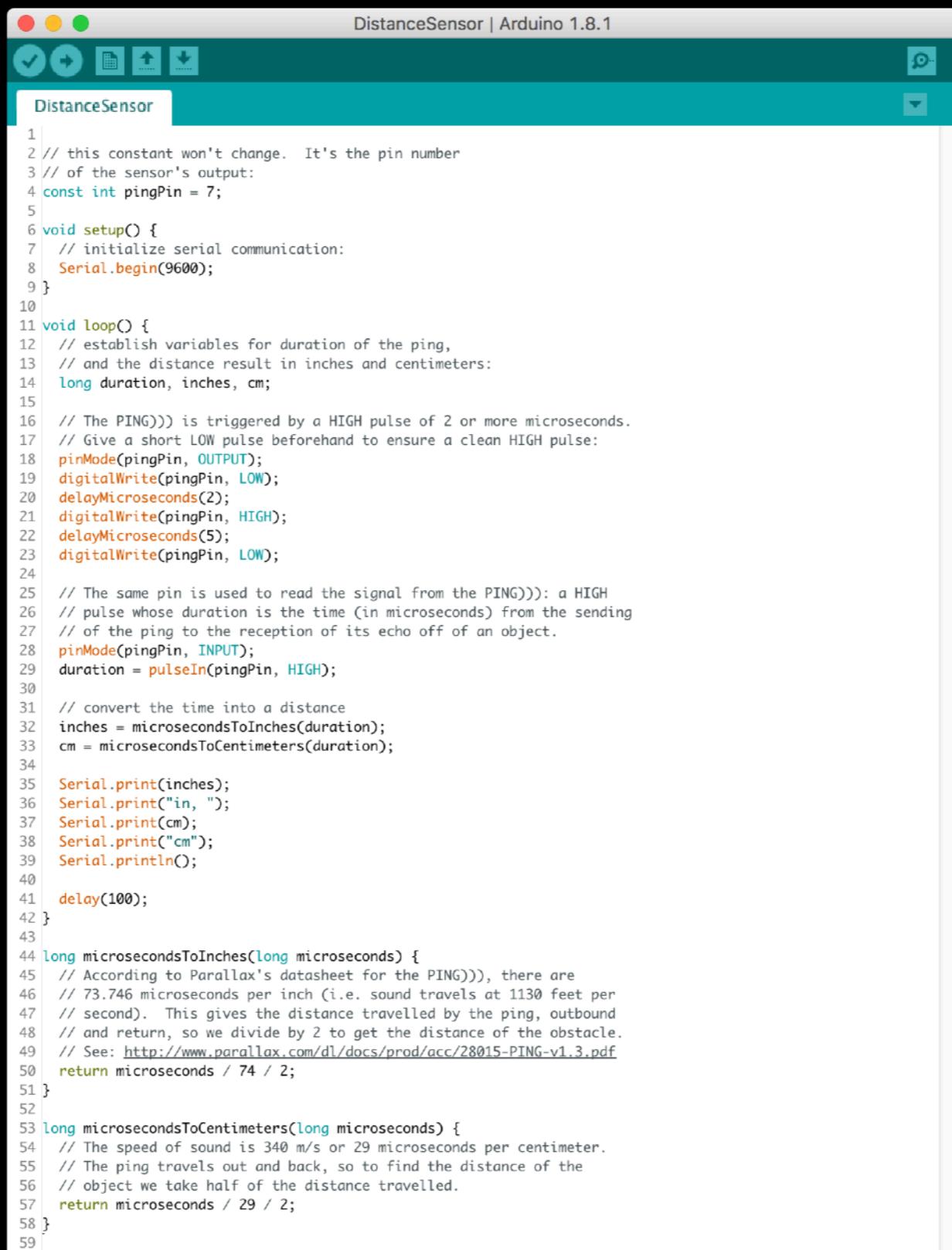
```
1 // these constants won't change:
2 const int ledPin = 13;      // led connected to digital pin 13
3 const int knockSensor = A0; // the piezo is connected to analog pin 0
4 const int threshold = 100; // threshold value to decide when the detected sound is a knock or not
5
6
7 // these variables will change:
8 int sensorReading = 0;      // variable to store the value read from the sensor pin
9 int ledState = LOW;         // variable used to store the last LED status, to toggle the light
10
11 void setup() {
12   pinMode(ledPin, OUTPUT); // declare the ledPin as as OUTPUT
13   Serial.begin(9600);     // use the serial port
14 }
15
16
17 void loop() {
18   // read the sensor and store it in the variable sensorReading:
19   sensorReading = analogRead(knockSensor);
20
21   // if the sensor reading is greater than the threshold:
22   if (sensorReading >= threshold) {
23     // toggle the status of the ledPin:
24     ledState = !ledState;
25     // update the LED pin itself:
26     digitalWrite(ledPin, ledState);
27     // send the string "Knock!" back to the computer, followed by newline
28     Serial.println("Knock!");
29   }
30   delay(100); // delay to avoid overloading the serial port buffer
31 }
32
33
```

