

Intro to Physical Computing

aka How to make Daft Punk Helmets 101.

CC Lab (Fall 2016)

What's an “Arduino”?

Microcontroller

Input/Output machine

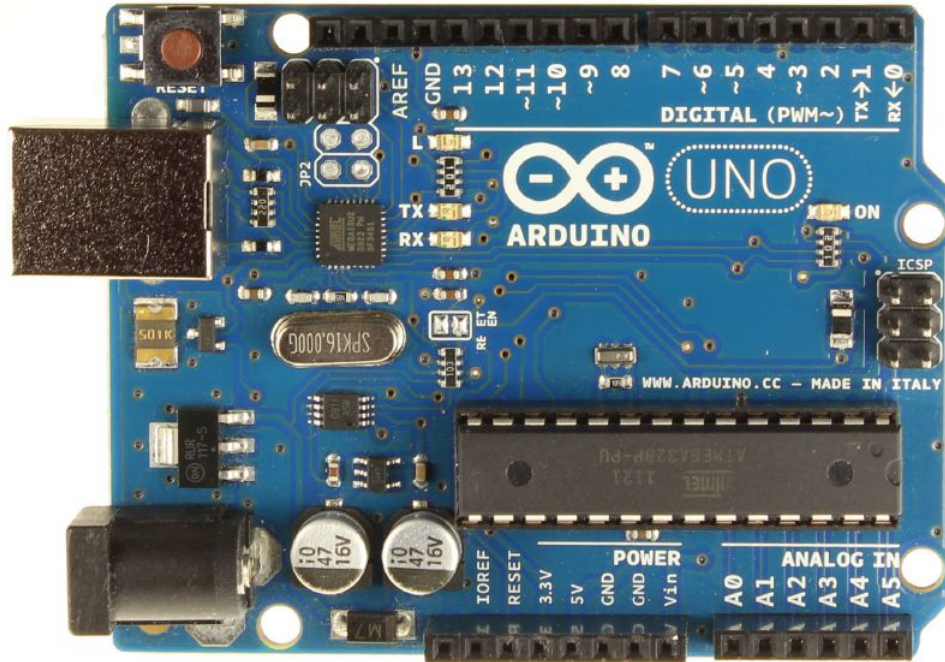
Made for rapid prototyping

(without requiring custom boards design)

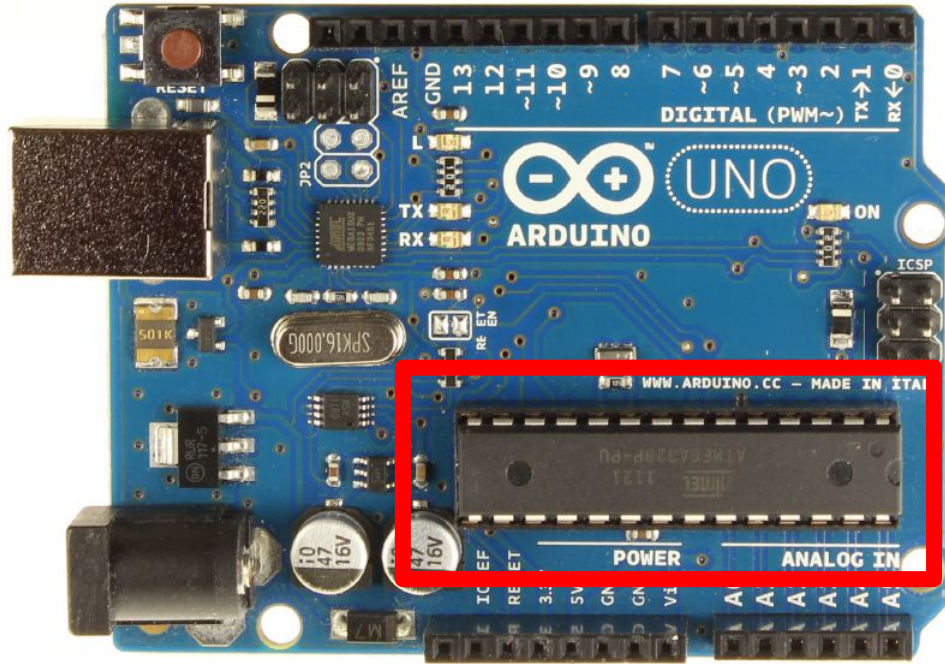
Open source

Large community to support it

What's on the board?

