HECK YEA, IT'S CCLAB!

Pull up your projects!

Oh, hello Arduino

WHAT IS IT?

Arduino

Arduino is a single-board microcontroller, intended to make the application of interactive objects or environments more accessible. The hardware consists of an open-source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM.

Arduino

Arduino

tl;dr

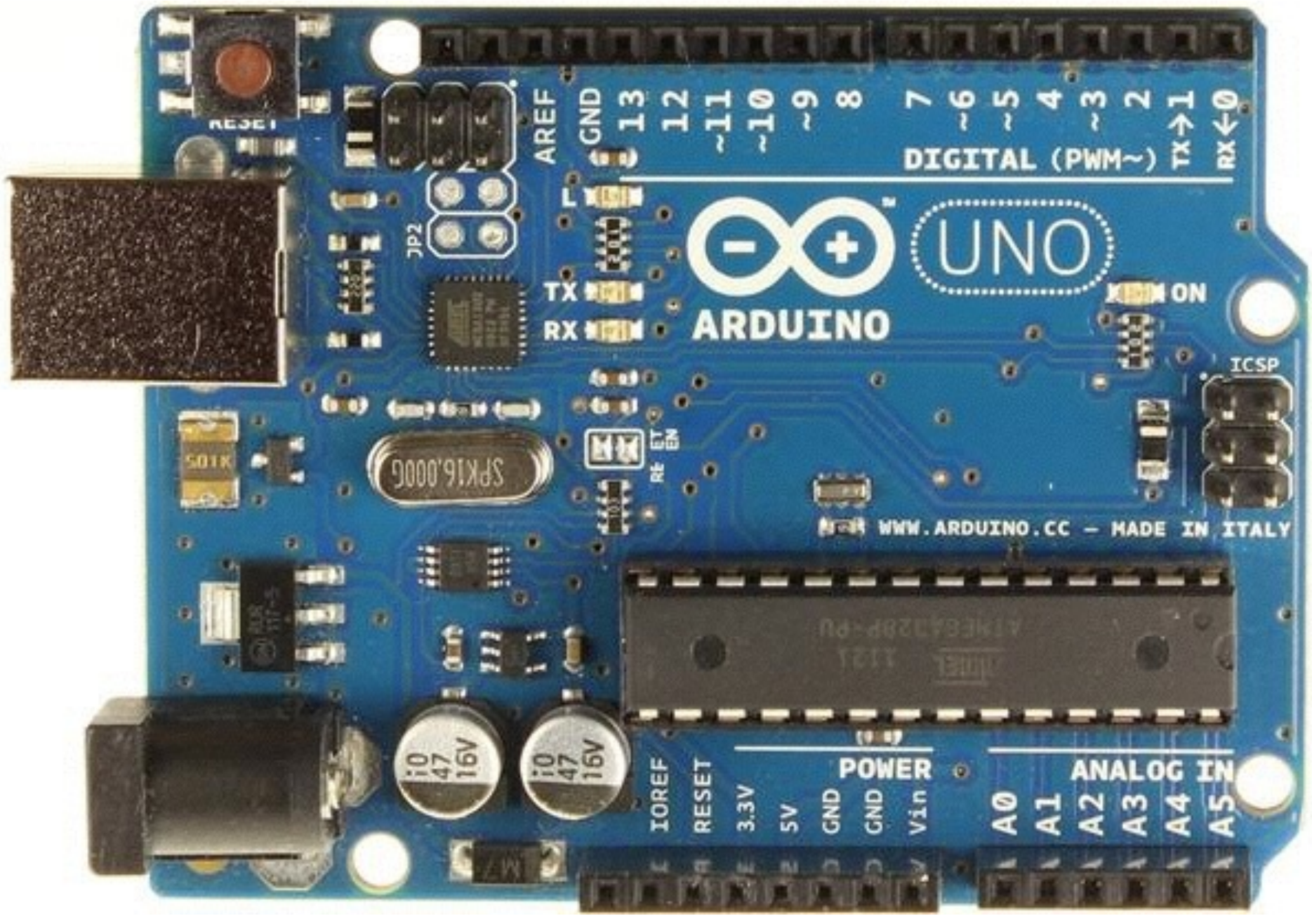
microcontroller
open source
rapid prototyping
(without the compsci degree)

Arduino

input - output machine

WHAT DOES IT DO...?

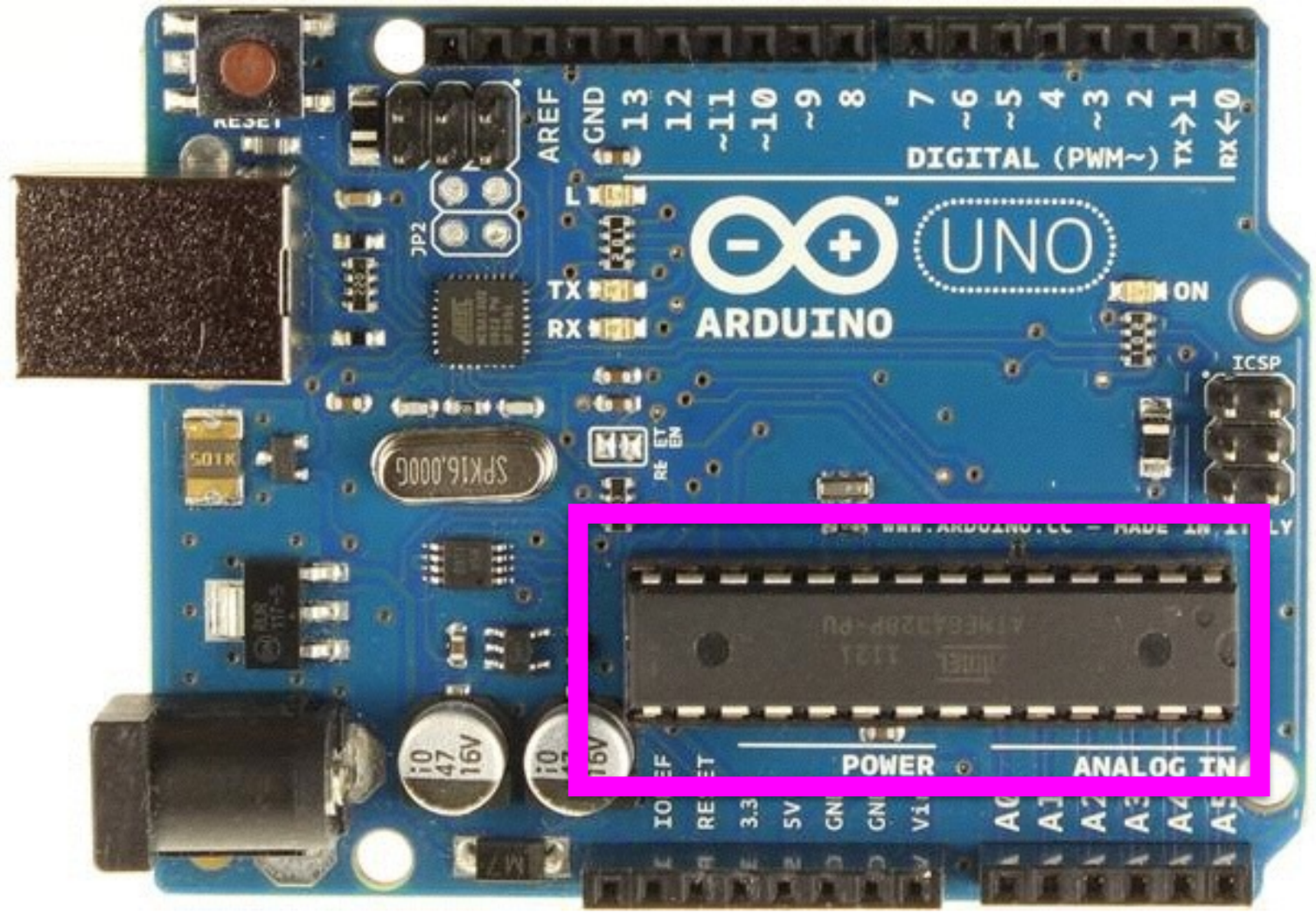
Arduino


RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
JP2
TX
RX
ARDUINO
UNO
ON
ICSP
16.000G
WWW.ARDUINO.CC – MADE IN ITALY
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