Done Saving.

Distance Sensor



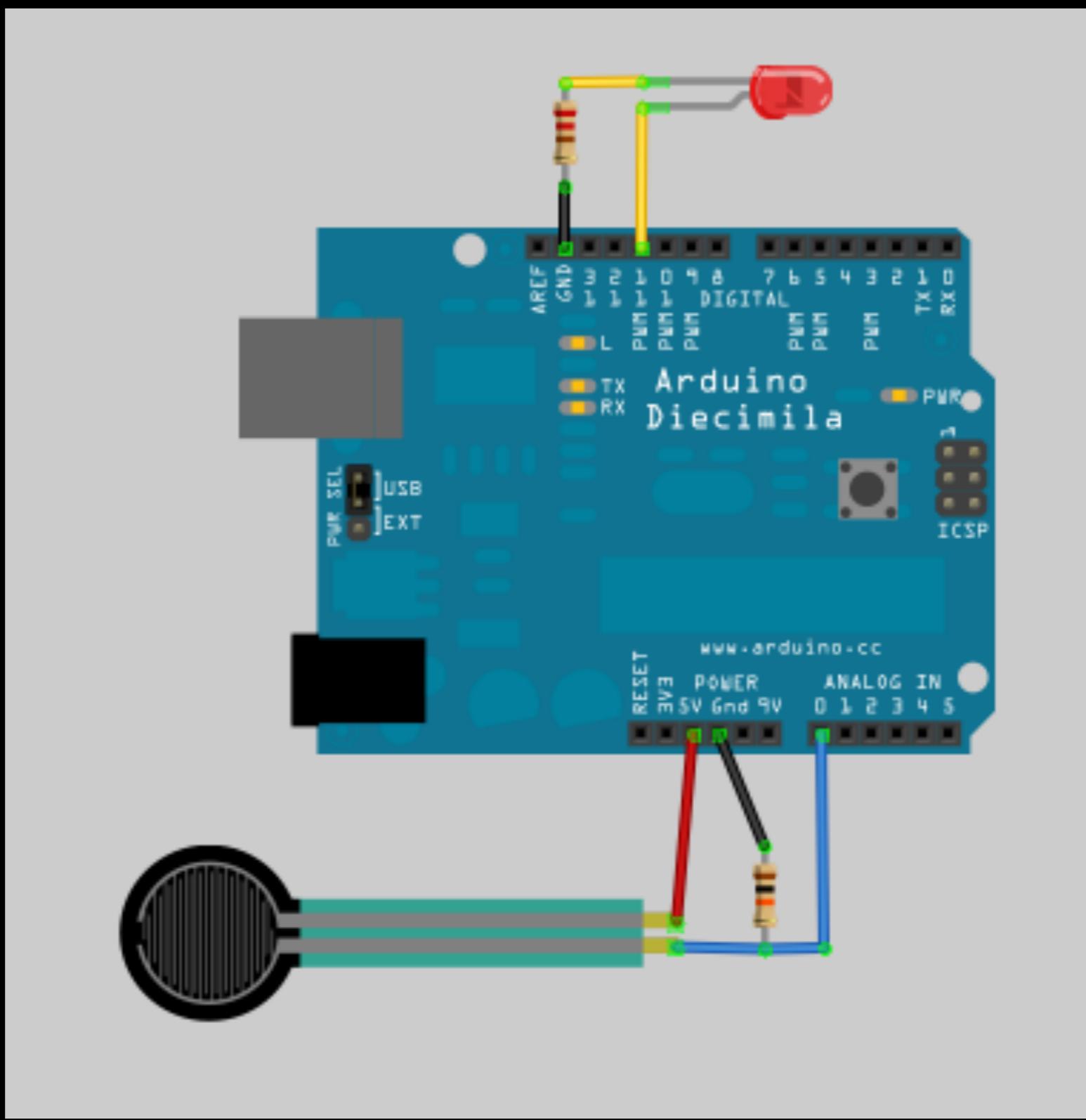
Distance Sensor



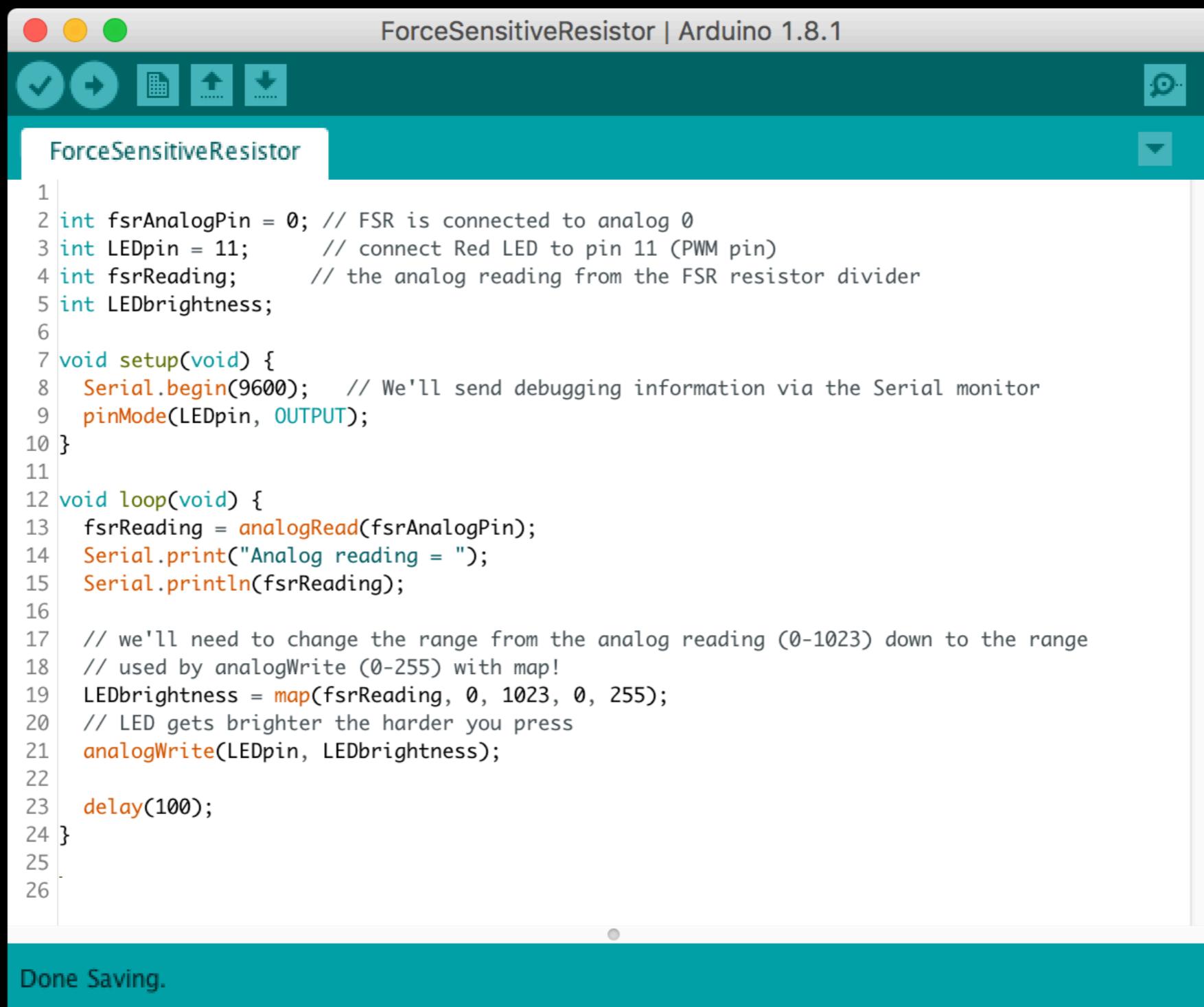
The screenshot shows the Arduino IDE interface with the title bar "DistanceSensor | Arduino 1.8.1". The main window displays the "DistanceSensor" sketch. The code is a C++ program designed to read distance measurements from a PING ultrasonic sensor connected to pin 7. It uses the Serial port for communication at 9600 baud. The sketch includes setup() and loop() functions, along with two helper functions: microsecondsToInches() and microsecondsToCentimeters(). The code is color-coded for readability, with keywords in blue, comments in green, and variables in black.

```
1 // this constant won't change. It's the pin number
2 // of the sensor's output:
3 const int pingPin = 7;
4
5 void setup() {
6     // initialize serial communication:
7     Serial.begin(9600);
8 }
9
10 void loop() {
11     // establish variables for duration of the ping,
12     // and the distance result in inches and centimeters:
13     long duration, inches, cm;
14
15     // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
16     // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
17     pinMode(pingPin, OUTPUT);
18     digitalWrite(pingPin, LOW);
19     delayMicroseconds(2);
20     digitalWrite(pingPin, HIGH);
21     delayMicroseconds(5);
22     digitalWrite(pingPin, LOW);