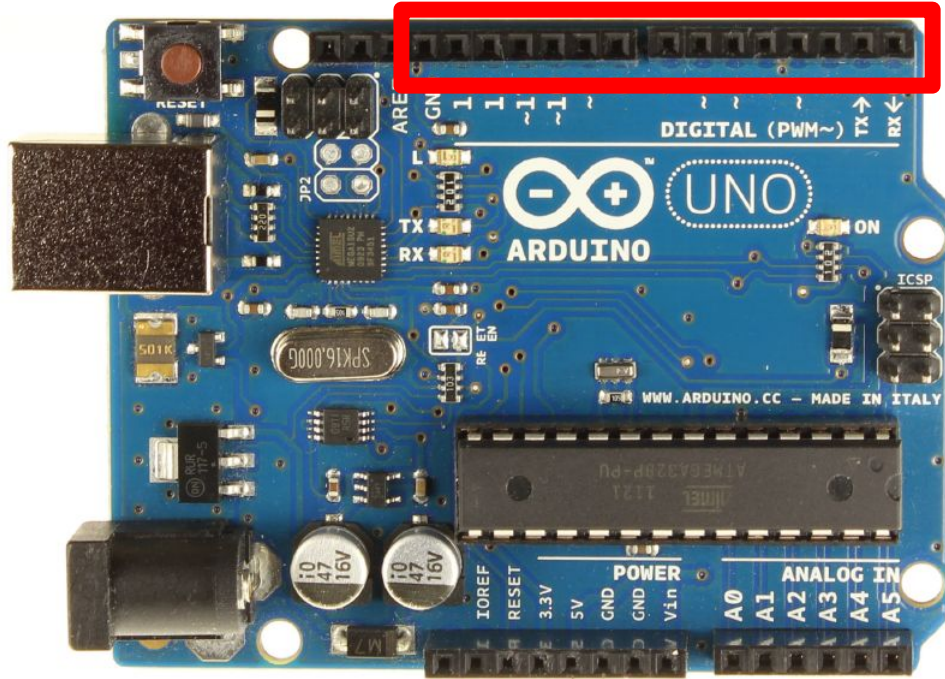


The Brain - ATmega 328p chip



You can also pull it out.
Careful! Don't bend the pins

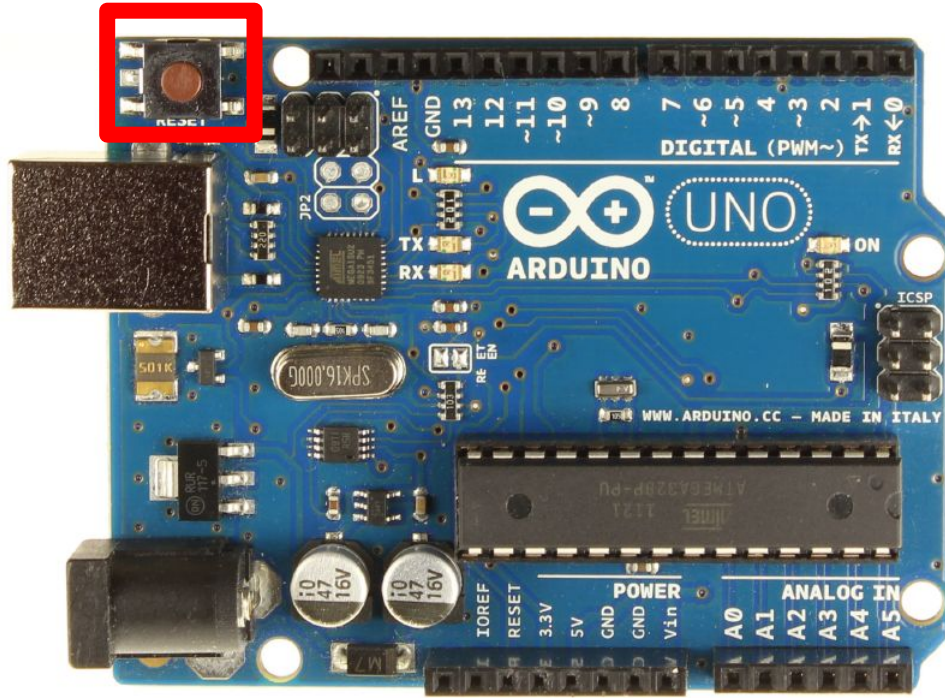
Digital Pins



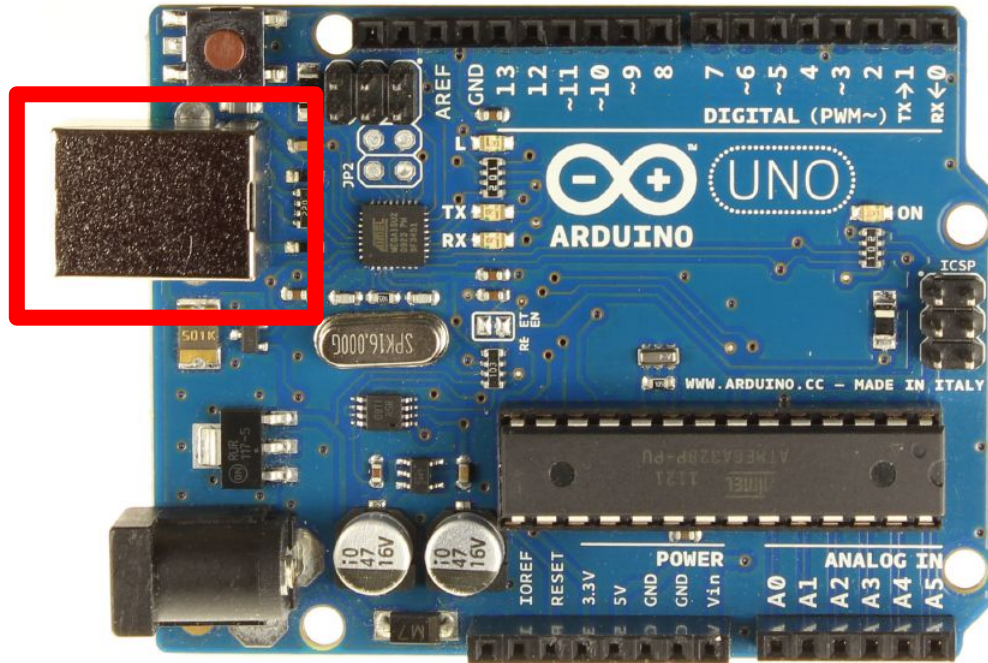
Analog Pins



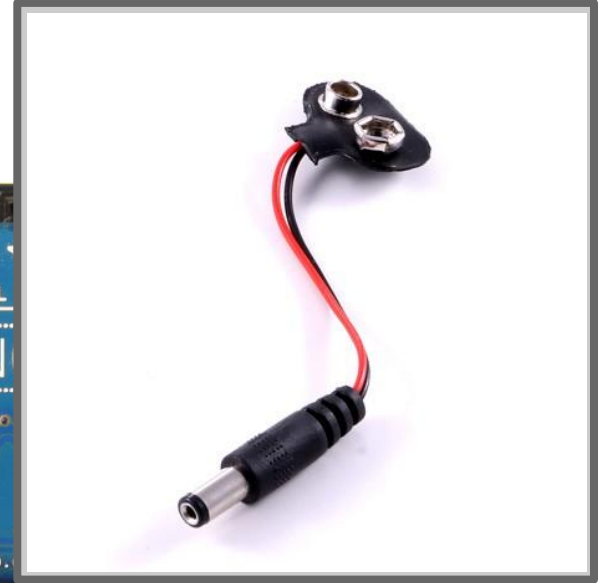
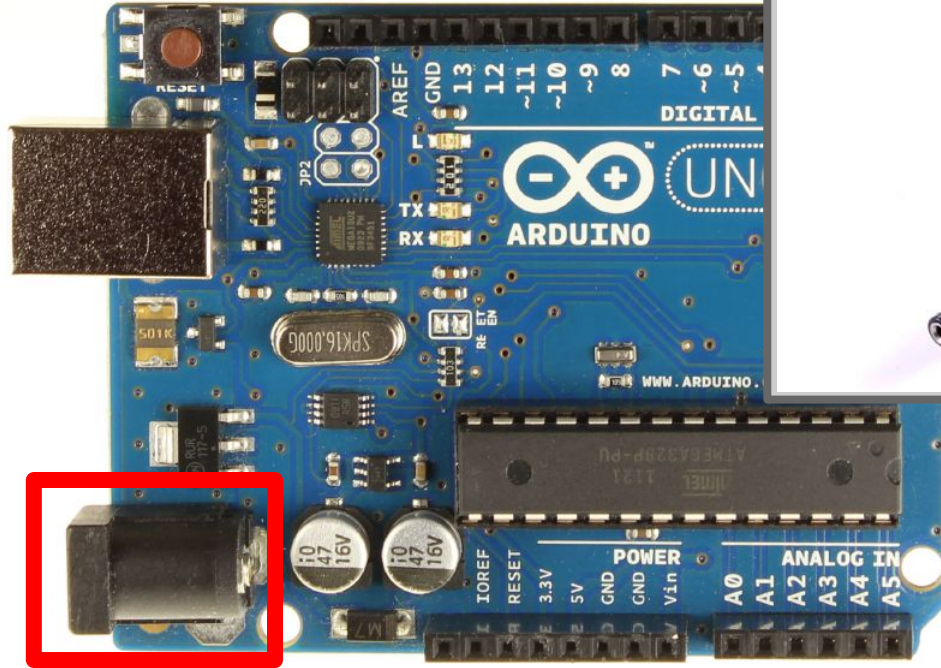
Reset Button