Arduino

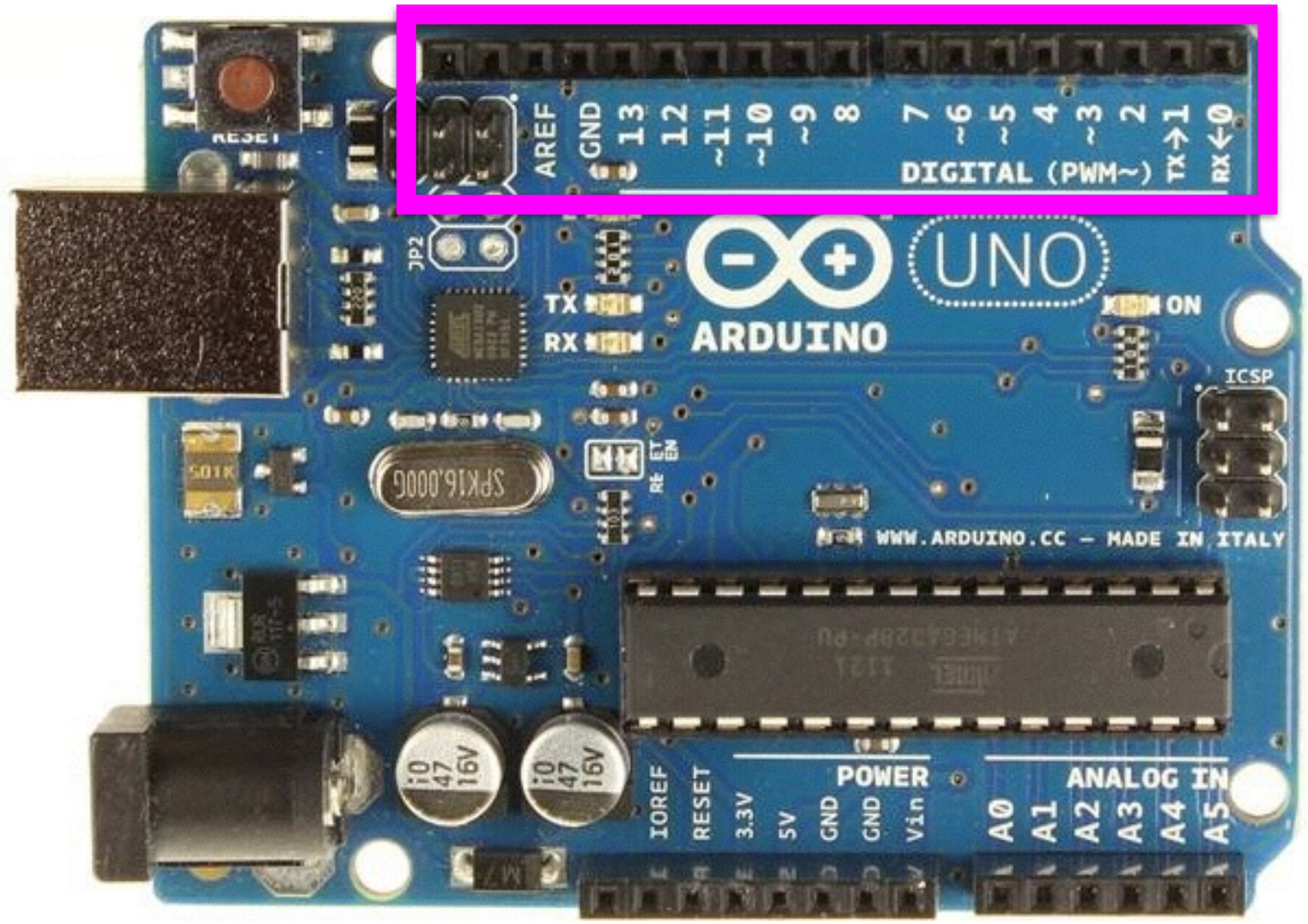
the brain

ATmega 328p chip



Arduino

digital pins

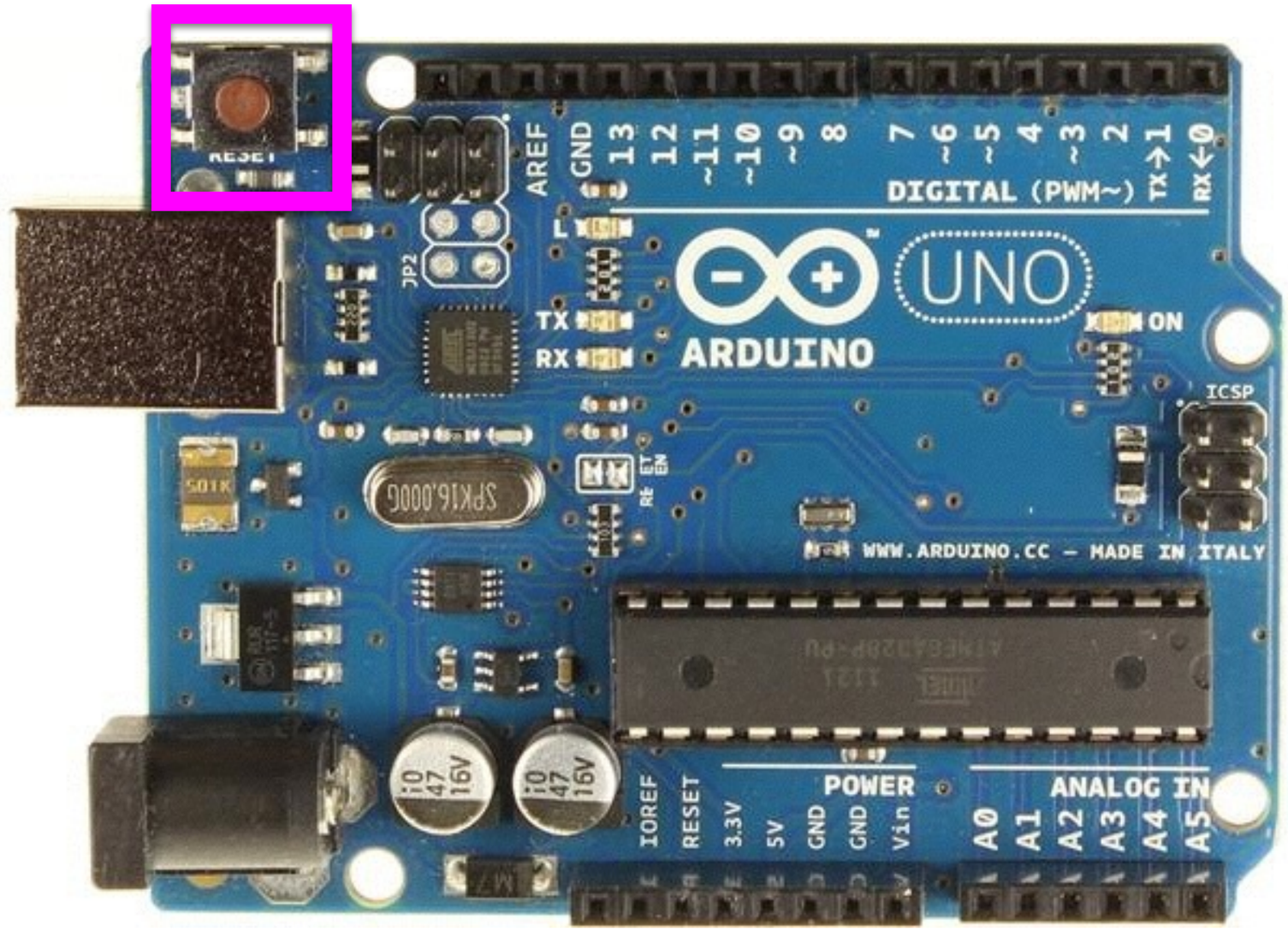


Arduino

analog pins

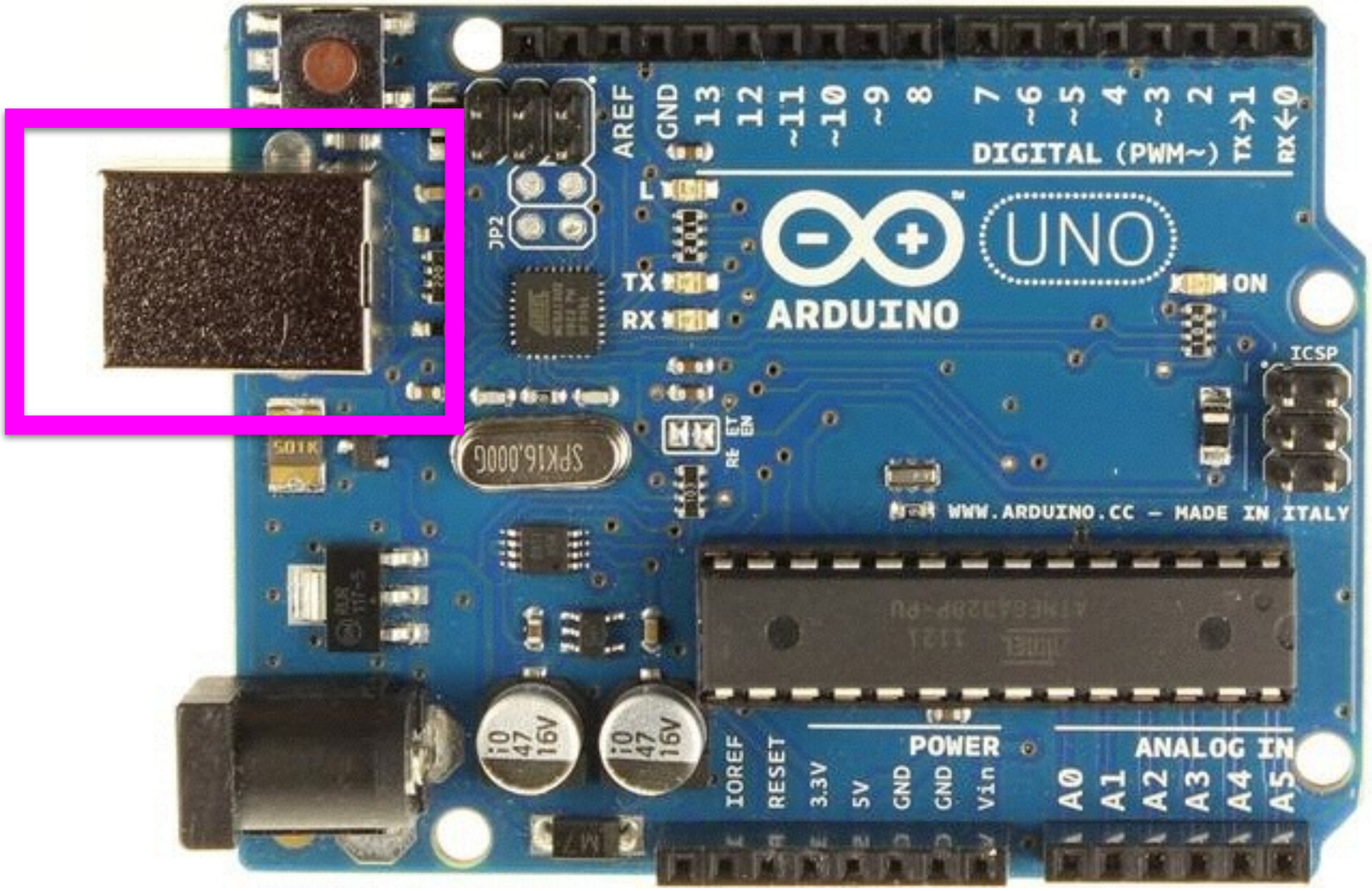
Arduino

reset button



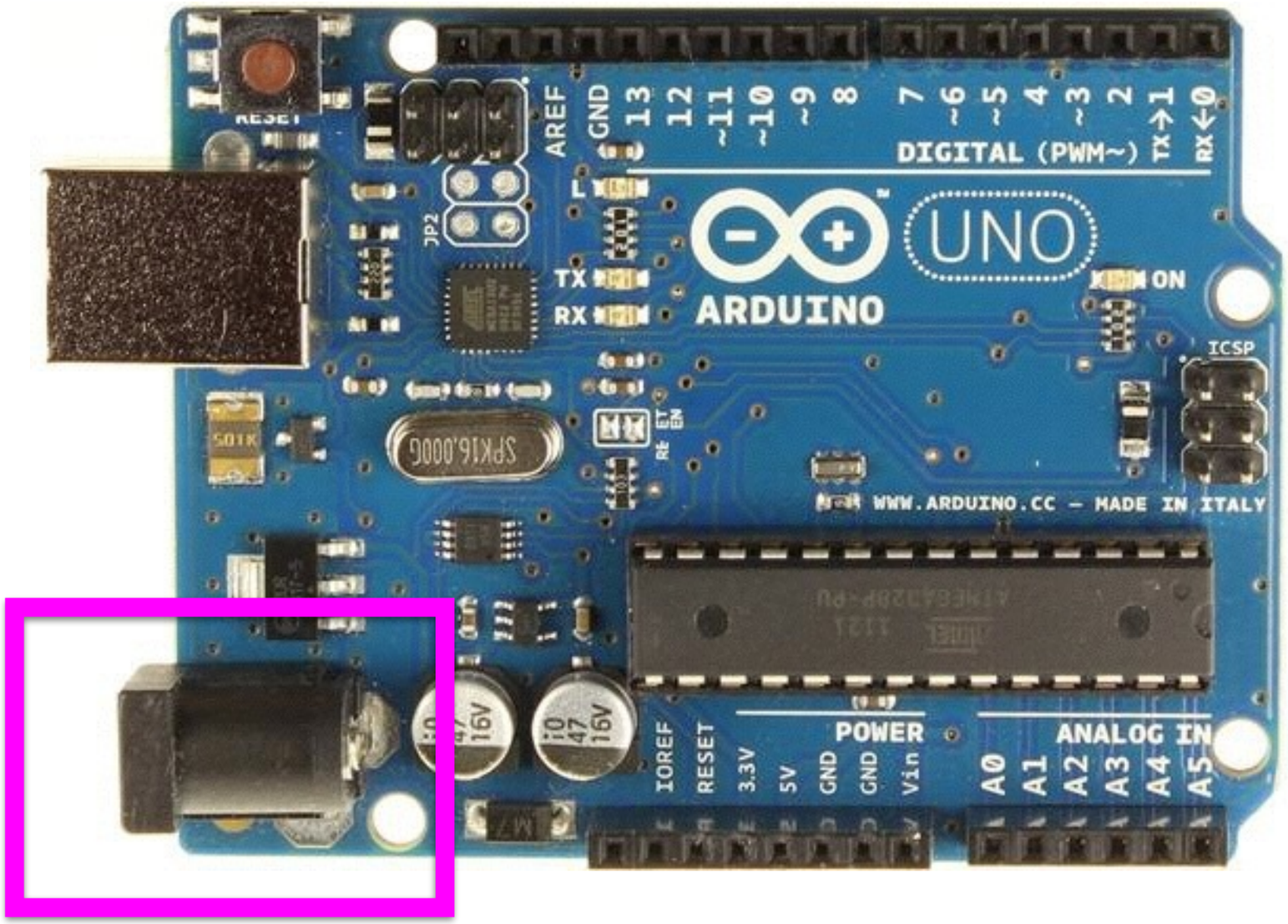
Arduino

usb port



Arduino

power jack

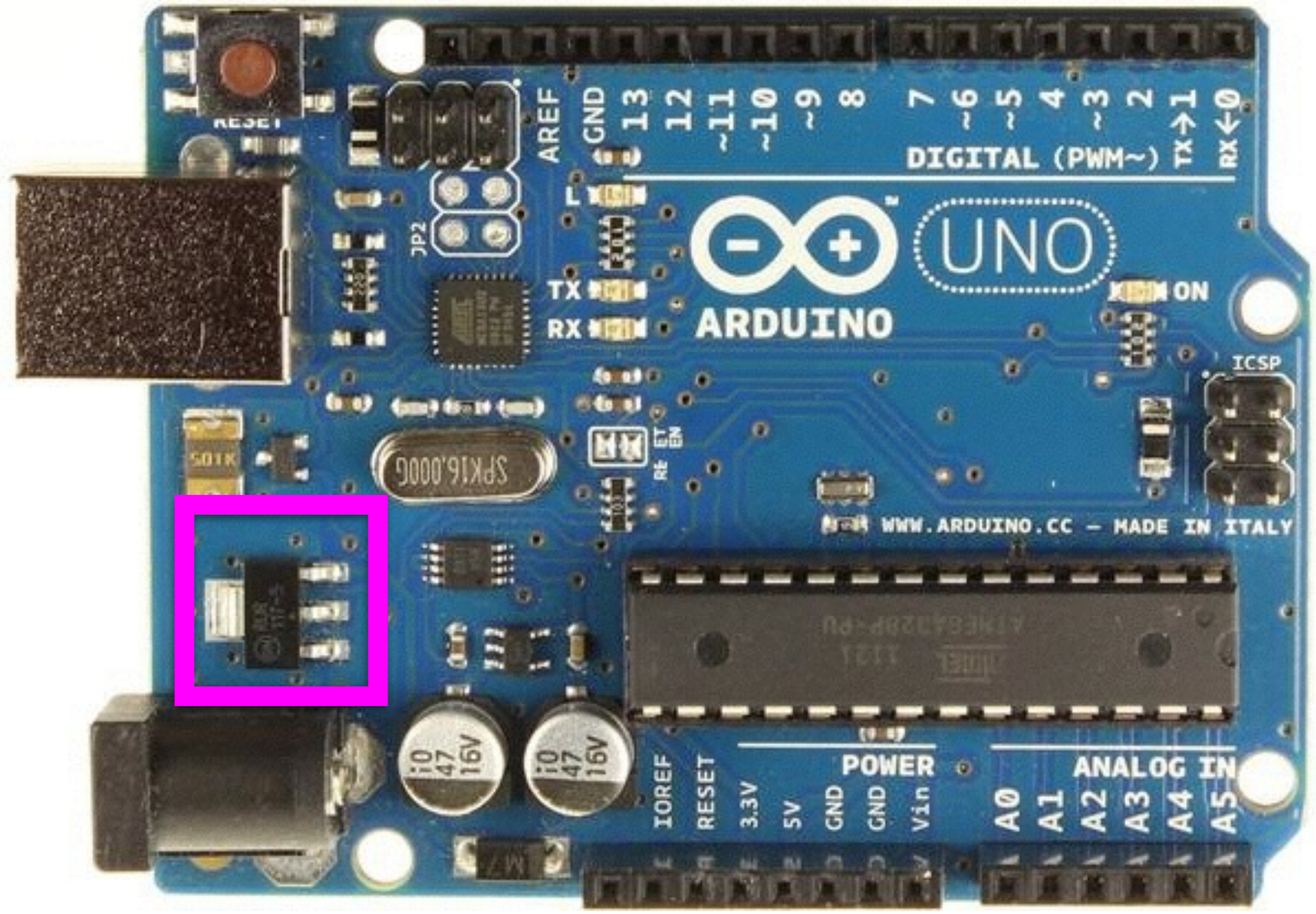


pro tip: buy this

(9 Volt Wall Adapter)