23
24     // The same pin is used to read the signal from the PING))): a HIGH
25     // pulse whose duration is the time (in microseconds) from the sending
26     // of the ping to the reception of its echo off of an object.
27     pinMode(pingPin, INPUT);
28     duration = pulseIn(pingPin, HIGH);
29
30     // convert the time into a distance
31     inches = microsecondsToInches(duration);
32     cm = microsecondsToCentimeters(duration);
33
34     Serial.print(inches);
35     Serial.print("in, ");
36     Serial.print(cm);
37     Serial.print("cm");
38     Serial.println();
39
40     delay(100);
41 }
42
43 long microsecondsToInches(long microseconds) {
44     // According to Parallax's datasheet for the PING))), there are
45     // 73.746 microseconds per inch (i.e. sound travels at 1130 feet per
46     // second). This gives the distance travelled by the ping, outbound
47     // and return, so we divide by 2 to get the distance of the obstacle.
48     // See: http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf
49     return microseconds / 74 / 2;
50 }
51
52
53 long microsecondsToCentimeters(long microseconds) {
54     // The speed of sound is 340 m/s or 29 microseconds per centimeter.
55     // The ping travels out and back, so to find the distance of the
56     // object we take half of the distance travelled.
57     return microseconds / 29 / 2;
58 }
```