USB Port



Power Jack

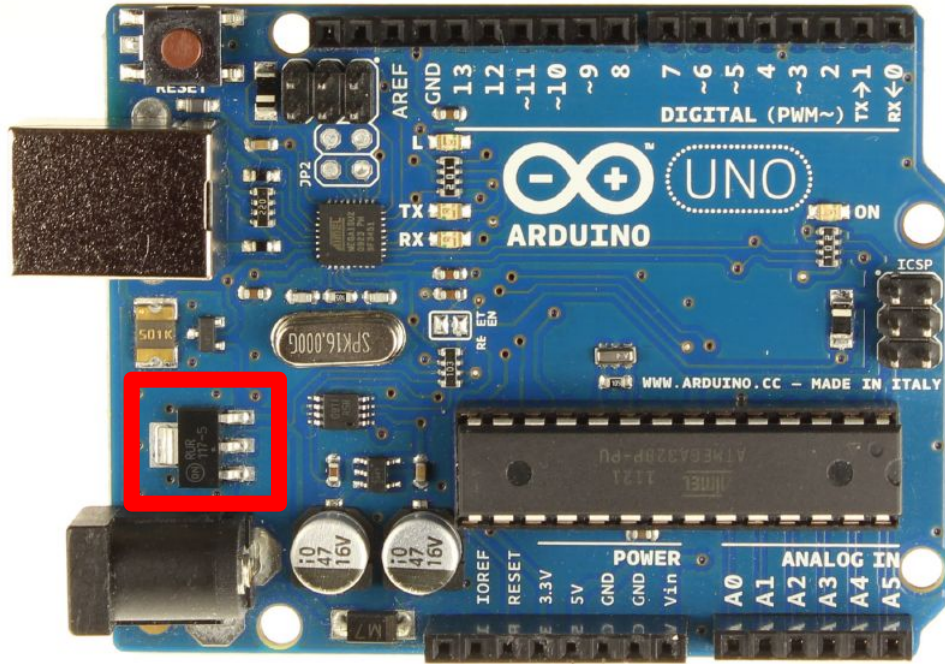


PRO-TIP: Buy one of these

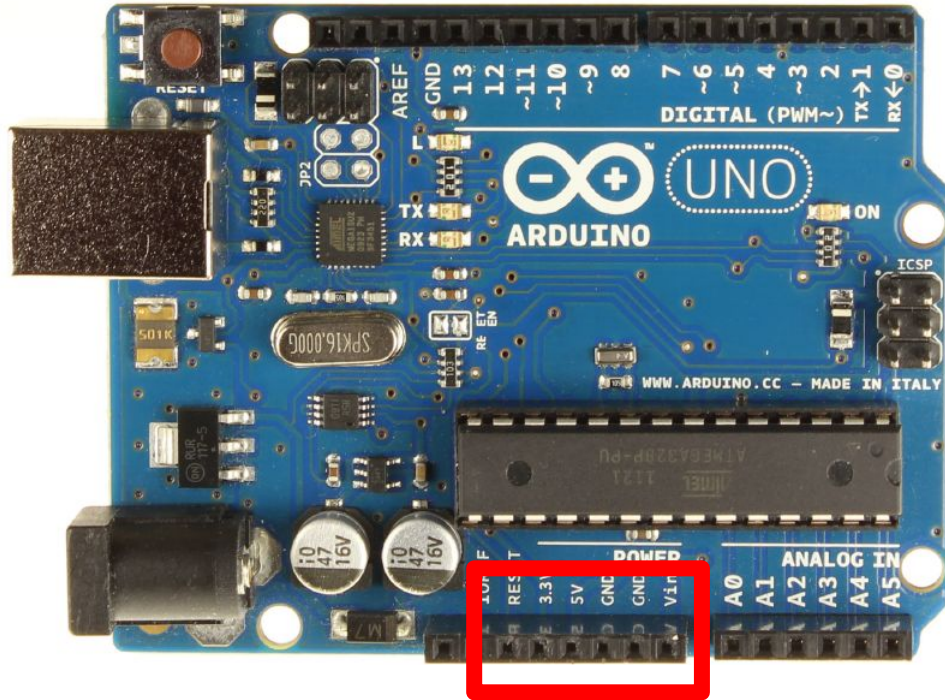


Specs: <http://playground.arduino.cc/Learning/WhatAdapter>

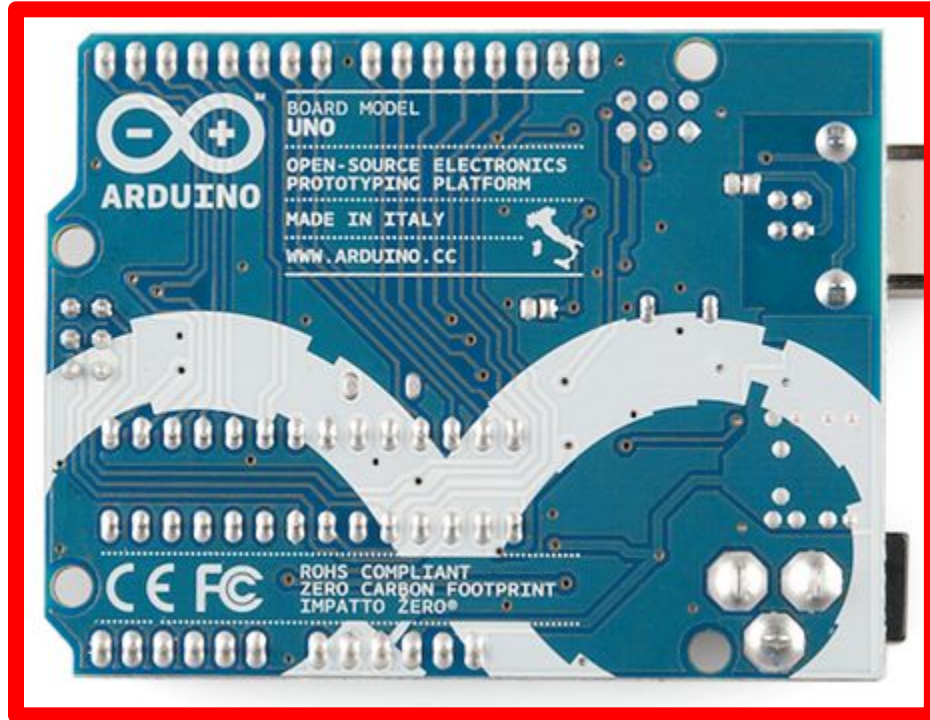
Voltage Regulator



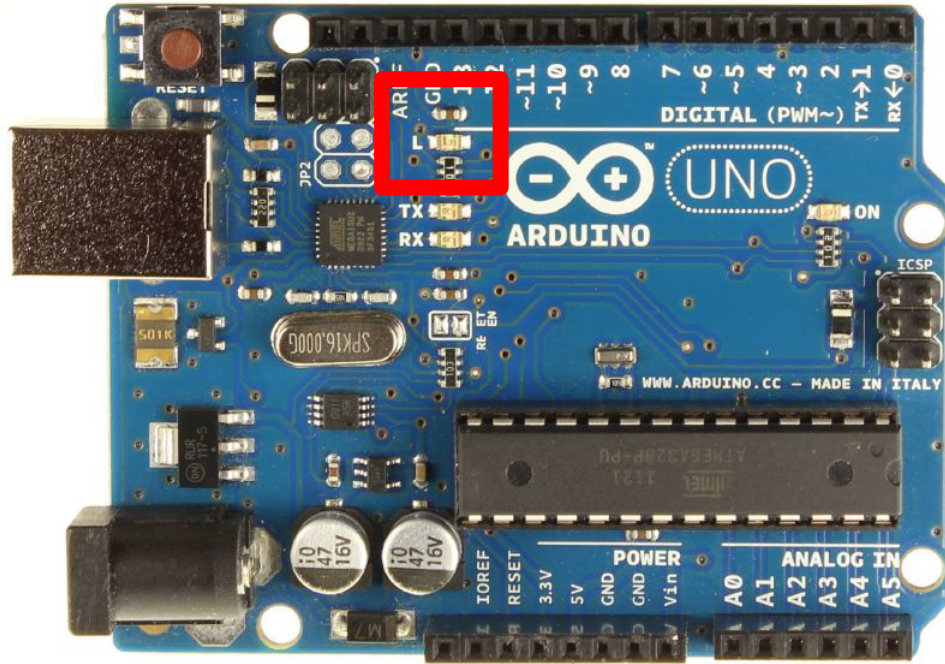
Vin, 5V, 3.3V, GND, Reset pins



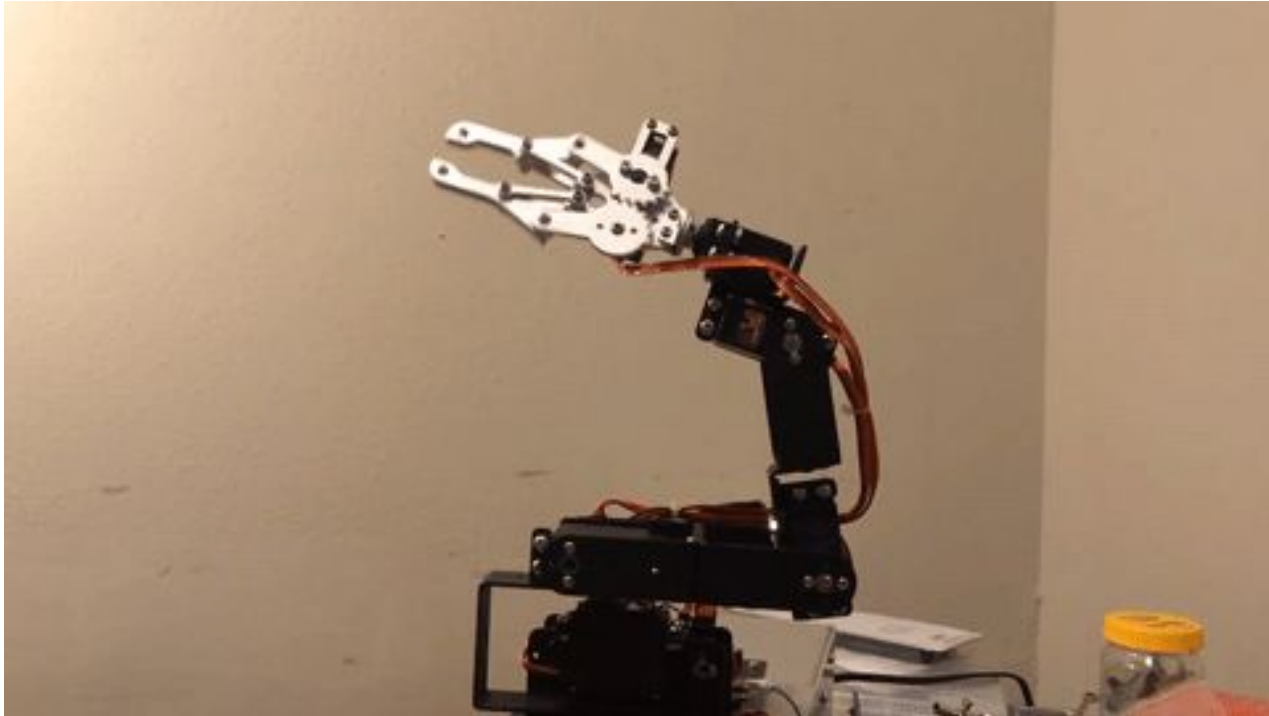
Back of Arduino



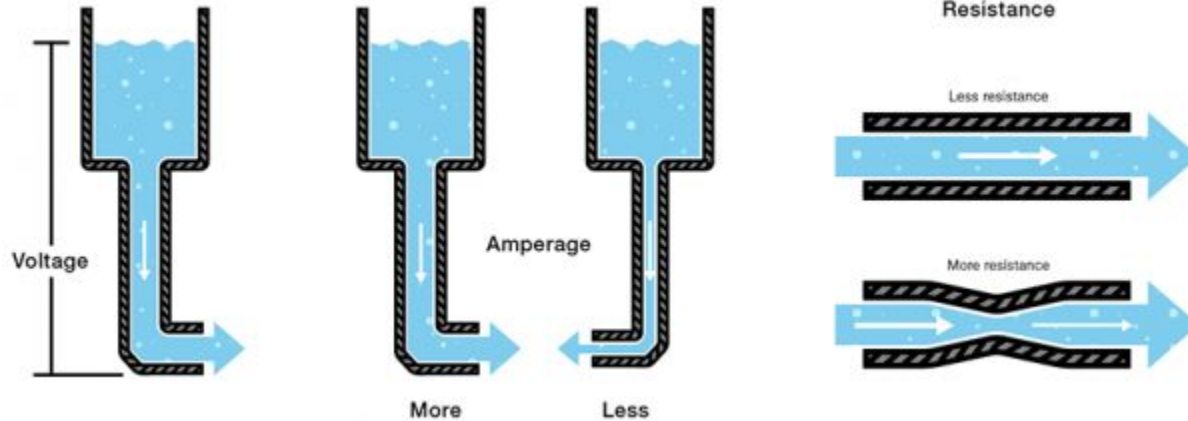
Internal LED



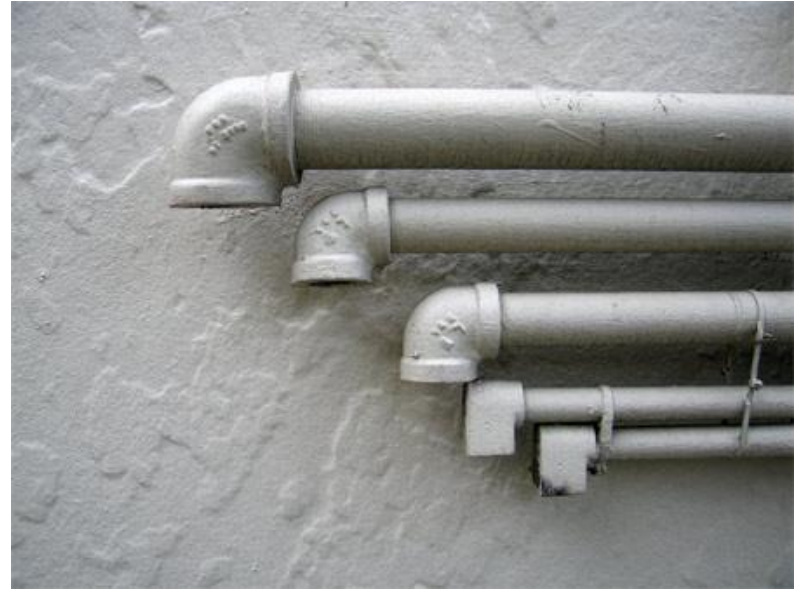
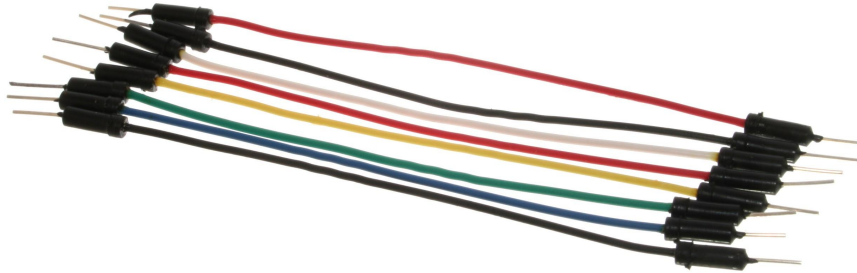
YAY ELECTRONICS!



Ohms Law:

$$V = IR$$


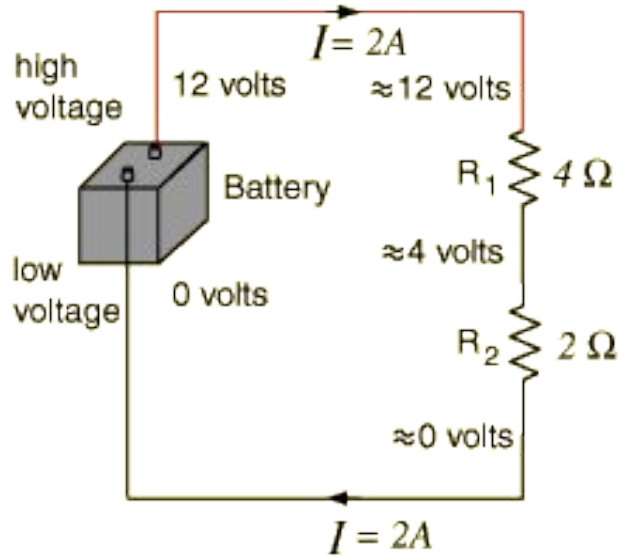
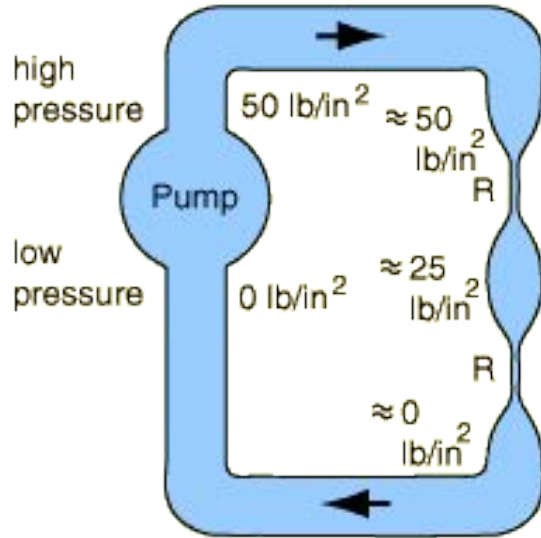
Wires = Pipes