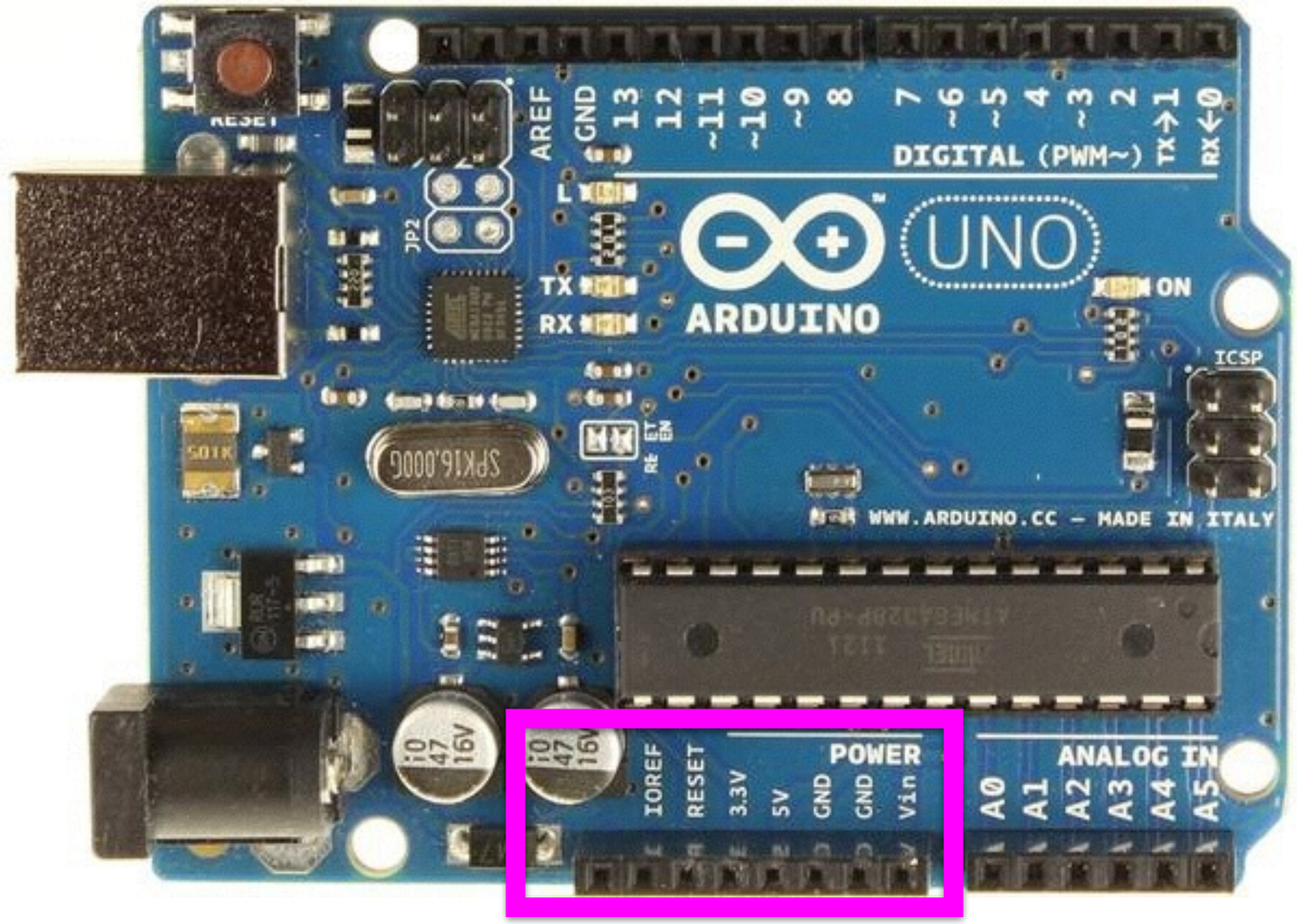
Arduino

voltage regulator

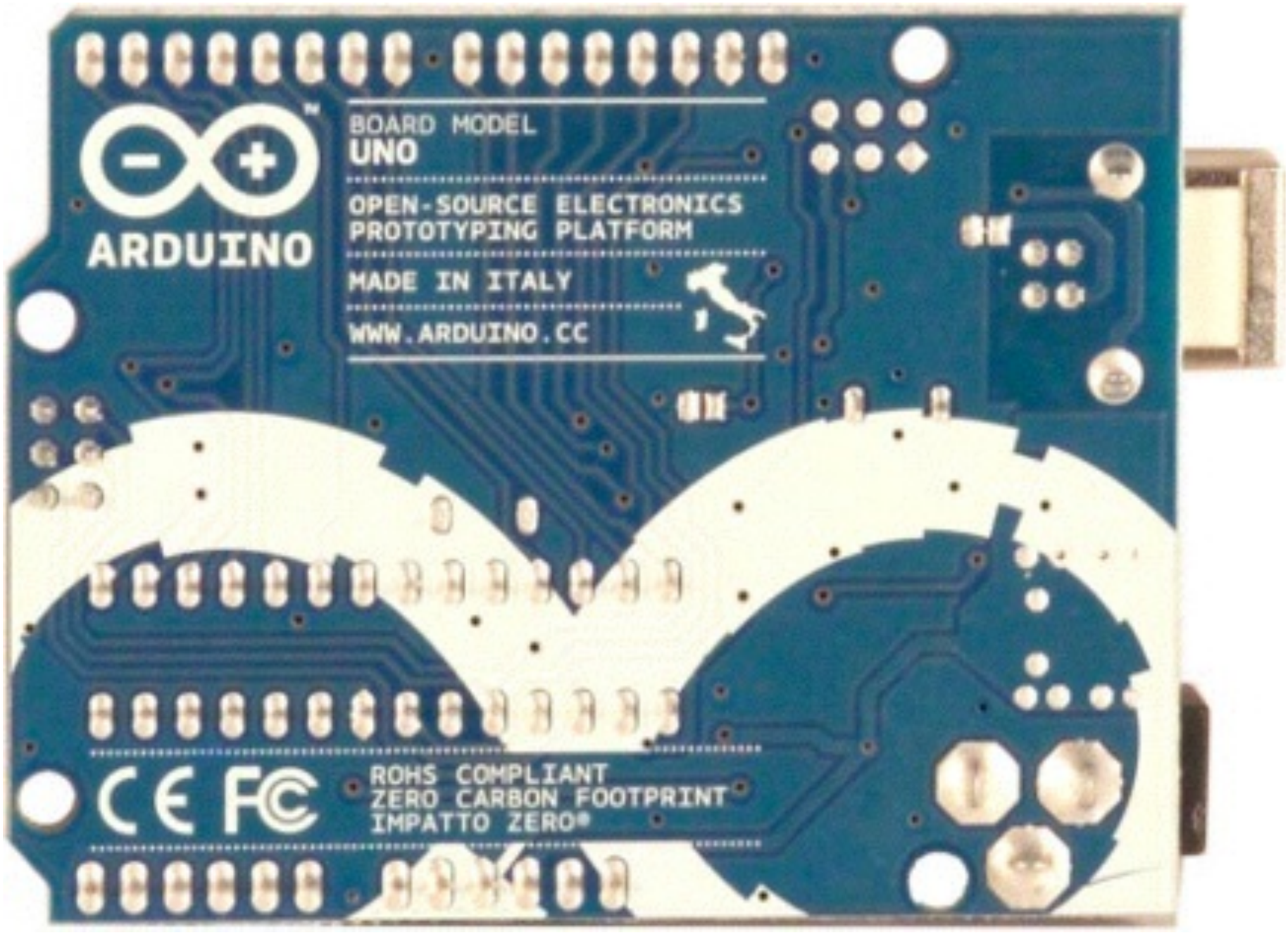


Arduino

Vin, 5V, 3.3V, GND + Reset Pins

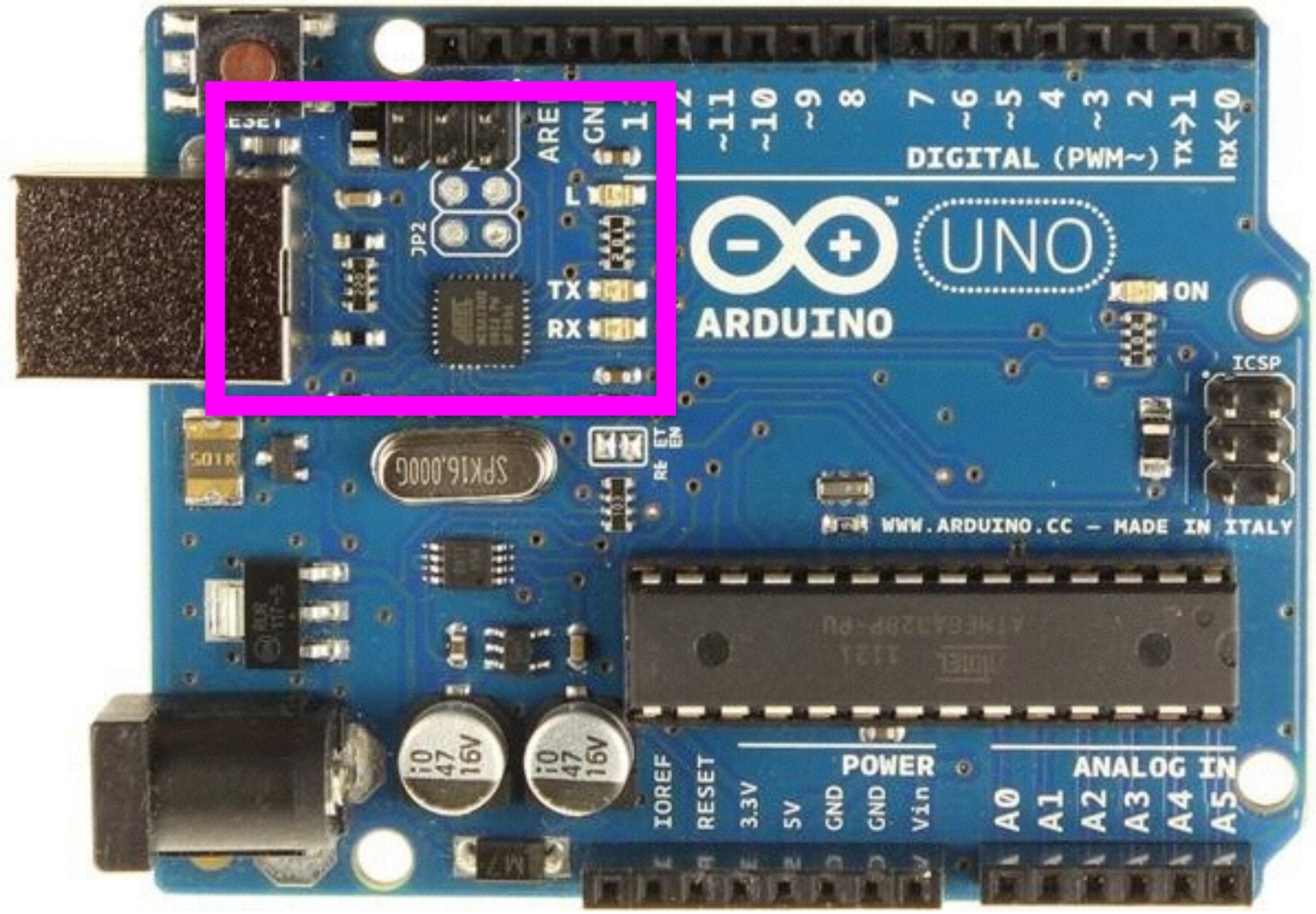


Arduino


ARDUINO
BOARD MODEL
UNO
OPEN-SOURCE ELECTRONICS
PROTOTYPING PLATFORM
MADE IN ITALY
WWW.ARDUINO.CC
ROHS COMPLIANT
ZERO CARBON FOOTPRINT
IMPATTO ZERO®

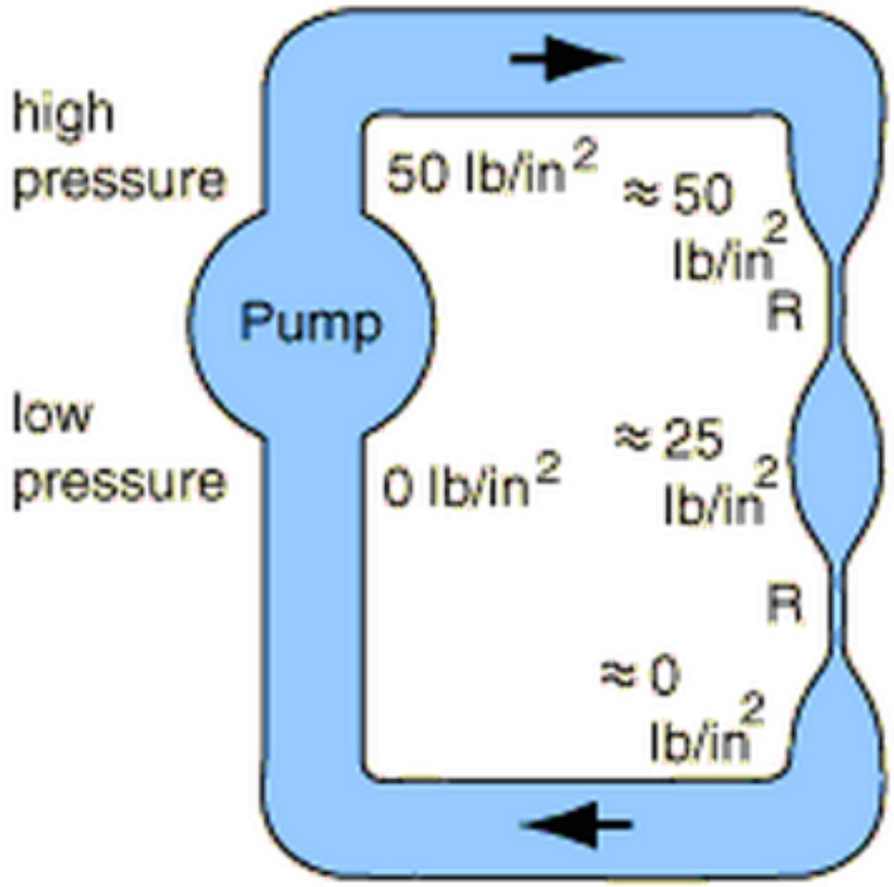
Arduino

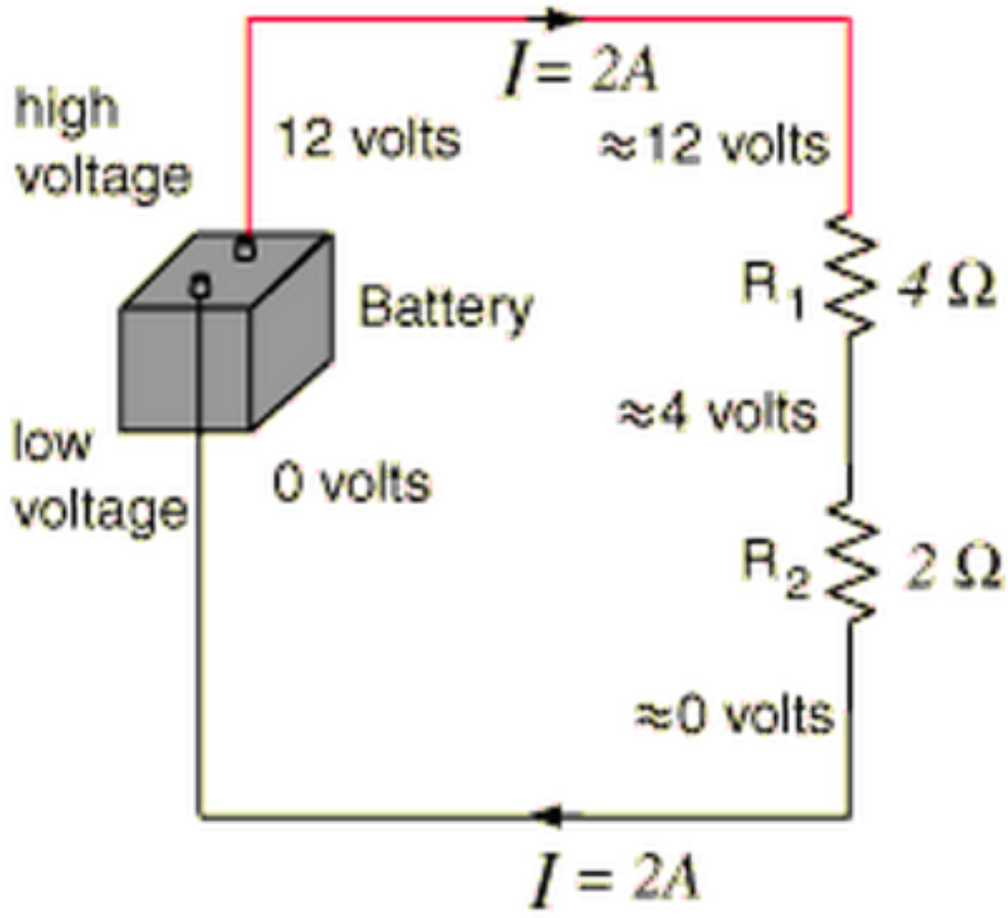
internal LED



ELECTRICITY

Electricity

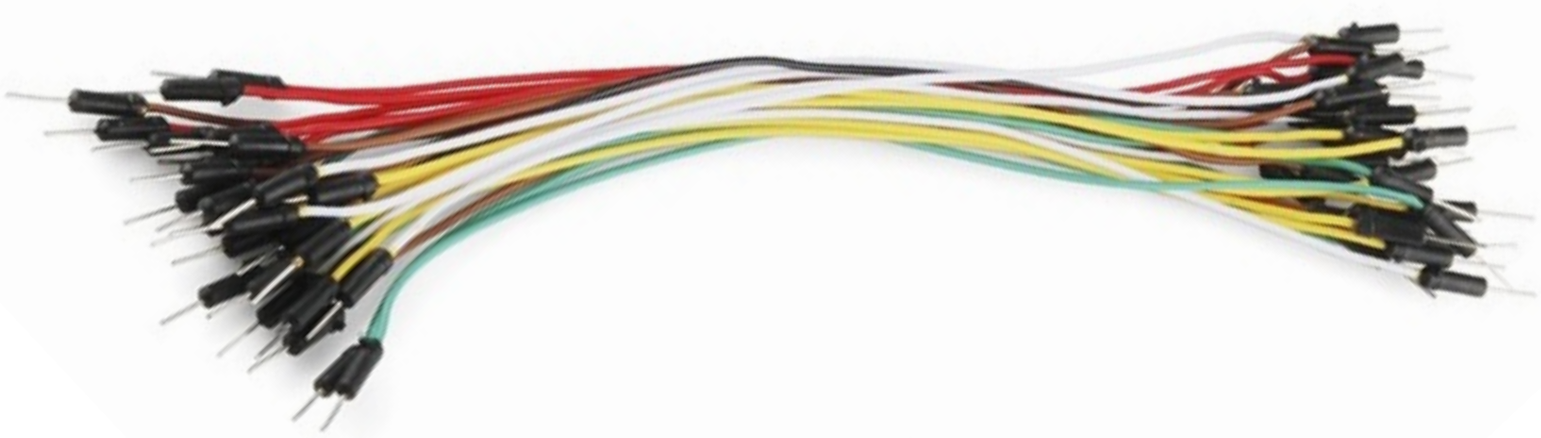




Electricity

wires

(think of them like pipes)



Electricity

battery
(it's the pump)

Electricity

voltage (V)

The force of which electrons are being pushed through the wire.