Force Sensing Resistor



Force Sensing Resistor

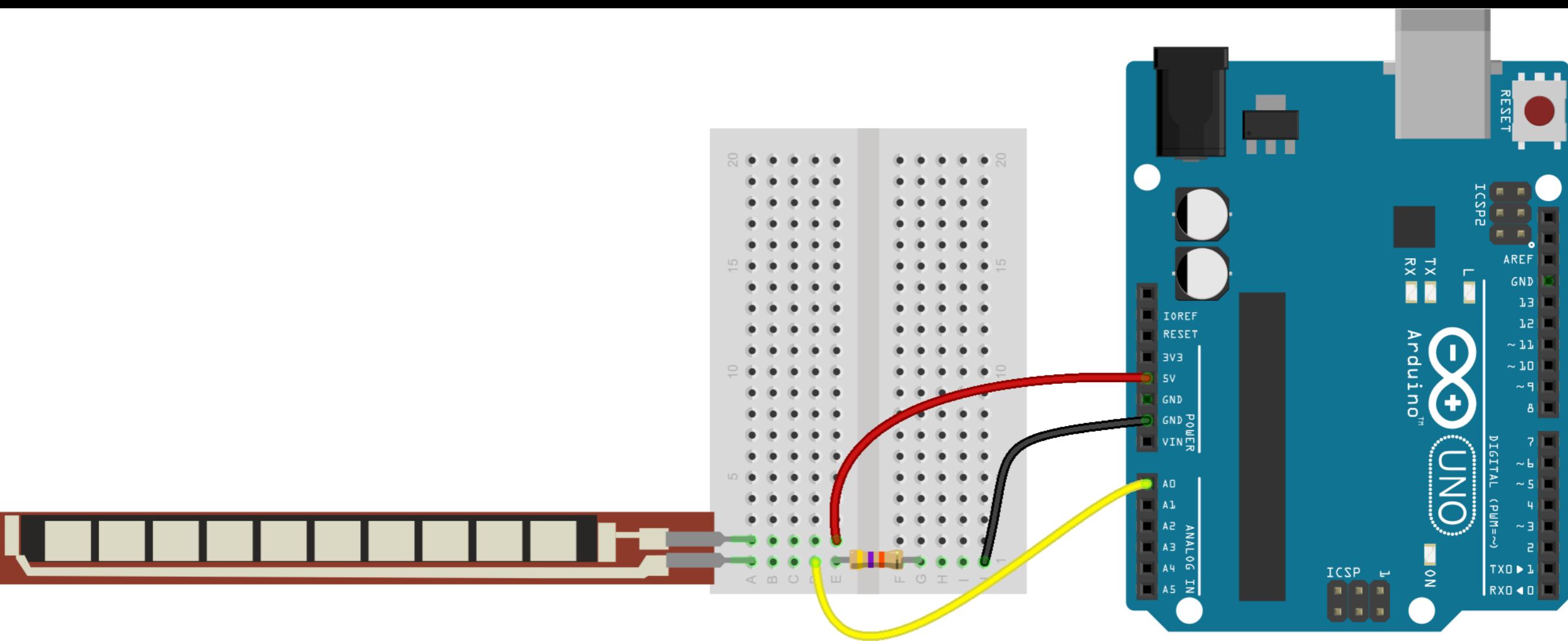


The screenshot shows the Arduino IDE interface with a sketch named "ForceSensitiveResistor". The code implements a simple circuit where a Force Sensitive Resistor (FSR) is connected to an analog input pin (0), and a red LED is connected to digital pin 11 (set as an output). The sketch reads the analog value from the FSR, prints it to the Serial monitor, and then maps this value to a brightness level for the LED using the `map()` function. The LED's brightness increases as pressure is applied to the FSR.

```
1 int fsrAnalogPin = 0; // FSR is connected to analog 0
2 int LEDpin = 11;      // connect Red LED to pin 11 (PWM pin)
3 int fsrReading;      // the analog reading from the FSR resistor divider
4 int LEDbrightness;
5
6
7 void setup(void) {
8   Serial.begin(9600); // We'll send debugging information via the Serial monitor
9   pinMode(LEDpin, OUTPUT);
10 }
11
12 void loop(void) {
13   fsrReading = analogRead(fsrAnalogPin);
14   Serial.print("Analog reading = ");
15   Serial.println(fsrReading);
16
17   // we'll need to change the range from the analog reading (0-1023) down to the range
18   // used by analogWrite (0-255) with map!
19   LEDbrightness = map(fsrReading, 0, 1023, 0, 255);
20   // LED gets brighter the harder you press
21   analogWrite(LEDpin, LEDbrightness);
22
23   delay(100);
24 }
```

Done Saving.

Flex Sensor



fritzing

Flex Sensor