Battery = Water Pump

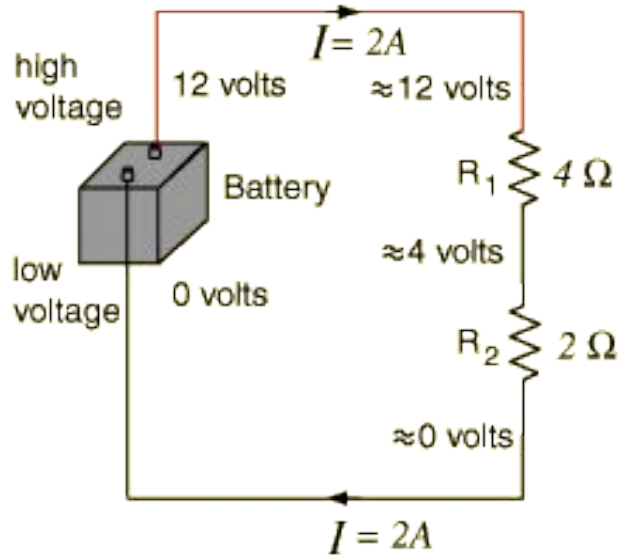
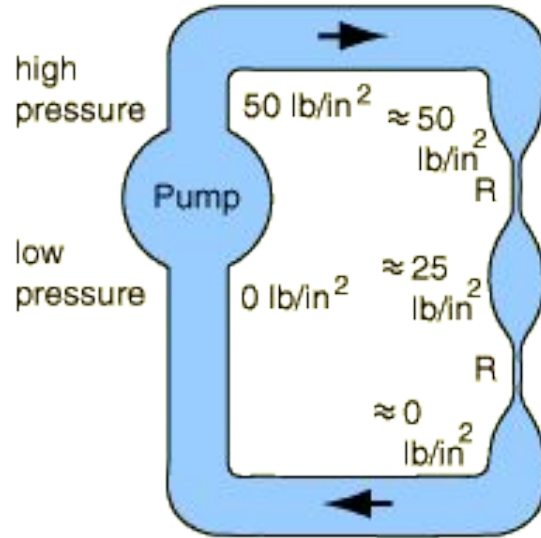


Voltage (V) = water pressure



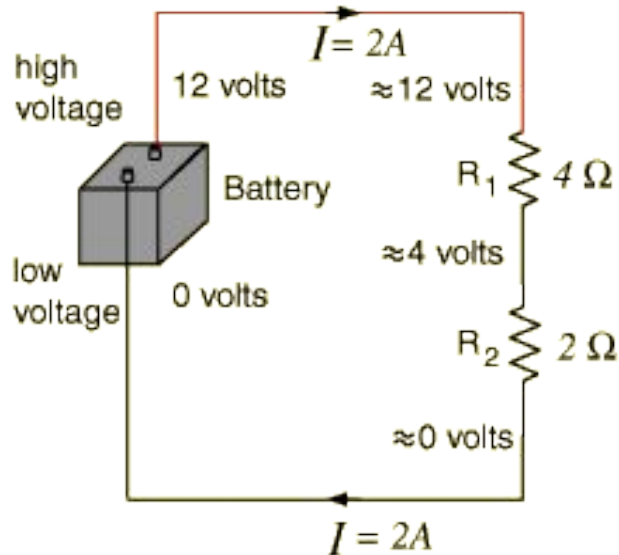
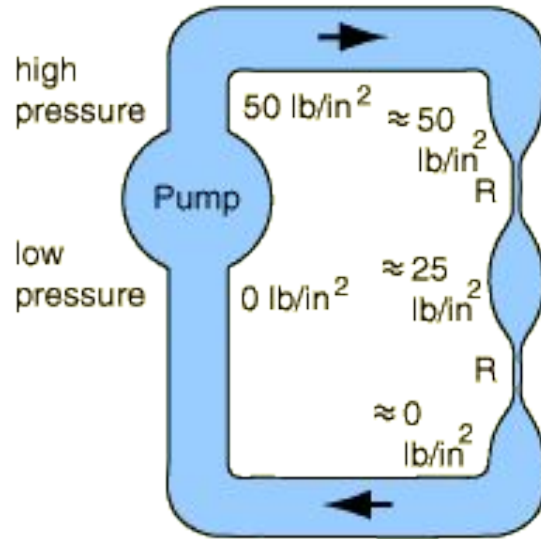
The force with which electrons are being pushed through the wire

Current (I) = how much water flow



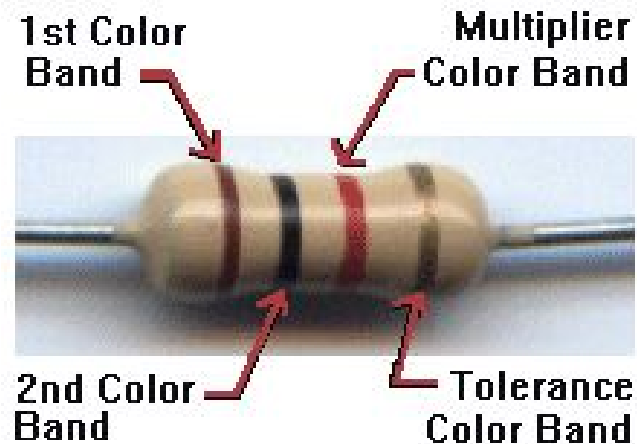
The amount of electrons moving through the wire at any given moment

Resistance (Ohm or Ω) = Pinched pipe



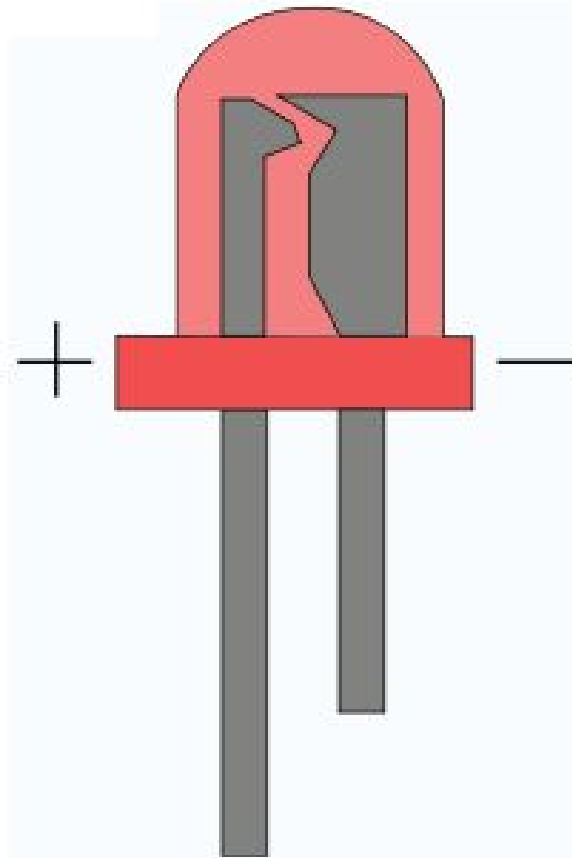
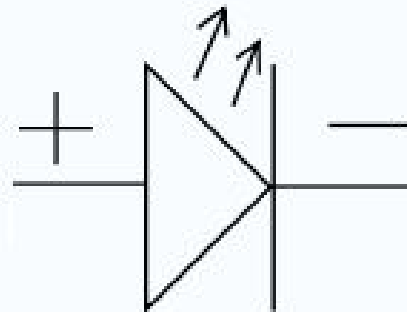
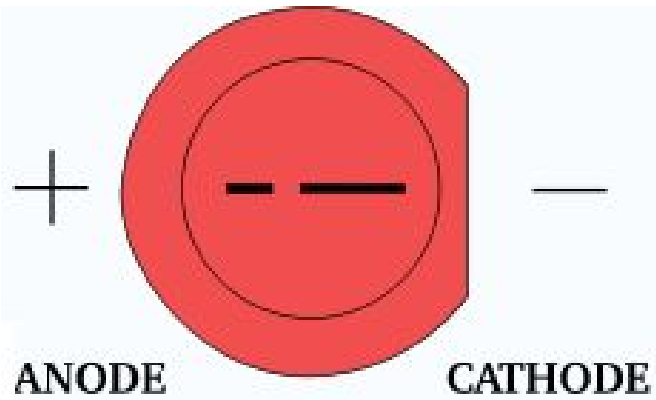
The amount of electrons moving through the wire at any given moment

Resistors



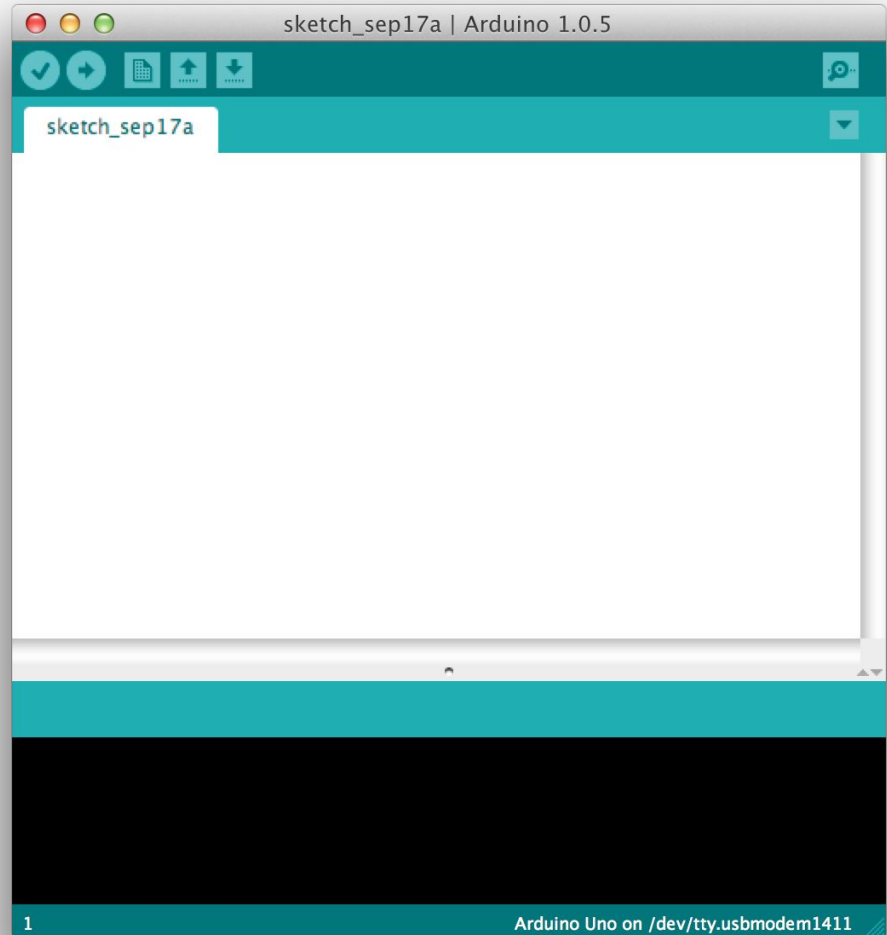
1st digit	2nd digit	Multiplier	Tolerance
0	0	x 1	
1	1	x 10	±1%
2	2	x 100	±2%
3	3	x 1K	
4	4	x 10K	
5	5	x 100K	
6	6	x 1M	
7	7		
8	8	x 0.1	±5%
9	9	x 0.01	±10%

LEDs



Arduino IDE

(Integrated Dev Environment)