(the water pressure)

Electricity

current (I)

The amount of electrons moving through the wire at any given moment

(the water itself)

Electricity

current vs. voltage

Electricity

current vs. voltage

Electricity

current vs. voltage

Electricity

resistance (Ω)

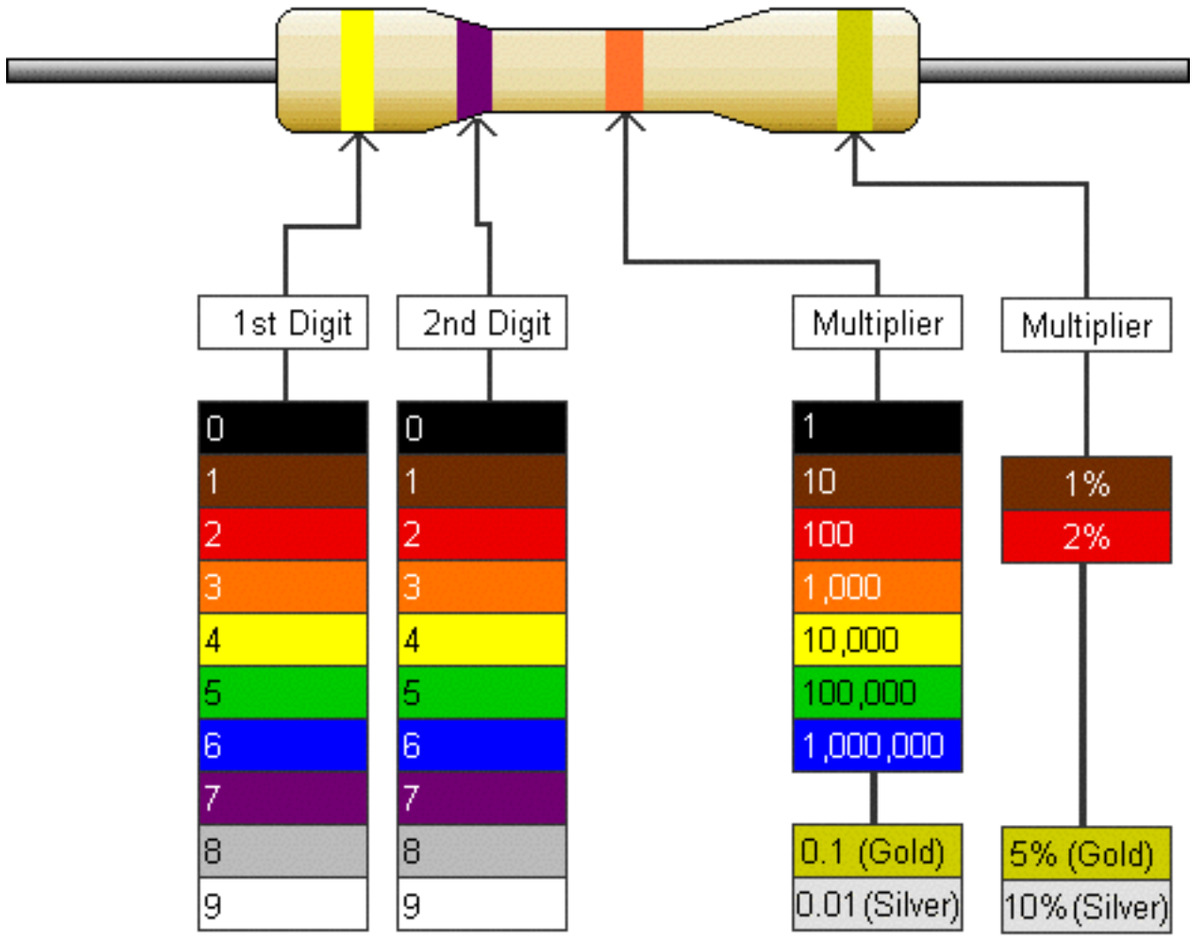
The opposition to the passage of electrons through a wire.

(a pinched pipe)

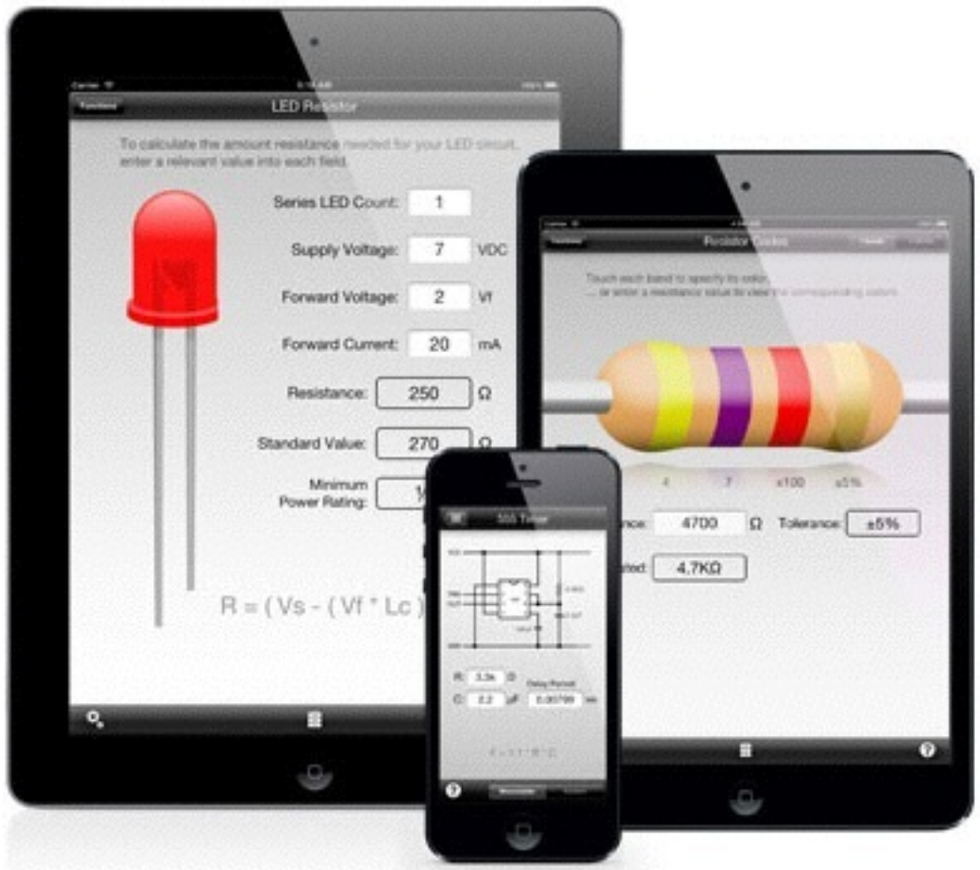
Electricity

Electricity

Resistors are pretty.



Electricity



CIRCUIT
PLAYGROUND

Available on the App Store

Electricity

input
(the handle)

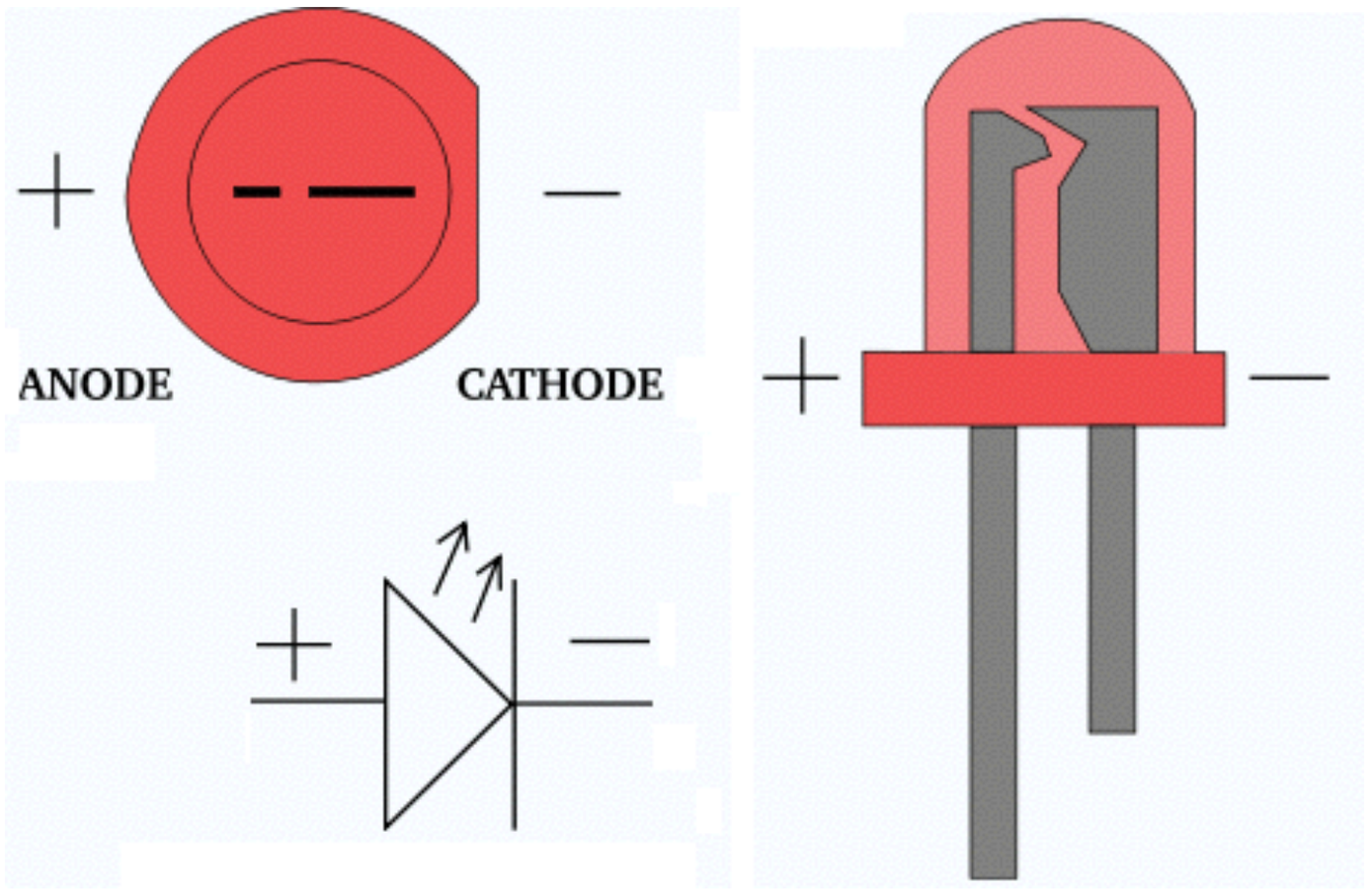
Electricity

output
(the tap)

Electricity

LED

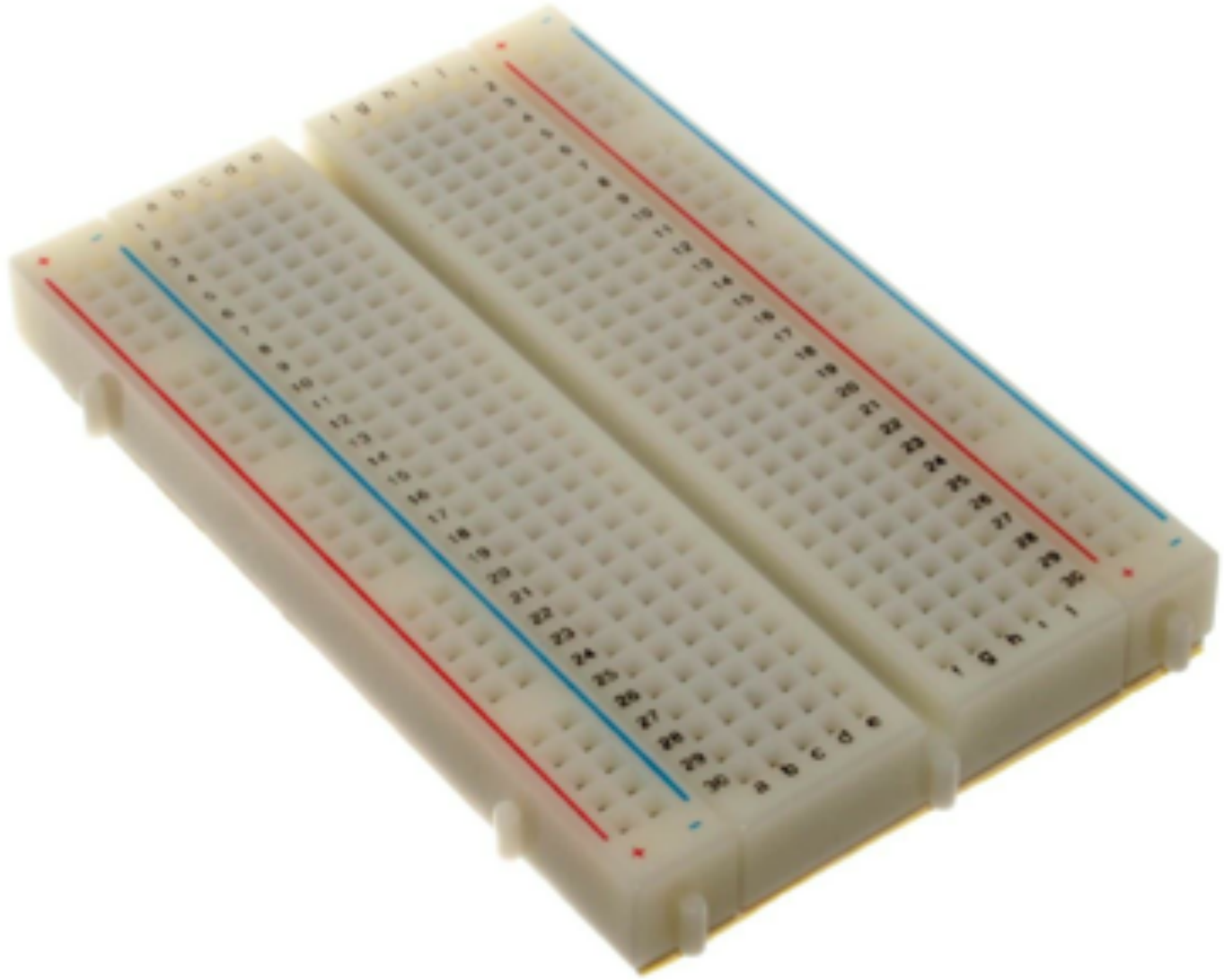
(everyone's favorite output)