The screenshot shows the Arduino IDE interface with the title bar "FlexSensor | Arduino 1.8.1". Below the title bar is a toolbar with icons for file operations (checkmark, arrow, folder, up, down) and a magnifying glass. The main area contains the "FlexSensor" code:

```
1 const int FLEX_PIN = A0; // Pin connected to voltage divider output
2
3 // Measure the voltage at 5V and the actual resistance of your
4 // 47k resistor, and enter them below:
5 const float VCC = 4.98; // Measured voltage of Arduino 5V line
6 const float R_DIV = 47500.0; // Measured resistance of 3.3k resistor
7
8 // Upload the code, then try to adjust these values to more
9 // accurately calculate bend degree.
10 const float STRAIGHT_RESISTANCE = 37300.0; // resistance when straight
11 const float BEND_RESISTANCE = 90000.0; // resistance at 90 deg
12
13 void setup()
14 {
15   Serial.begin(9600);
16   pinMode(FLEX_PIN, INPUT);
17 }
18
19 void loop()
20 {
21   // Read the ADC, and calculate voltage and resistance from it
22   int flexADC = analogRead(FLEX_PIN);
23   float flexV = flexADC * VCC / 1023.0;
24   float flexR = R_DIV * (VCC / flexV - 1.0);
25   Serial.println("Resistance: " + String(flexR) + " ohms");
26
27   // Use the calculated resistance to estimate the sensor's
28   // bend angle:
29   float angle = map(flexR, STRAIGHT_RESISTANCE, BEND_RESISTANCE,
30                     0, 90.0);
31   Serial.println("Bend: " + String(angle) + " degrees");
32   Serial.println();
33
34   delay(500);
35 }
36
37
```

At the bottom of the code editor, a message "Done Saving." is displayed.

Questions?

Imanol Gómez

imanolgomez.net

yo@imanolgomez.net