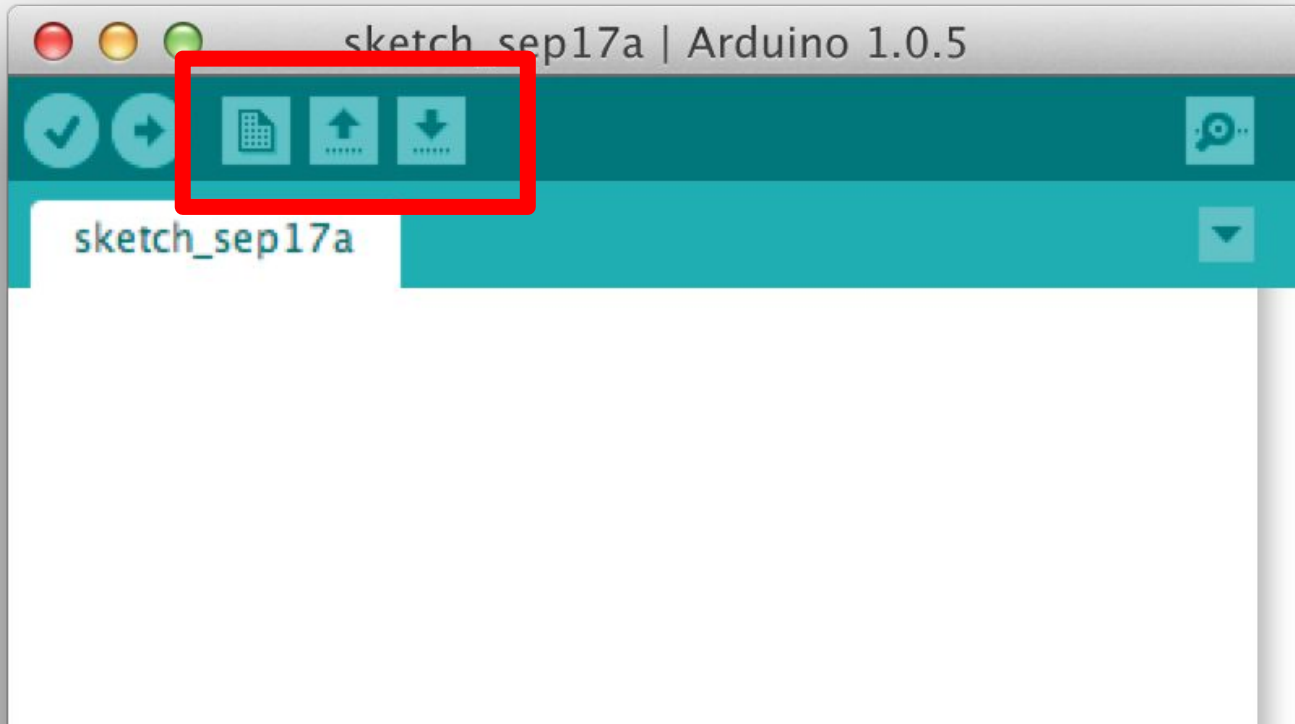
Verify Button



Upload Button



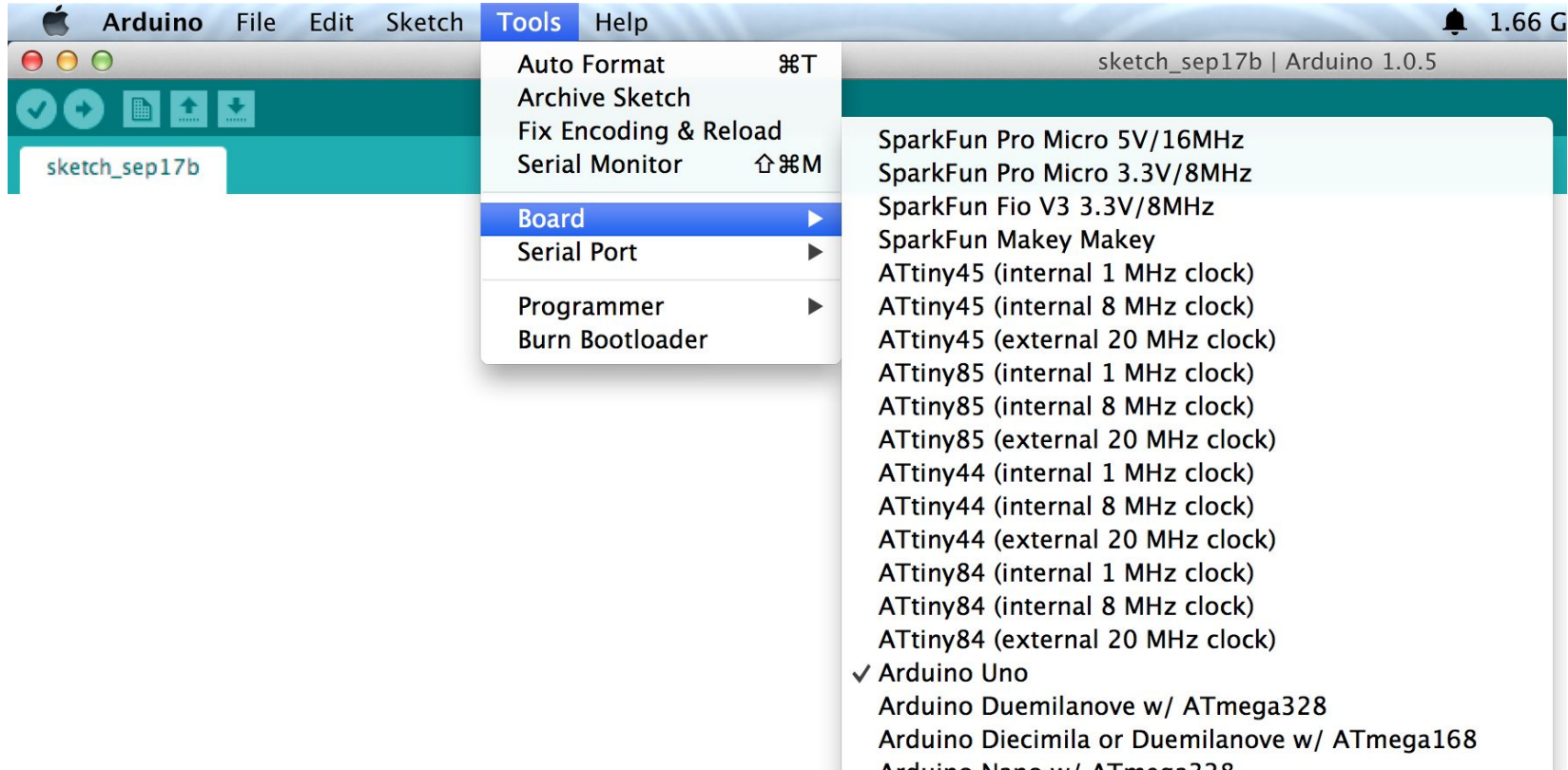
New, Open, Save...



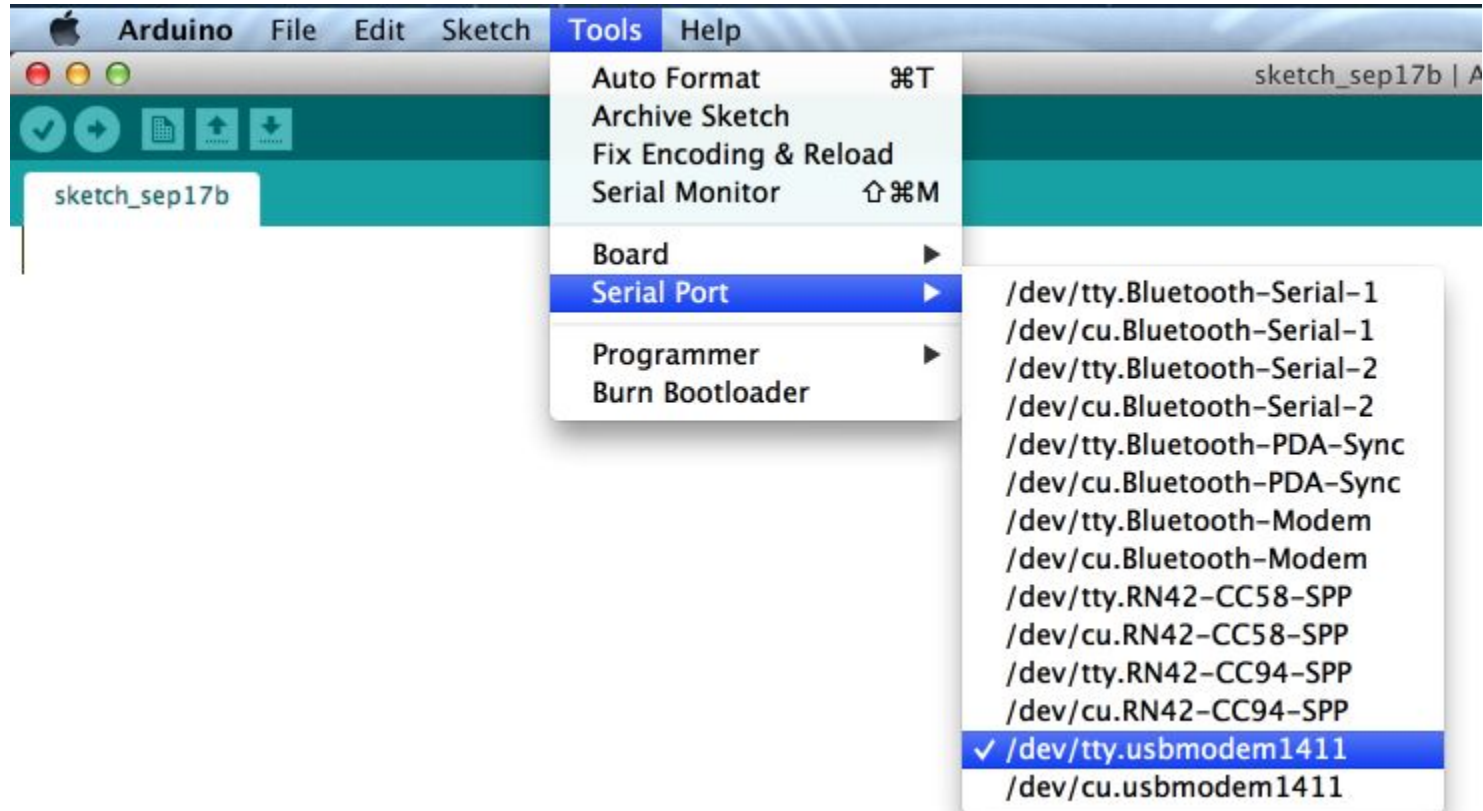
Serial Monitor



Tools > Board



Tools > Serial Port



File > Examples > Basics > Blink sketch

```
// 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  
}
```