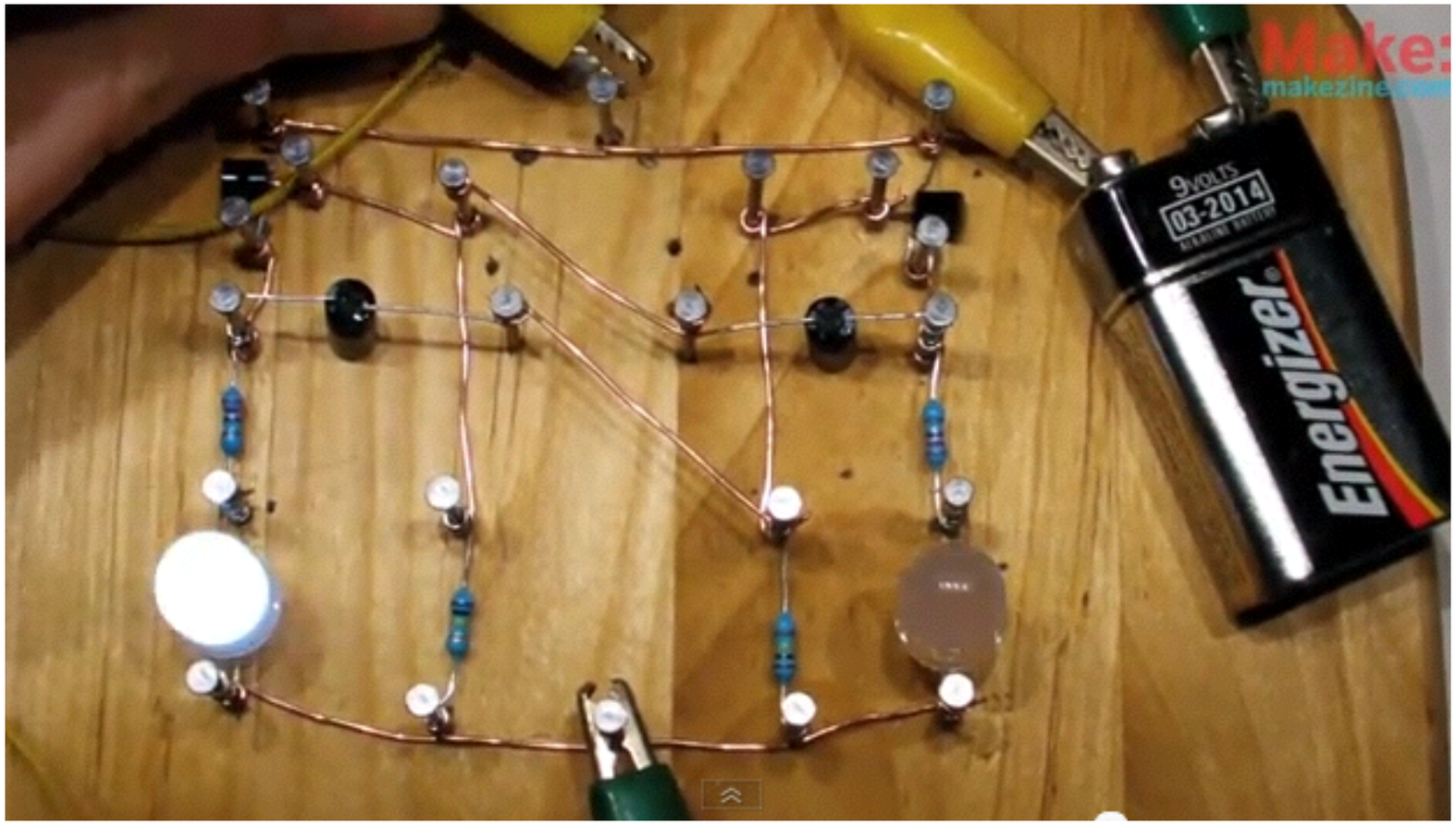
BTW:

Breadboards

Breadboard

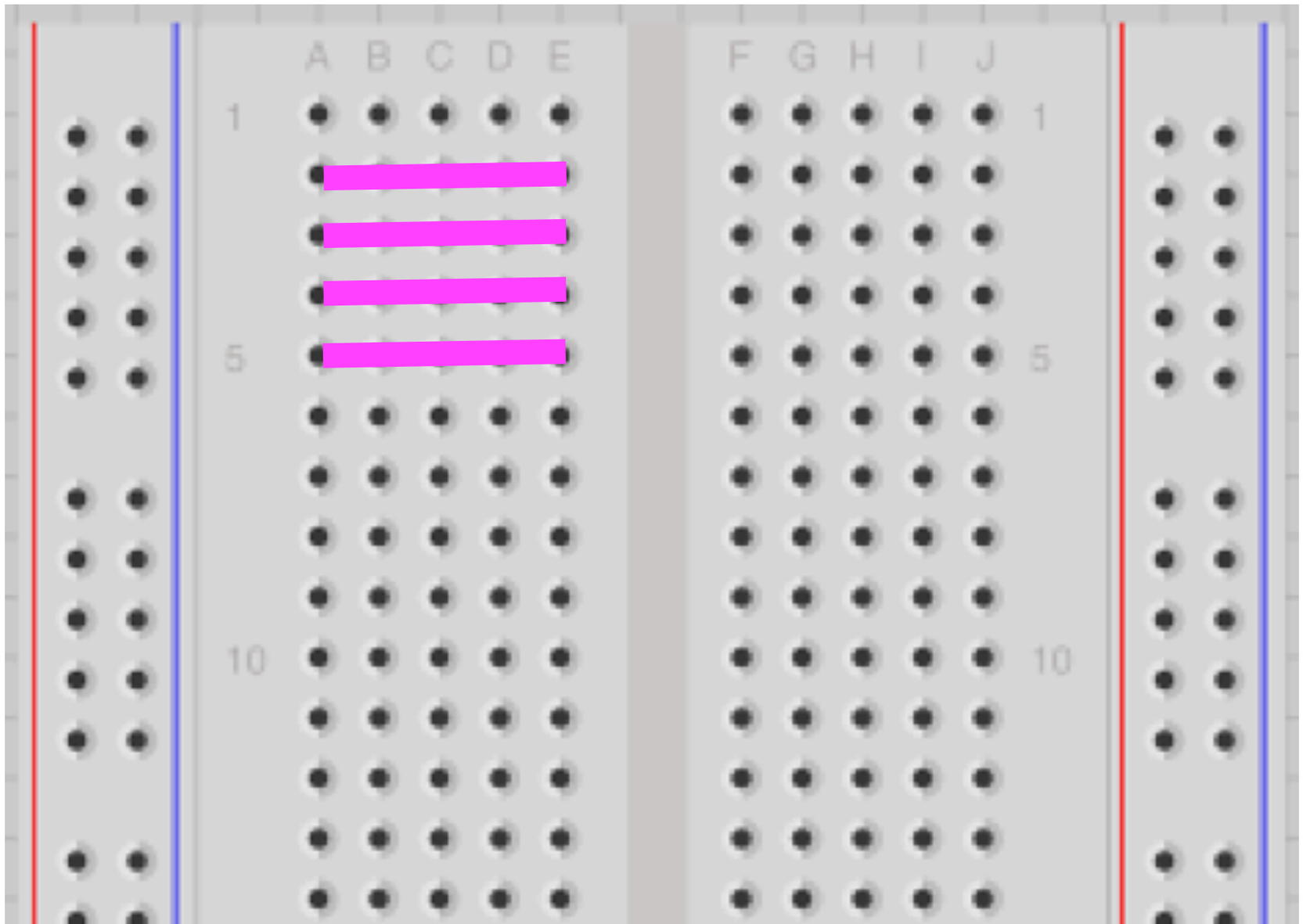


Breadboard



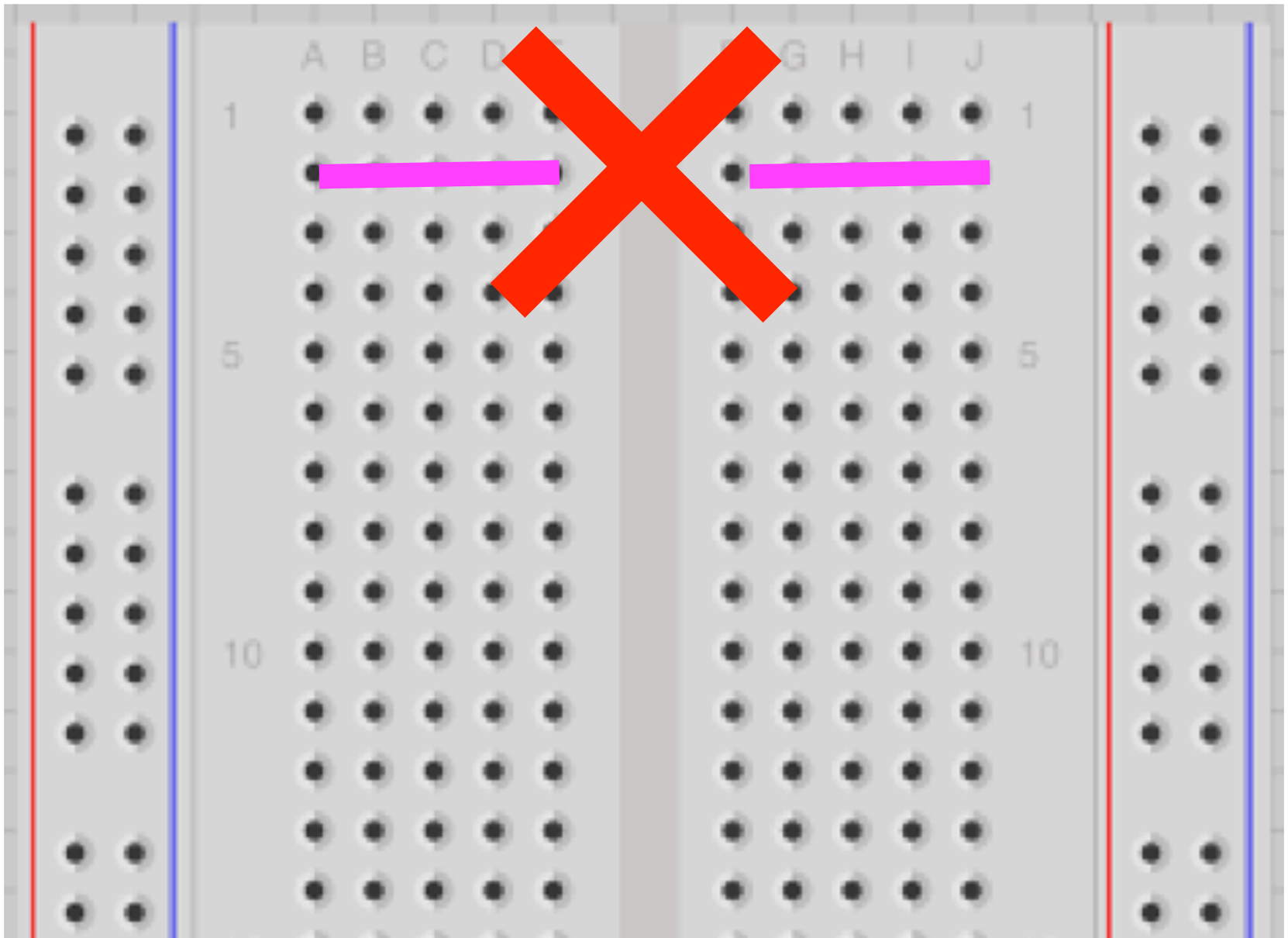
Breadboard

Connections



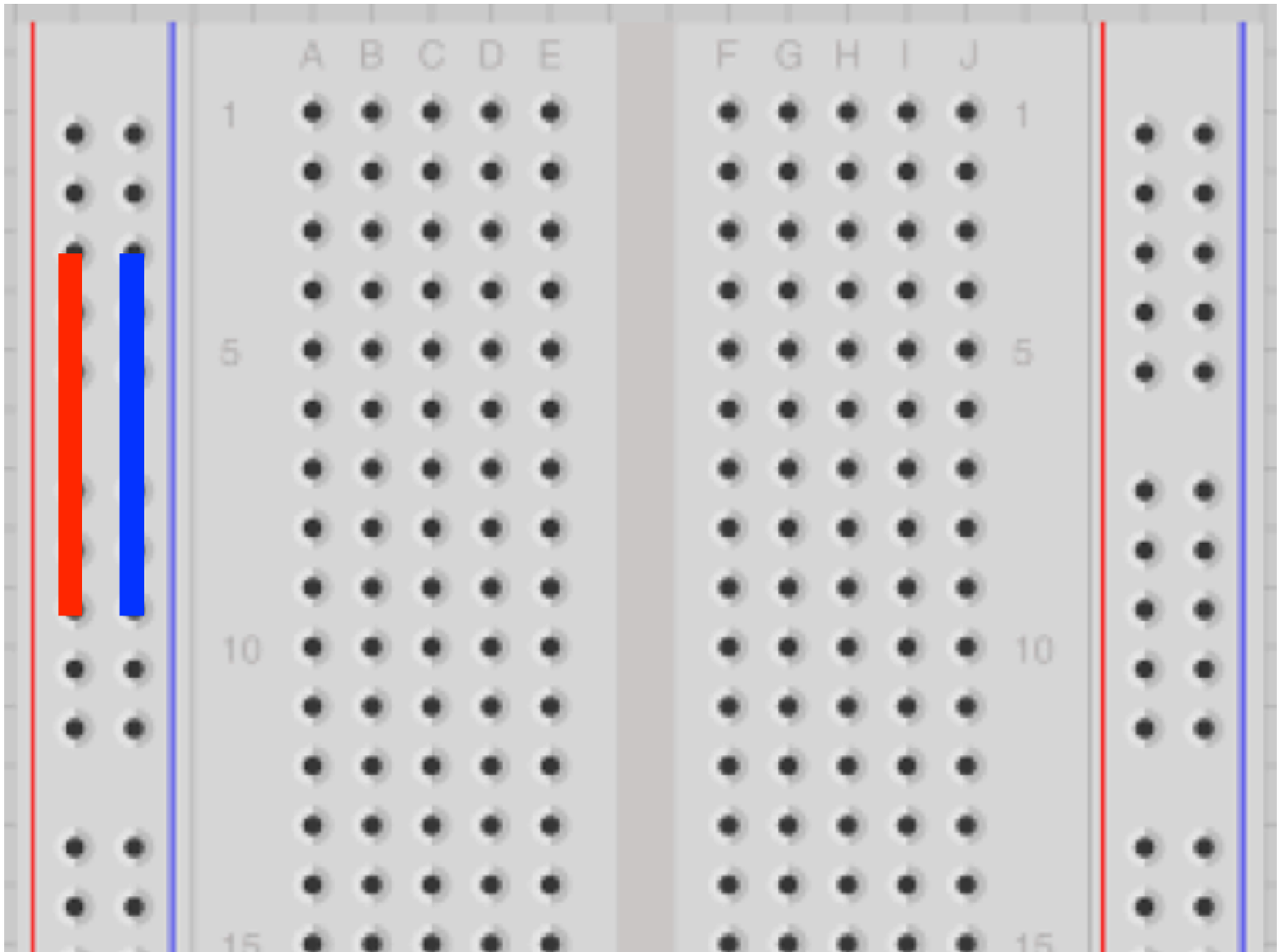
Breadboard

Rows



Breadboard

Ground + Power Rails



LET'S CODE, ALREADY.

Arduino IDE


sketch_sep17a | Arduino 1.0.5
sketch_sep17a
1
Arduino Uno on /dev/tty.usbmodem1411

WAIT — WTF is an IDE?

(Integrated Development Environment)

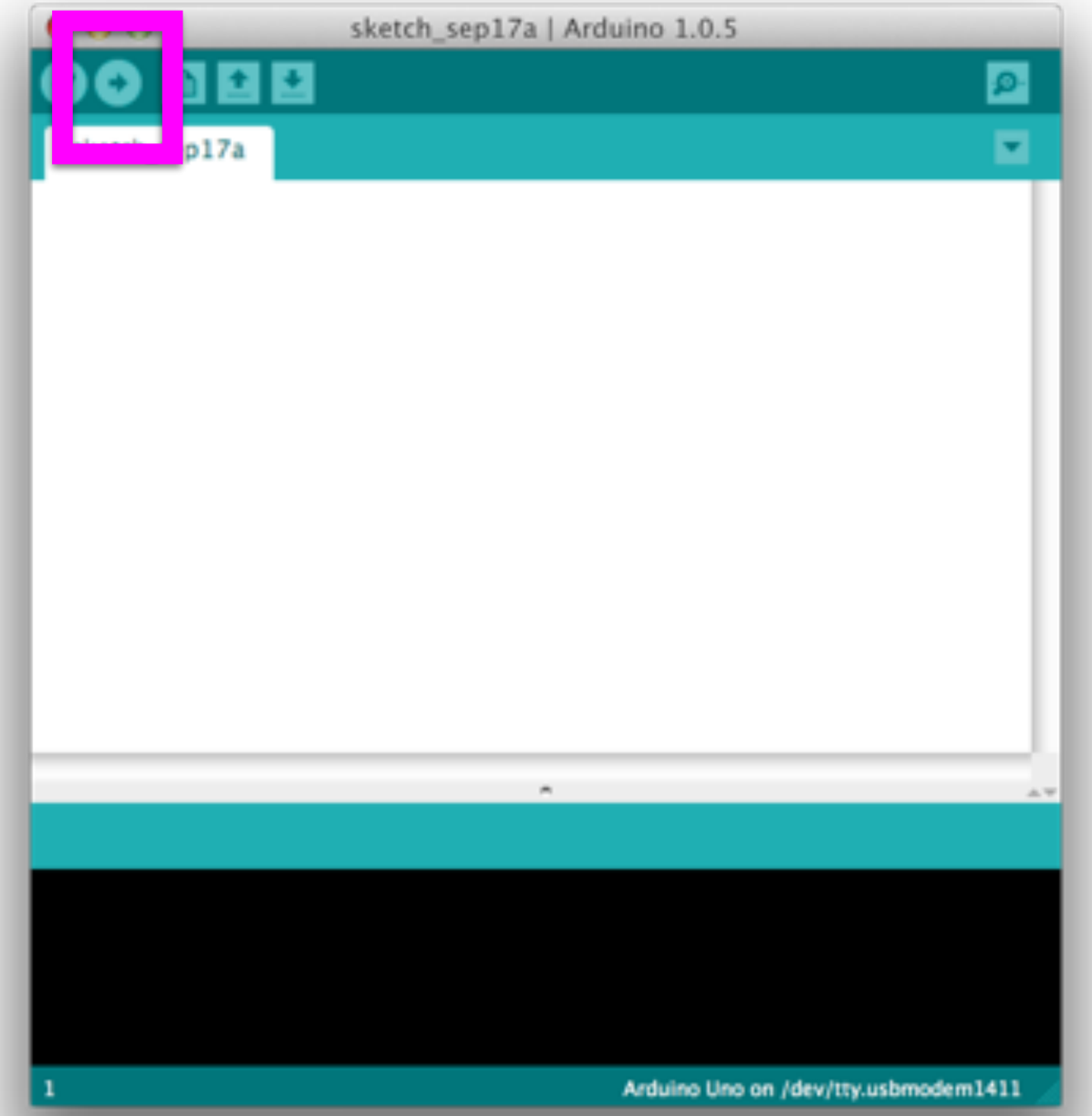
Arduino IDE


sketch_sep17a | Arduino 1.0.5
sketch_sep17a
1
Arduino Uno on /dev/tty.usbmodem1411