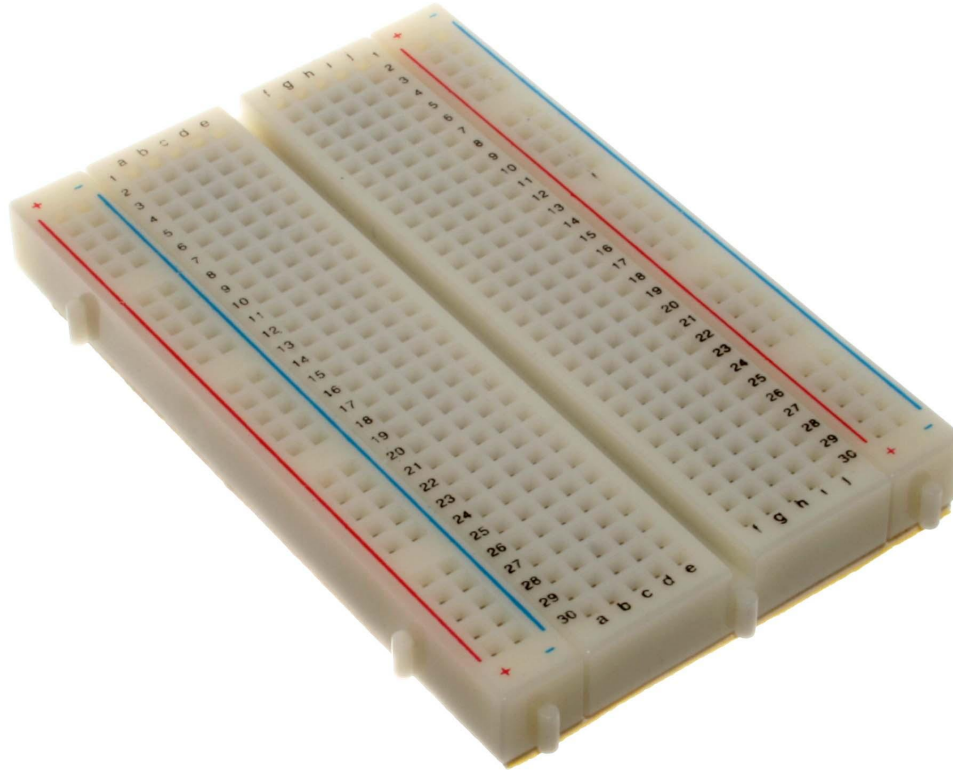
File > Examples > Basics > Blink sketch

```
// 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  
}
```

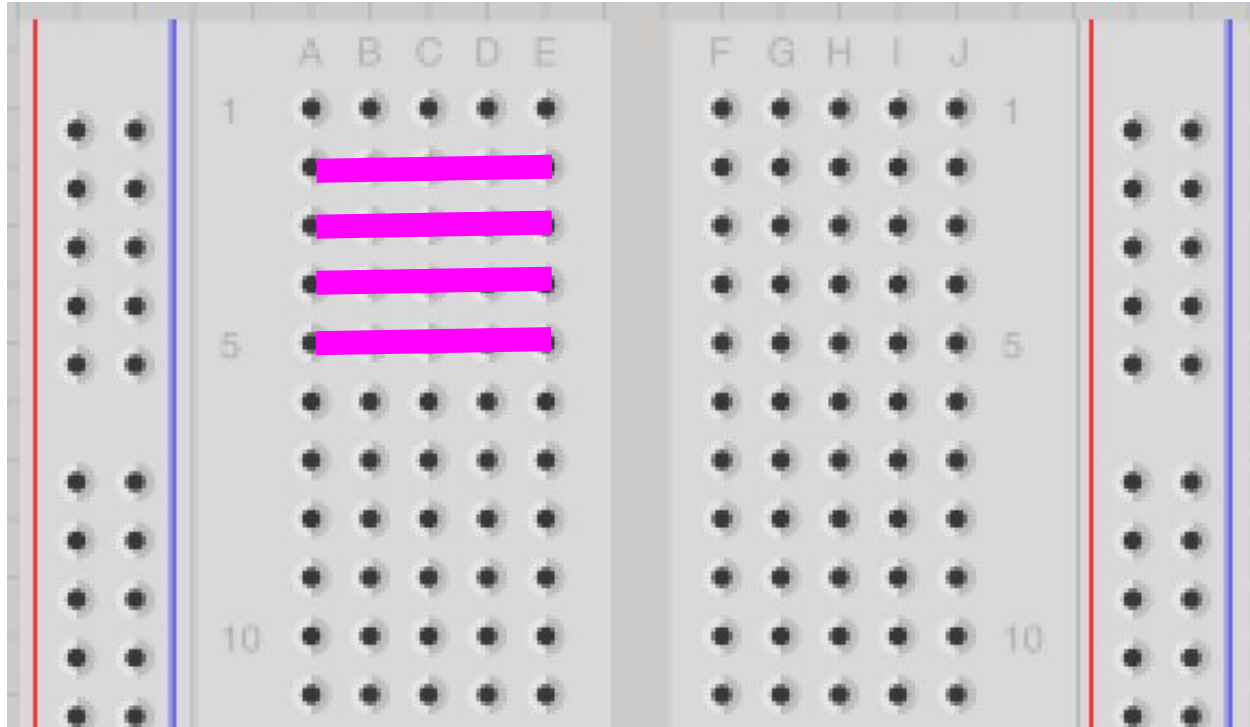

Add some “debugging” code

```
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  
}
```