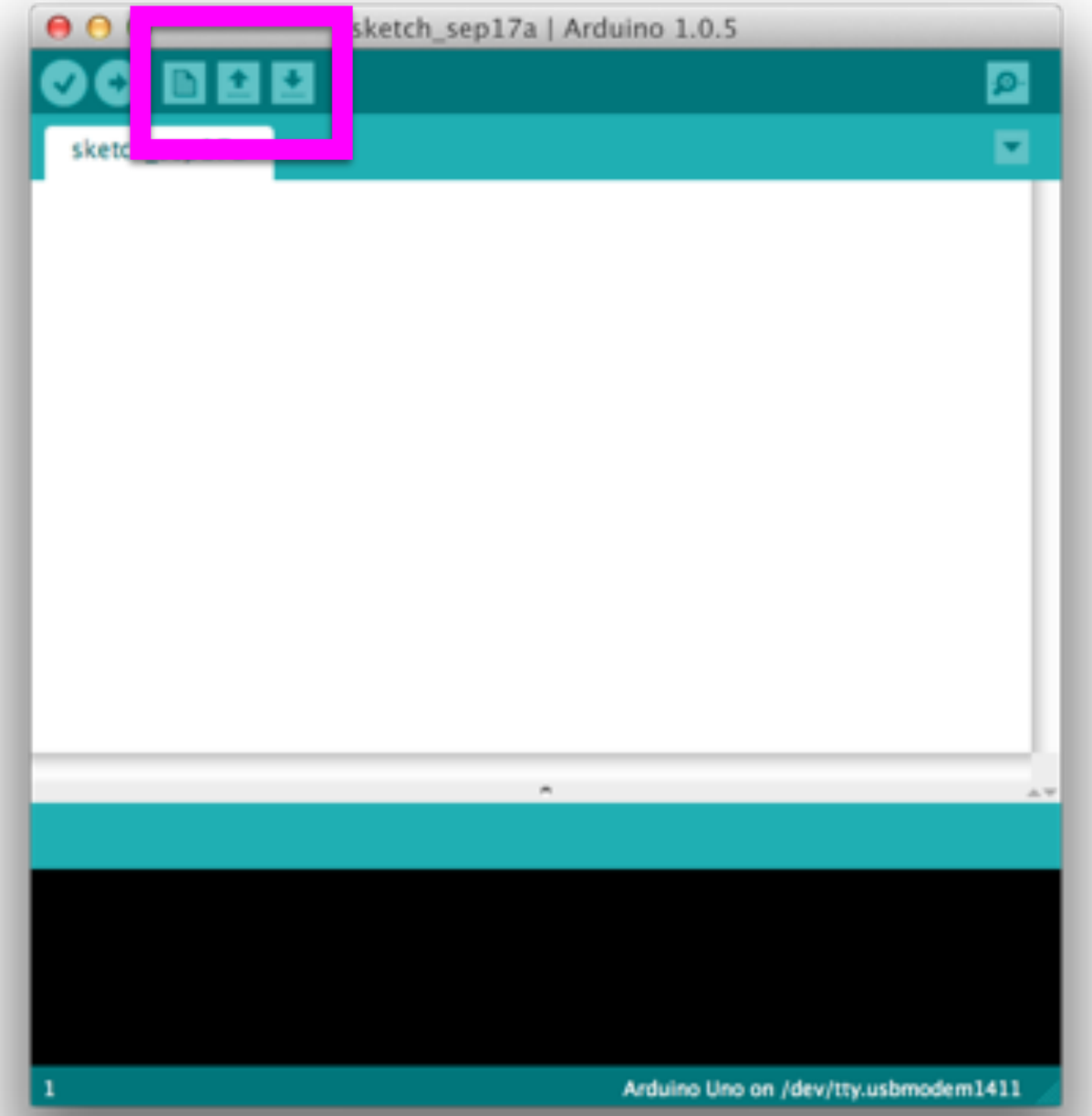
Arduino IDE


sketch_sep17a | Arduino 1.0.5
1
Arduino Uno on /dev/tty.usbmodem1411

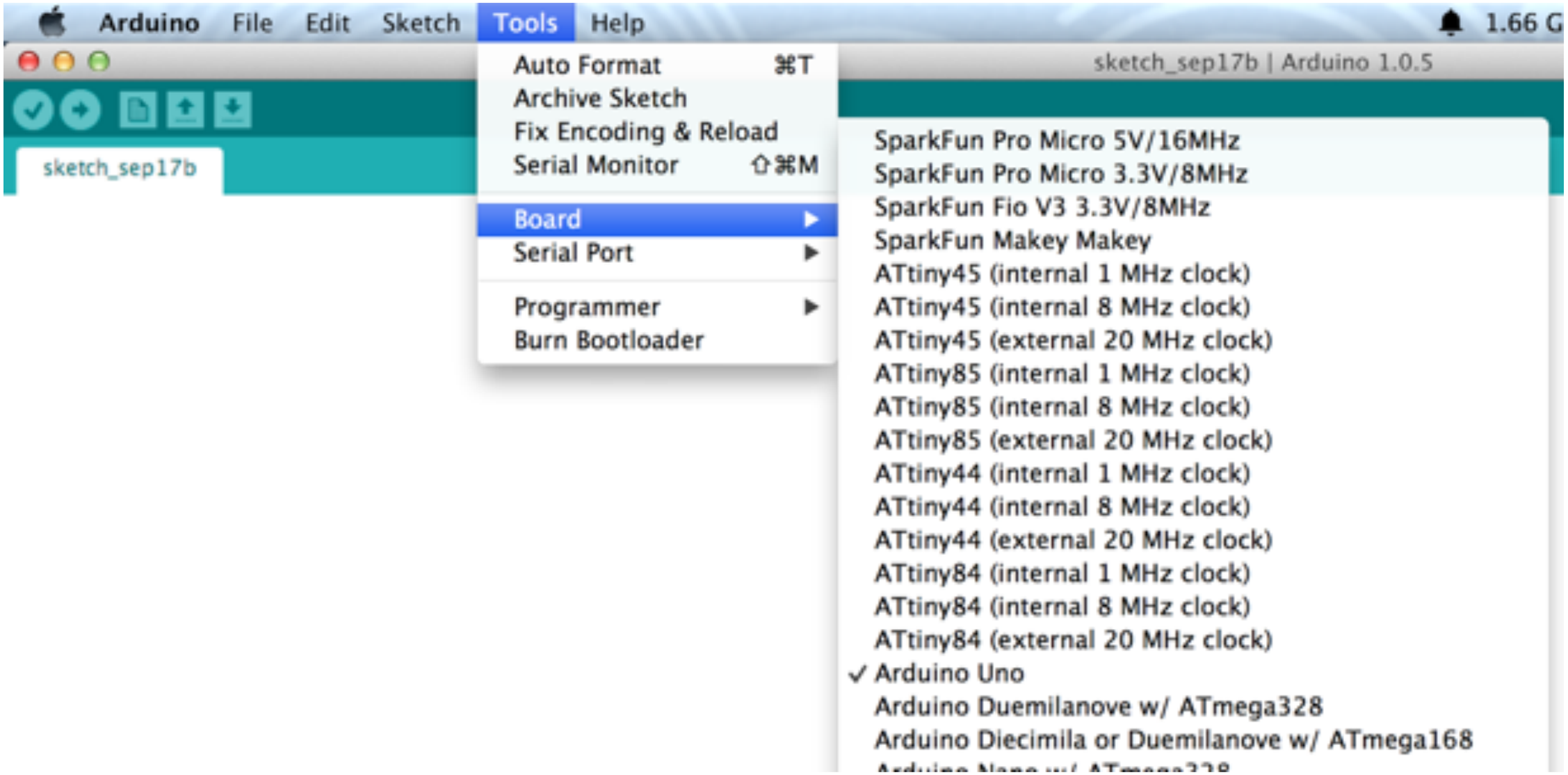
Arduino IDE


sketch_sep17a | Arduino 1.0.5
p17a
1
Arduino Uno on /dev/tty.usbmodem1411

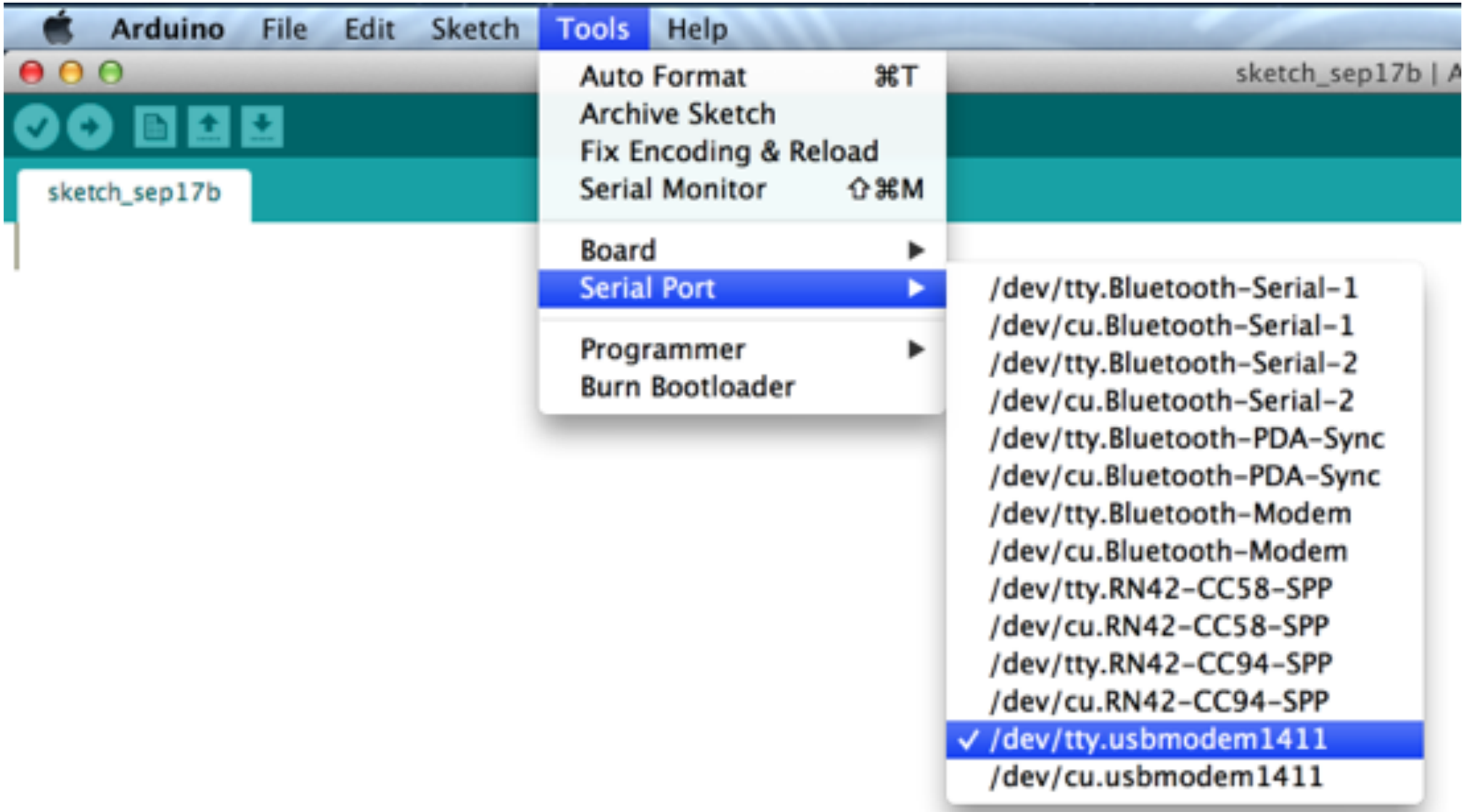
Arduino IDE


sketch_sep17a | Arduino 1.0.5
1
Arduino Uno on /dev/tty.usbmodem1411

Arduino IDE


Arduino
File
Edit
Sketch
Tools
Help
1.66 G
sketch_sep17b | Arduino 1.0.5
sketch_sep17b
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
SparkFun Pro Micro 5V/16MHz
SparkFun Pro Micro 3.3V/8MHz
SparkFun Fio V3 3.3V/8MHz
SparkFun Makey Makey
ATtiny45 (internal 1 MHz clock)
ATtiny45 (internal 8 MHz clock)
ATtiny45 (external 20 MHz clock)
ATtiny85 (internal 1 MHz clock)
ATtiny85 (internal 8 MHz clock)
ATtiny85 (external 20 MHz clock)
ATtiny44 (internal 1 MHz clock)
ATtiny44 (internal 8 MHz clock)
ATtiny44 (external 20 MHz clock)
ATtiny84 (internal 1 MHz clock)
ATtiny84 (internal 8 MHz clock)
ATtiny84 (external 20 MHz clock)
✓ Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168

Arduino IDE


Arduino
File
Edit
Sketch
Tools
Help
sketch_sep17b | A
sketch_sep17b
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
/dev/tty.Bluetooth-Serial-1
/dev/cu.Bluetooth-Serial-1
/dev/tty.Bluetooth-Serial-2
/dev/cu.Bluetooth-Serial-2
/dev/tty.Bluetooth-PDA-Sync
/dev/cu.Bluetooth-PDA-Sync
/dev/tty.Bluetooth-Modem
/dev/cu.Bluetooth-Modem
/dev/tty.RN42-CC58-SPP
/dev/cu.RN42-CC58-SPP
/dev/tty.RN42-CC94-SPP
/dev/cu.RN42-CC94-SPP
✓ /dev/tty.usbmodem1411
/dev/cu.usbmodem1411

Arduino IDE

Open up the Blink sketch.

(FILE > EXAMPLES > BASICS > Blink Sketch)

Arduino IDE

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Arduino IDE

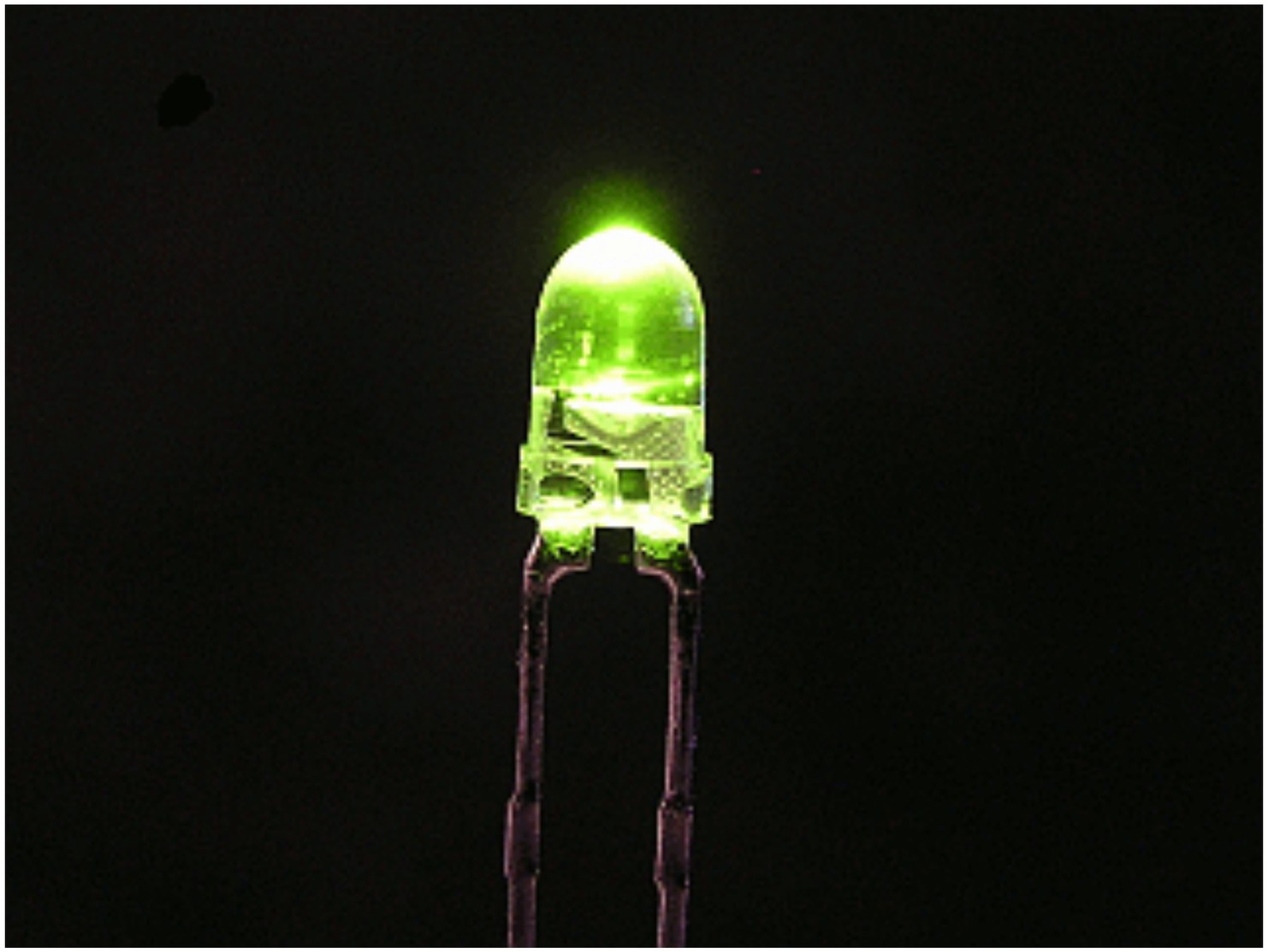
```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Arduino IDE

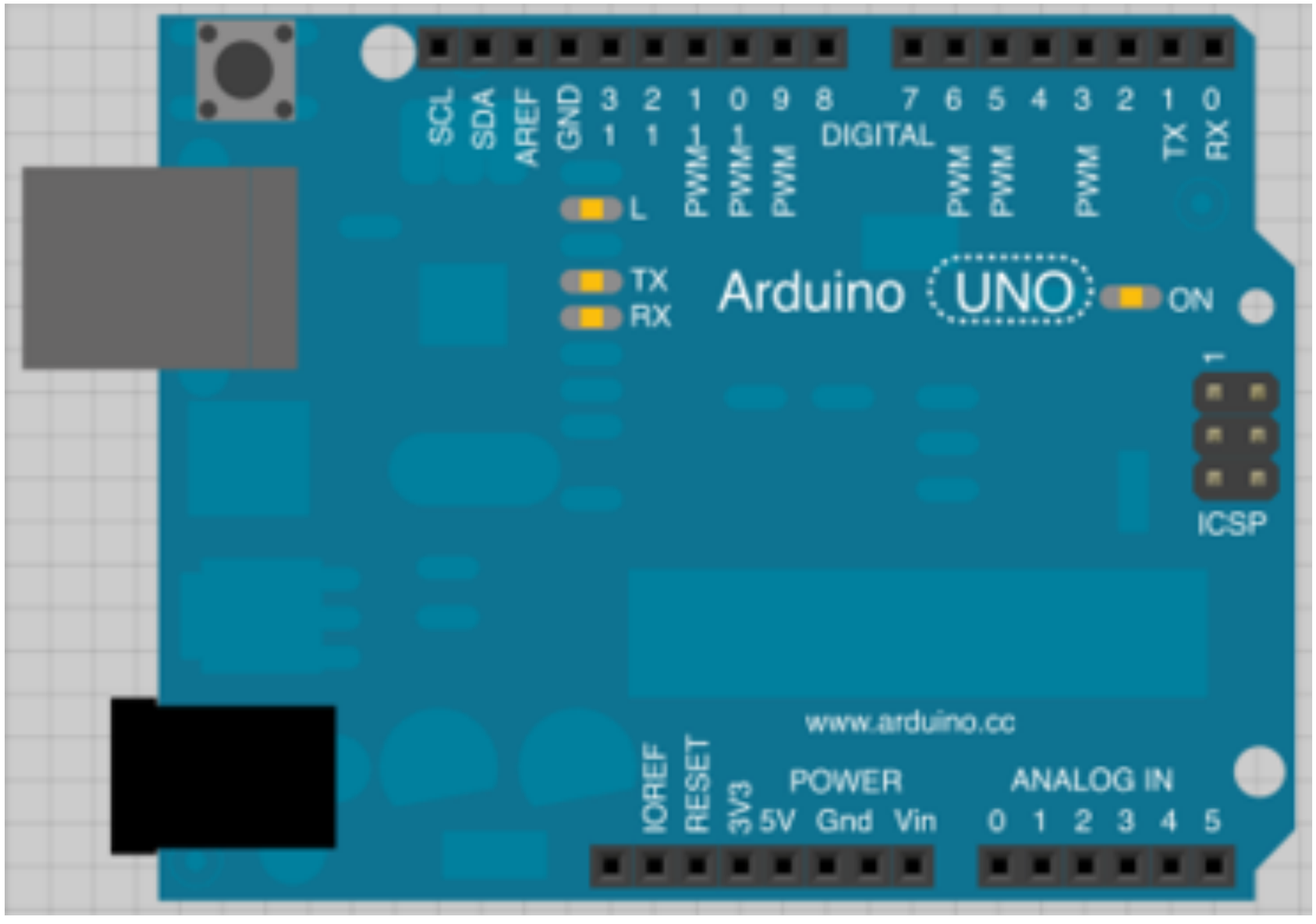
```
void setup() {
  //start the serial connection from Arduino back to computer
  Serial.begin(9600);

  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

```
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  Serial.println("LED is On");
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  Serial.println("LED is Off");
  delay(1000);               // wait for a second
}
```

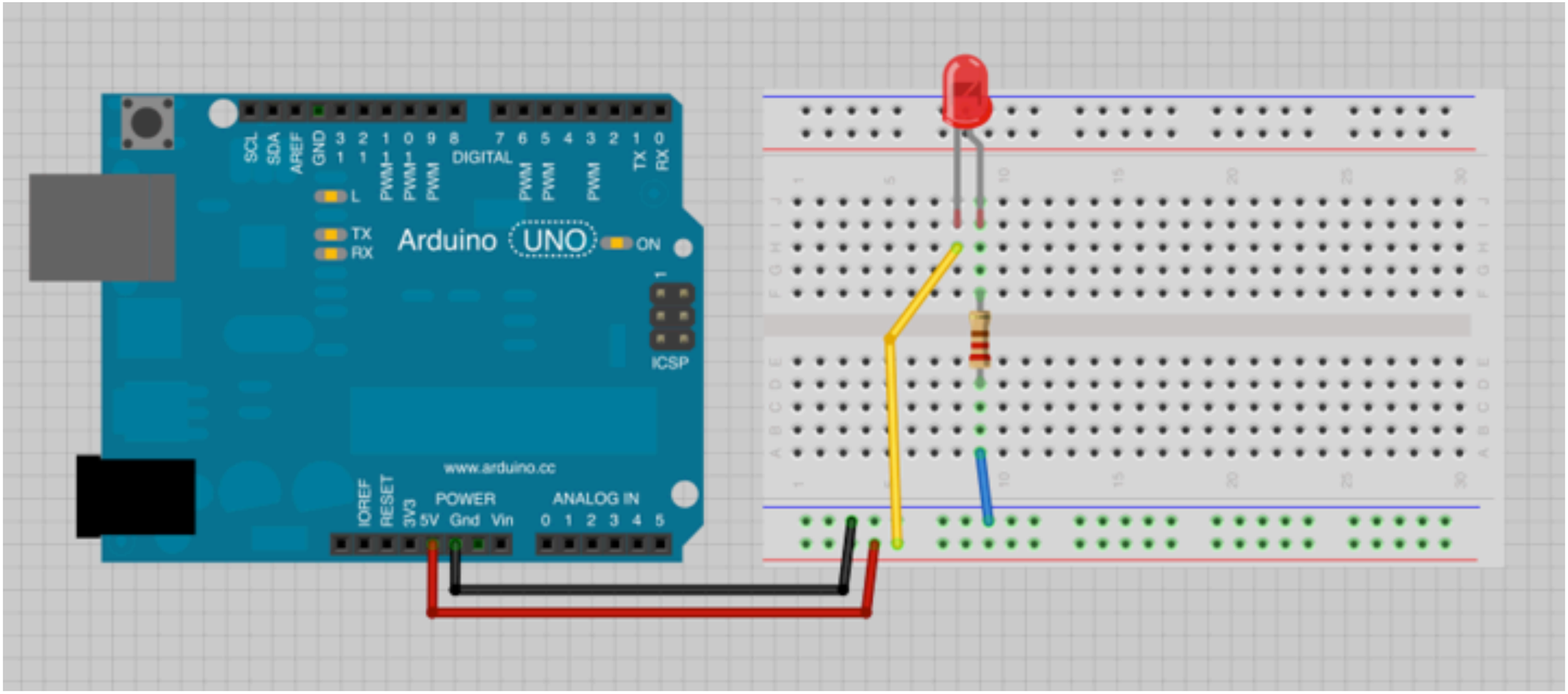
Go Time!