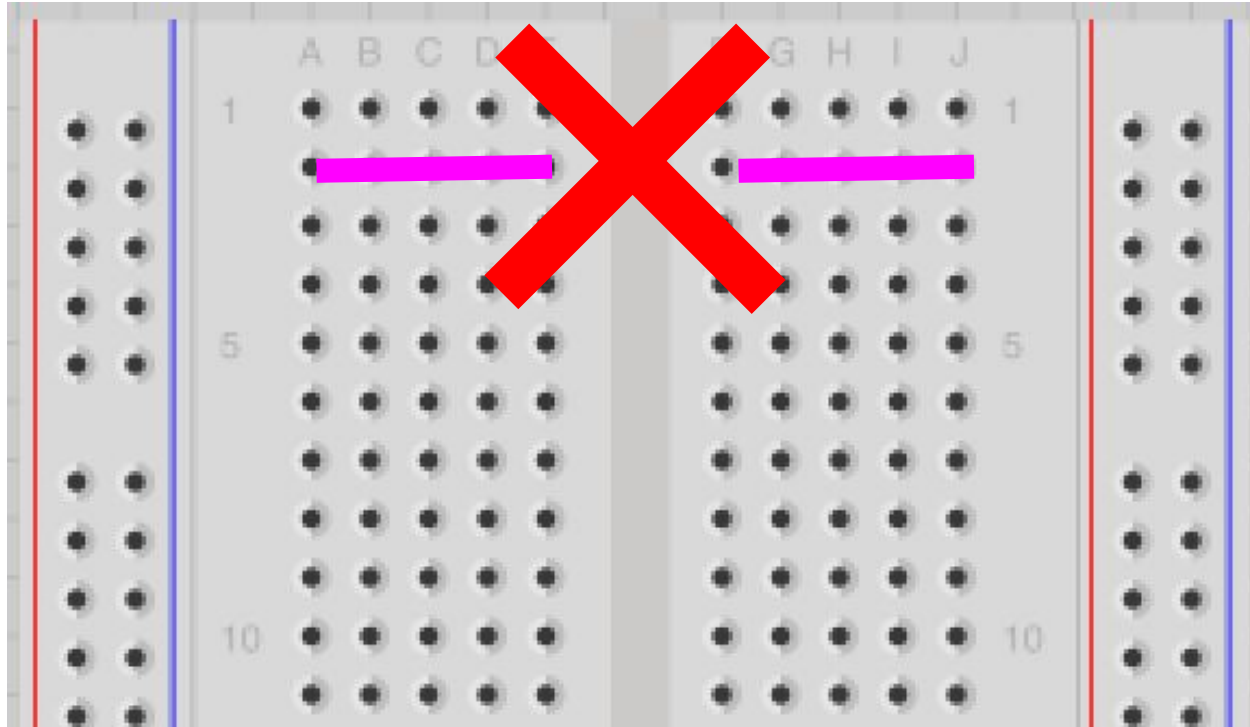
Breadboarding



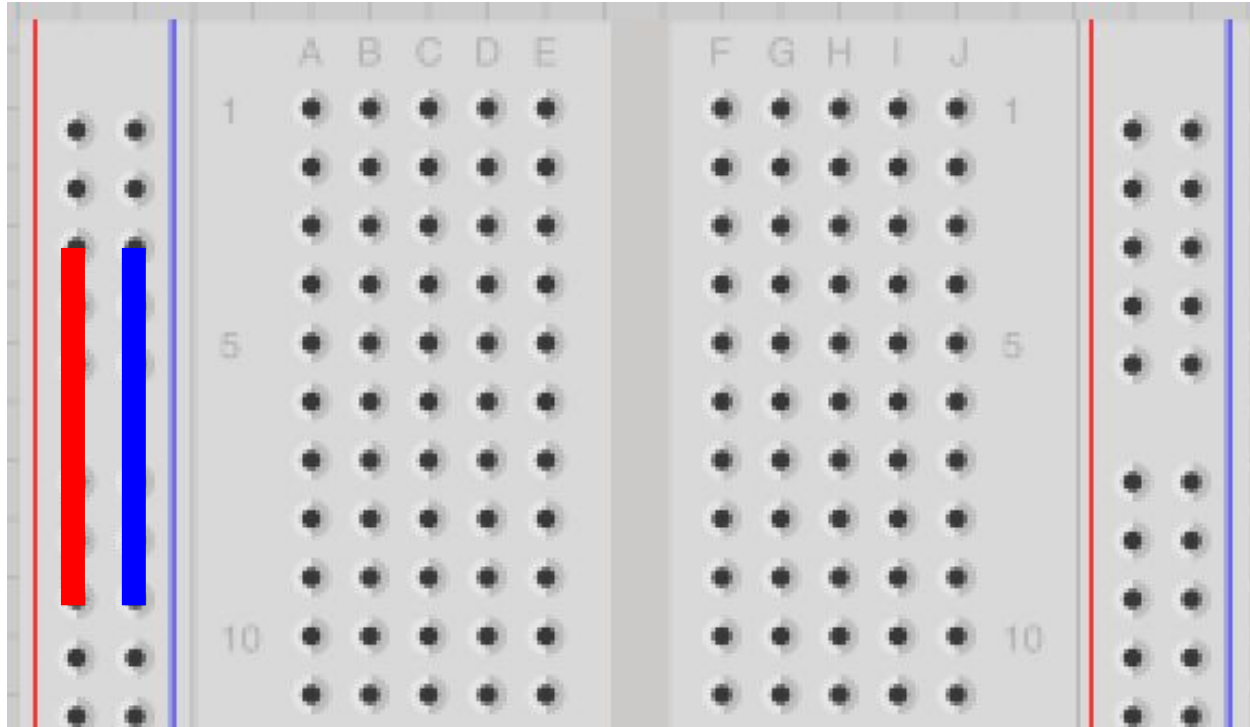
Breadboard Connections



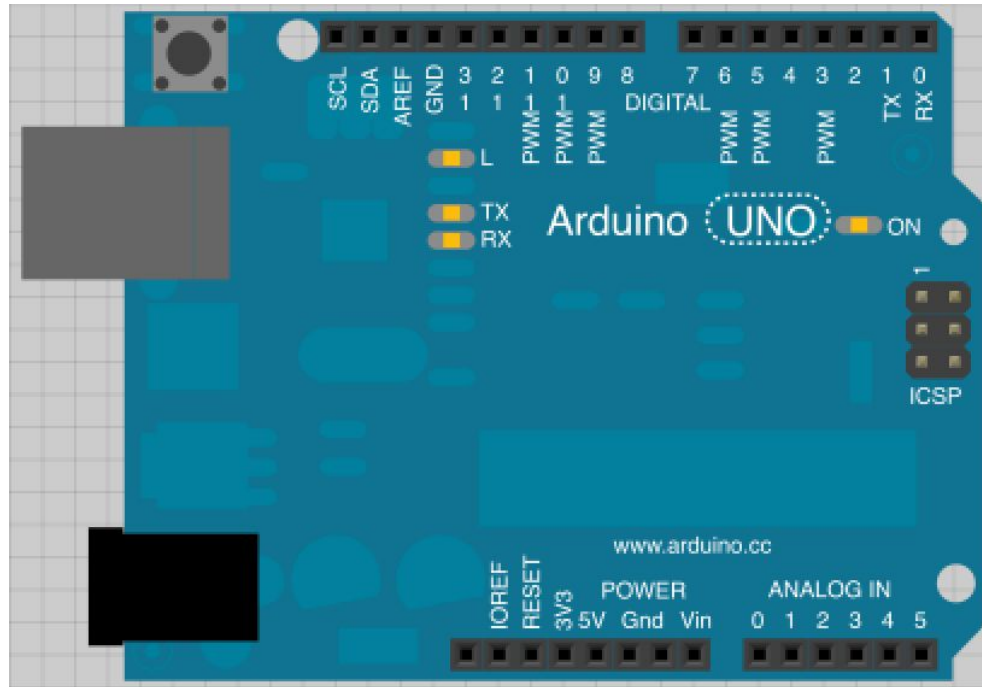
Breadboard Rows



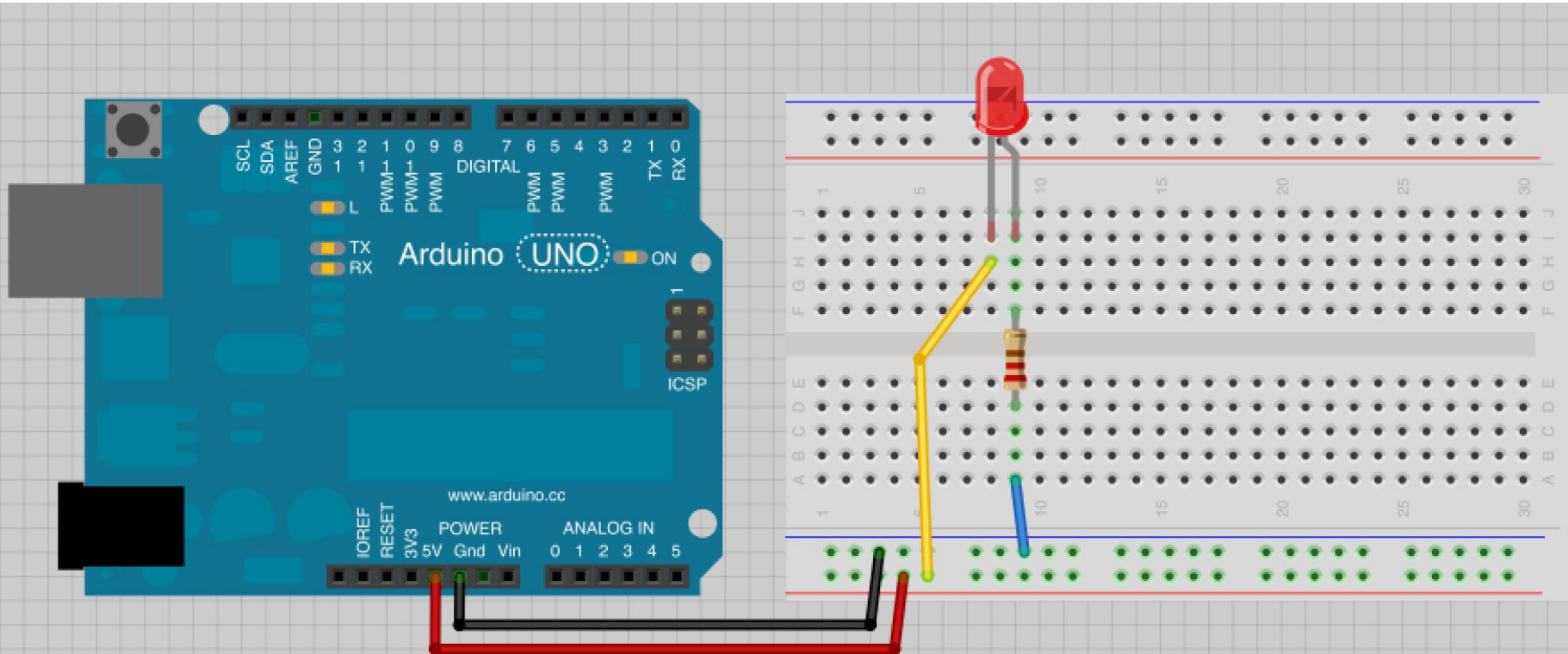
Breadboard Power/Ground “Rails”



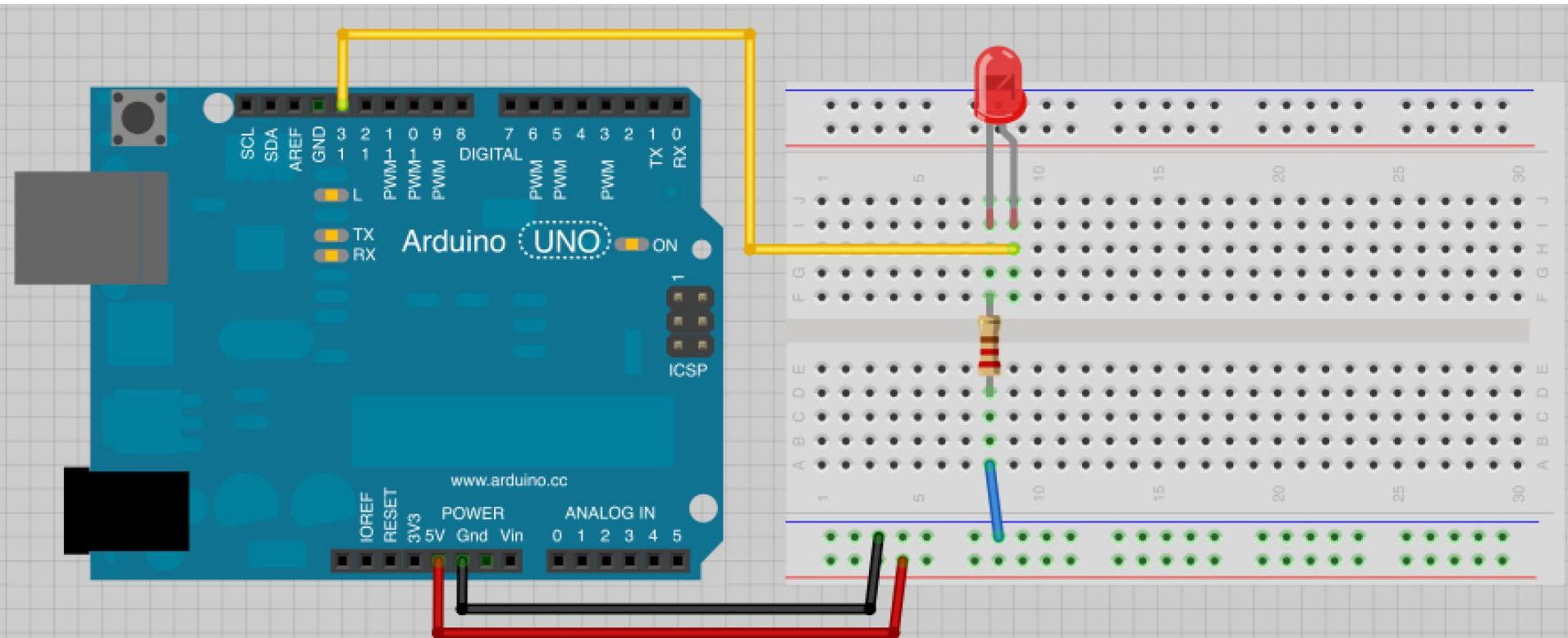
Connecting an LED



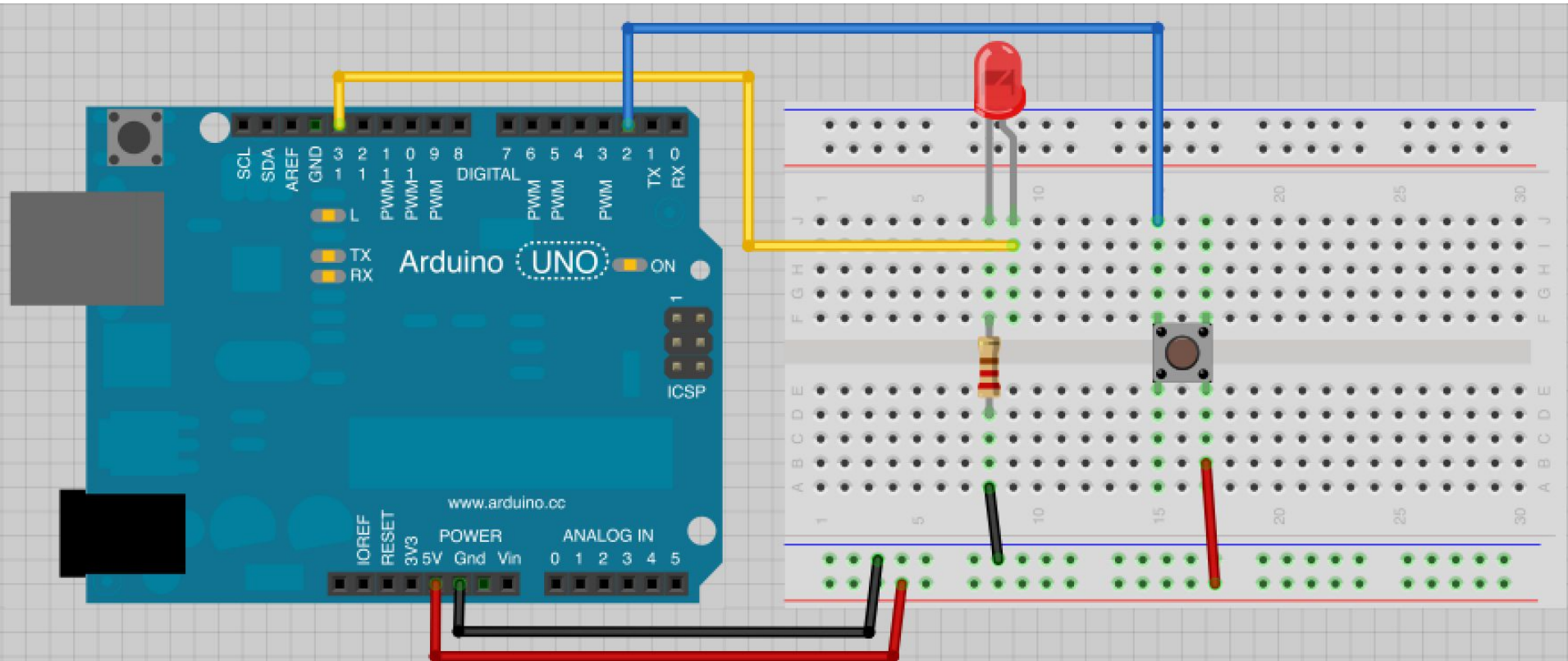
Connecting an LED



Control blink via Pin 13

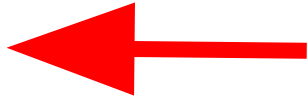


Manually blink using push button.



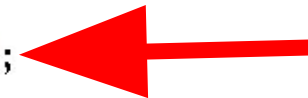
Read the button with code

```
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);  
}
```

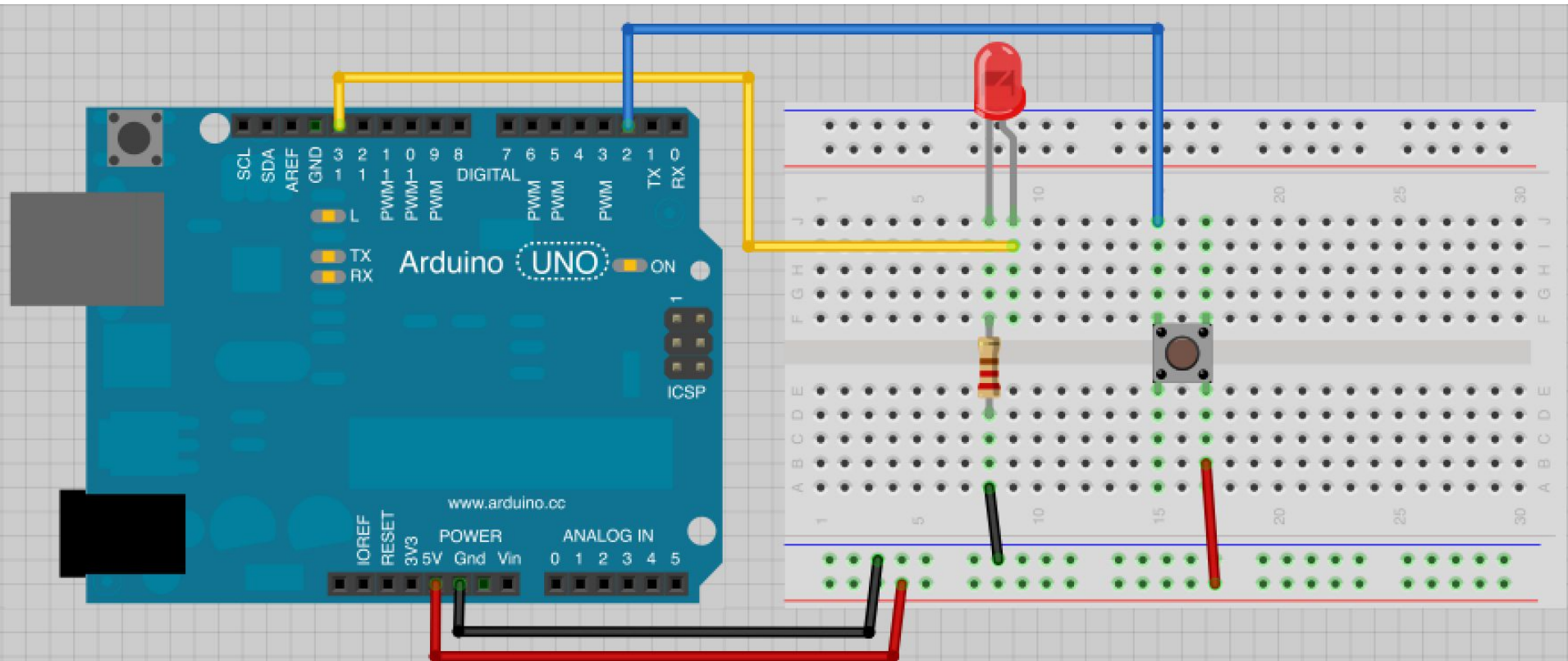


```
// the loop routine runs over and over again forever:
```

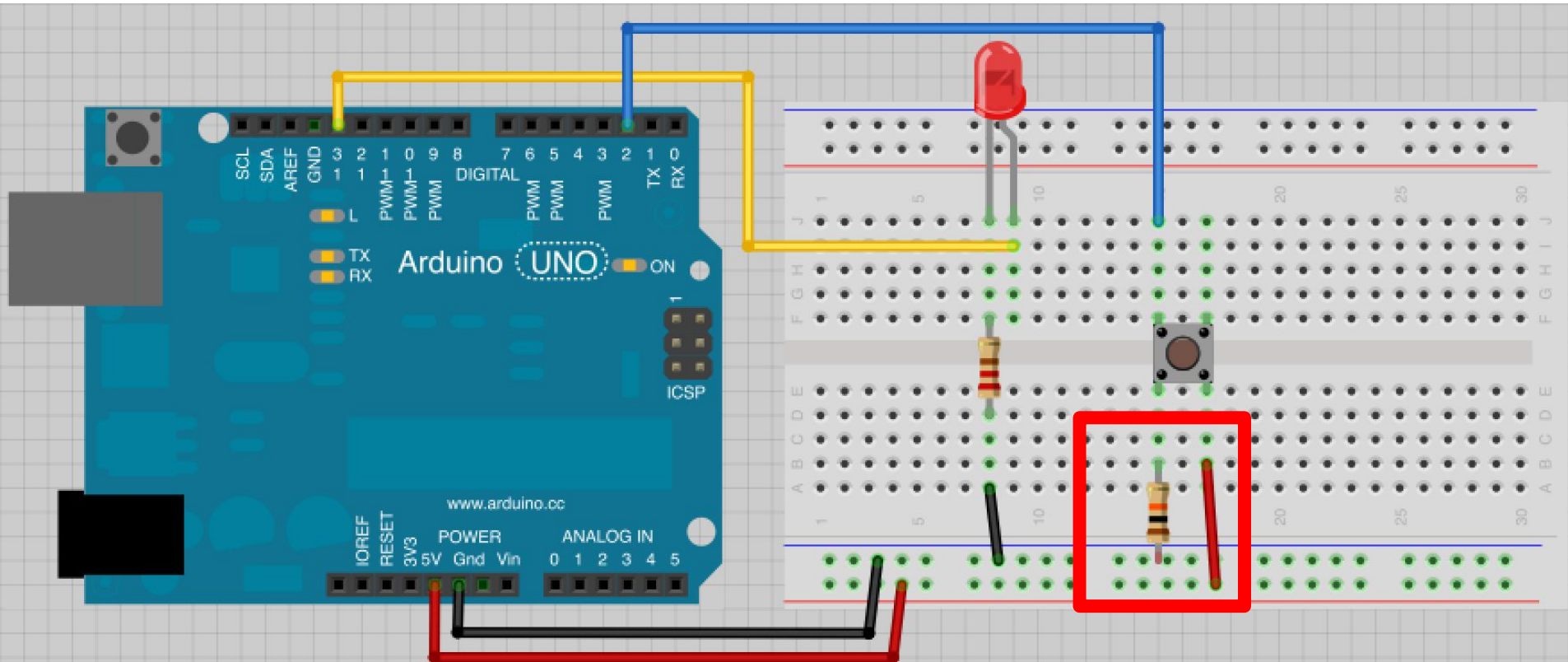
```
void loop() {  
  //read the button  
  buttonState = digitalRead(buttonPin);  
  //Perform different actions depending on the state of the button  
  if(buttonState == HIGH){  
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
    delay(1000);             // wait for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



Connecting a Button

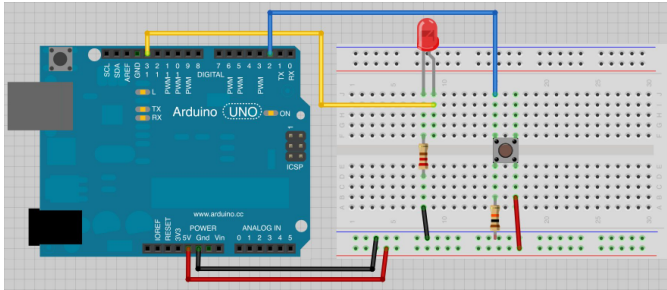


Add a “pull-down” resistor

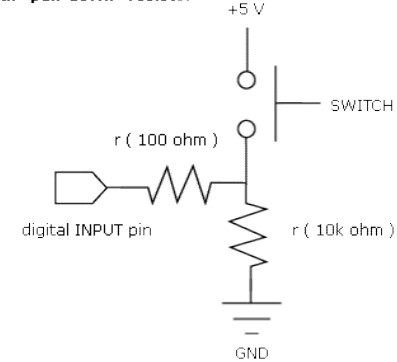


Pull-Up / Pull-Down Resistors

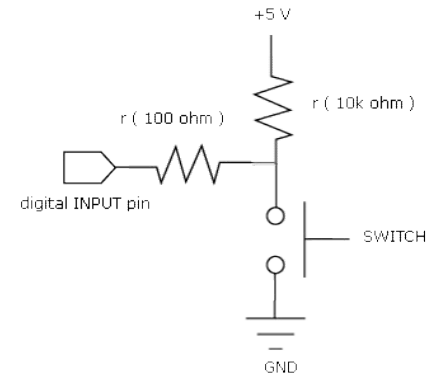
ensure that the signal will be a valid logic level if external devices are disconnected



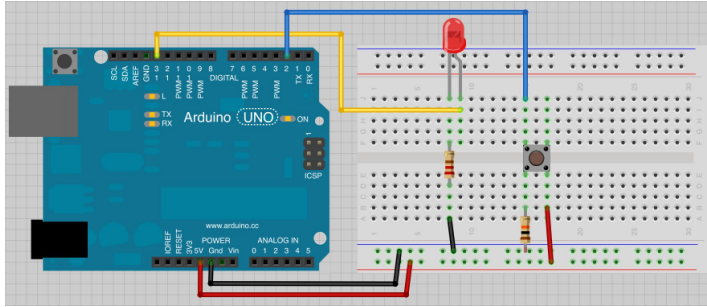
Switch with "pull-down" resistor



Switch with "pull-up" resistor



HOMEWORK



Update your code so the button triggers a state change

For example: the LED stays on when you push it and turns off when the button is pressed again

Try out different blink patterns

Documentation:

Upload your code to github with a link to the online video