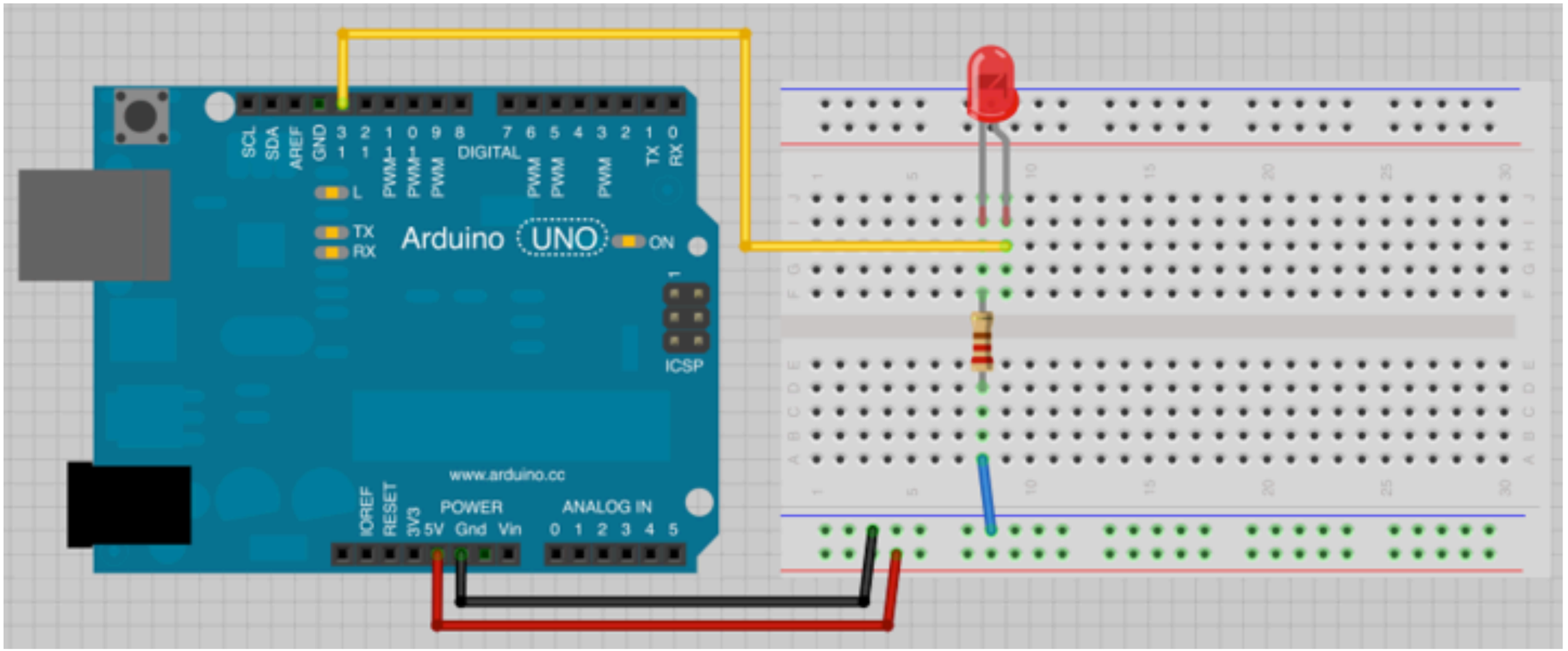
Connecting an LED



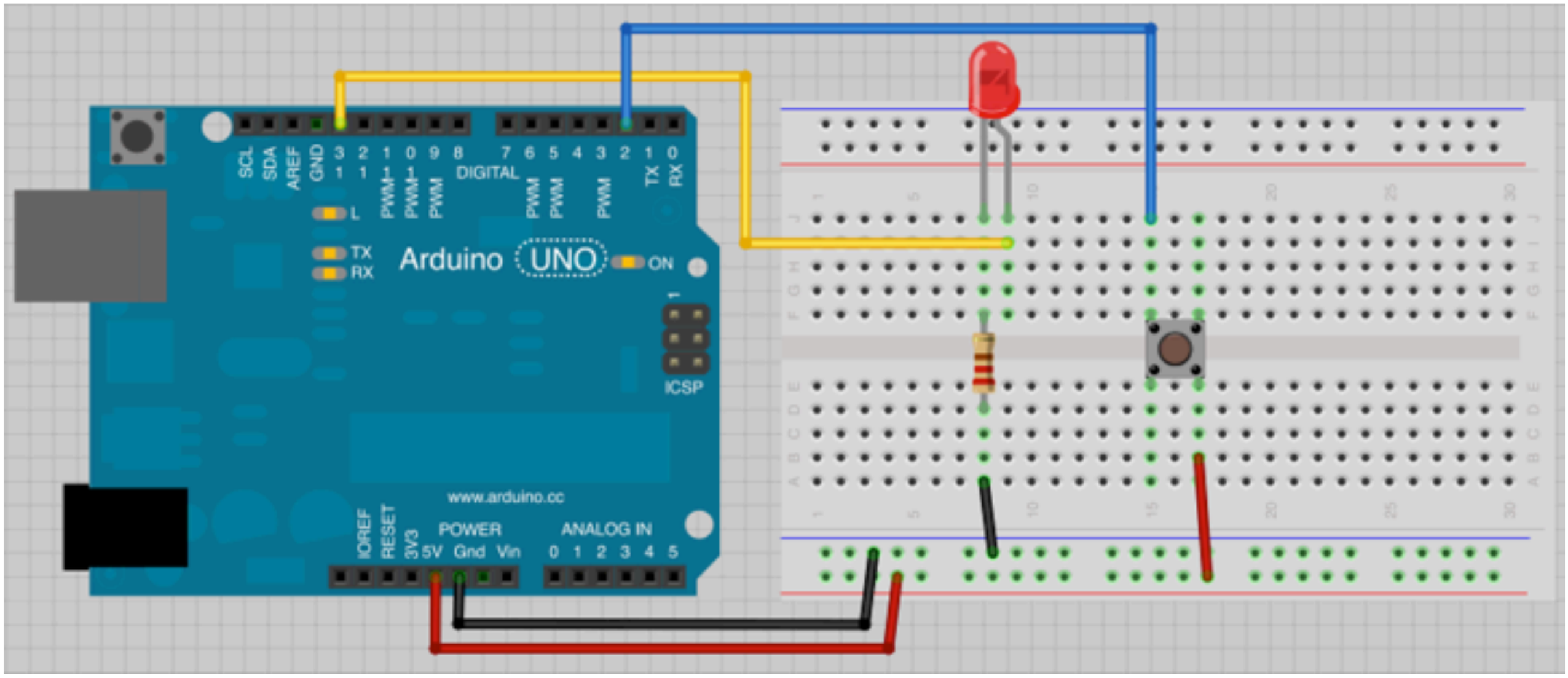
Connecting an LED



Connecting an LED



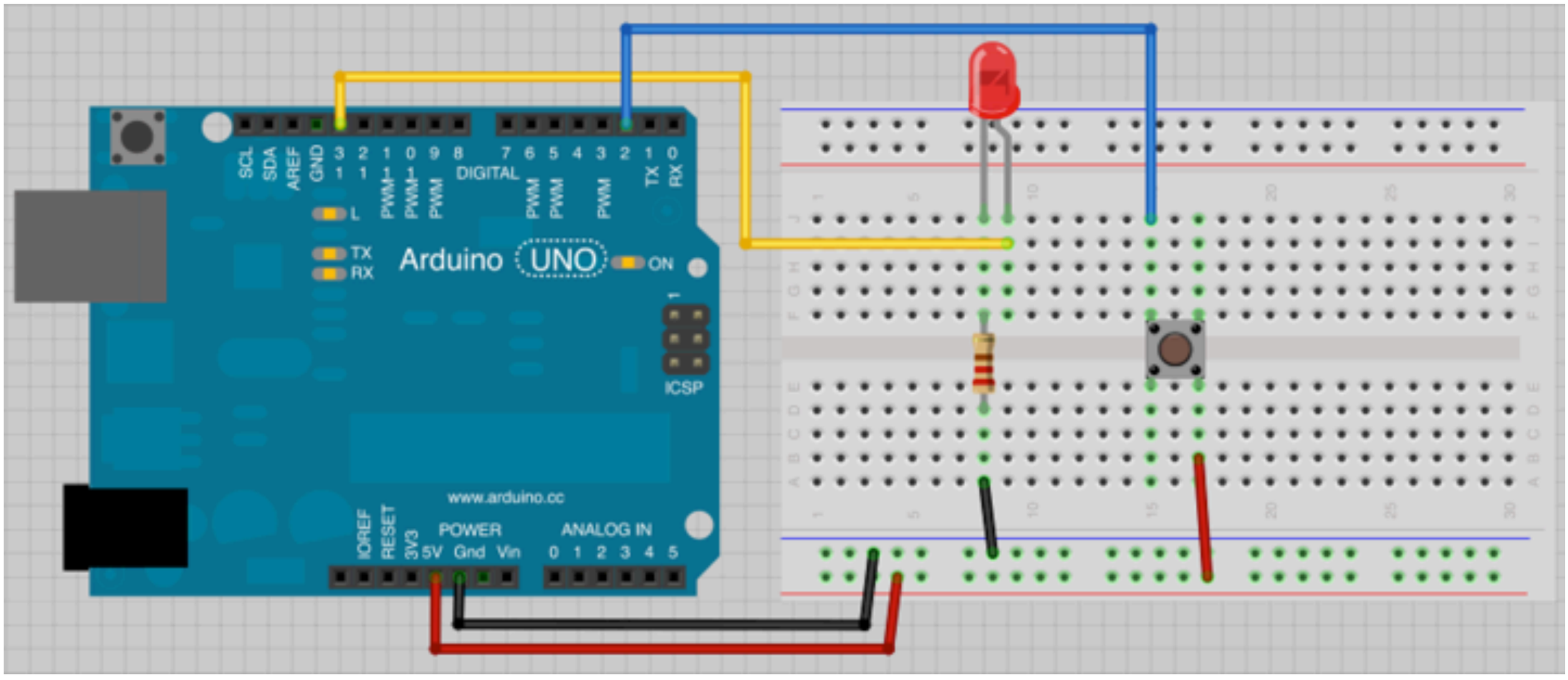
Connecting an LED



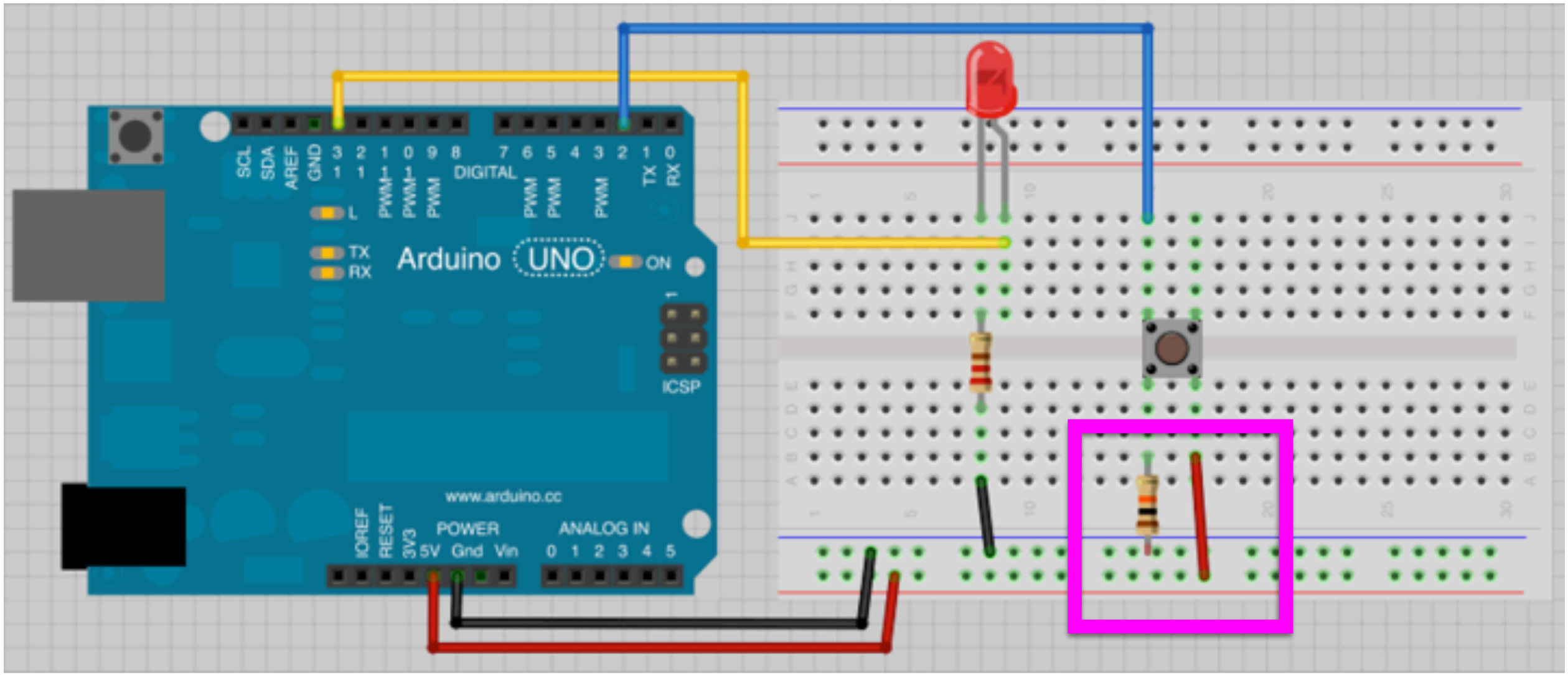
Connecting an LED Read the button with code.

```
int led = 13;
int buttonPin = 2;
int buttonState = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the pinModes
  pinMode(led, OUTPUT);
  pinMode(buttonPin, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  //read the button
  buttonState = digitalRead(buttonPin);
  //Perform different actions depending on the state of the button
  if(buttonState == HIGH){
    digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);               // wait for a second
  } else {
    digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);               // wait for a second
  }
}
```

Connecting an LED Connecting a button.



Connecting an LED



Yup, there's Homework

Homework

Get the class code up and running + take a five second video

Homework

Update your code so the button triggers a state change + take a 5 second video.

(As in — the LED stays on when you push it and turns off when the button is pressed again)

Hint: Look up DEBOUNCING.

Push all your code + videos to GIT before class.

```
git add .
git commit -m “YOUR MESSAGE”
git push origin master
```