INTRODUCTION TO MICRO CONTROLLERS WITH ARDUINO

WHAT IS A MICROCONTOLLER:

Most of us know what a computer looks like. It usually has a keyboard, monitor, CPU (Central Processing Unit), printer, and a mouse. These types of computers, like the Mac or PC, are primarily designed to communicate (or "interface") with humans.

WHAT IS A MICROCONTOLLER:

But did you know that there are computers all around us, running program and quietly doing calculations, not interacting with humans at all? These computers are in your car, on the Space Shuttle, in your kid brother's toy, and maybe even inside your hairdryer.

WHAT IS A MICROCONTOLLER:

We call these devices "microcontrollers". Micro because they're small, and controller because they "control" machines, gadgets, whatever.

They're cool because, you can build a machine or device,
write programs to control it and then let it work for you automatically.

WHAT IS A MICROCONTOLLER:

There is an infinite number of applications for microcontrollers. Your imagination is the only limiting factor!

WHAT IS ARDUINO:

Arduino is an open source physical computing platform based on a simple input/output (I/O) board and a development environment that implements the Processing language (www.processing.org).

Arduino can be used to develop standalone interactive objects
or can be connected to software on your computer (such as Flash, Processing, NodeJS or Max/MSP).

PHYSICAL COMPUTING?:

Physical Computing uses electronics to prototype new materials for designers and artists.

It involves the design of interactive objects that can communicate with humans using sensors and actuators controlled by a behavior implemented as software running inside a microcontroller (a small computer on a single chip).

PHYSICAL COMPUTING?:

In the past, using electronics meant having to deal with engineers, and building circuits one small component at a time. These issues kept creative people from experimenting with the medium directly. In recent years, micro-controllers have become cheaper and easier to use.

PHYSICAL COMPUTING?:

The progress that has been made with Arduino has been to bring these tools one step closer to the non expert allowing people to start building things after only two or three days of a workshop.

With Arduino, a designer or artist can get to know the basics of electronics and sensors very quickly and can start building prototypes with very little investment.

THE ARDUINO PLATFORM

Arduino is composed of two major parts:

-The Arduino board, which is the piece of hardware you program to control your projects.

-The Arduino IDE, the piece of software you run on your computer. You use the IDE to create a sketch(a program) that you upload to the Arduino board. The sketch tells the board what to do.

ARDUINO TERMS

"sketch" – a program you write to run on an Arduino board

"pin" – an input or output connected to something. e.g. output to an LED, input from a knob.

"digital" – value is either HIGH or LOW. (aka on/off, one/zero) e.g. switch state

"analog" – value ranges, usually from 0-255. e.g. LED brightness, motor speed, etc.

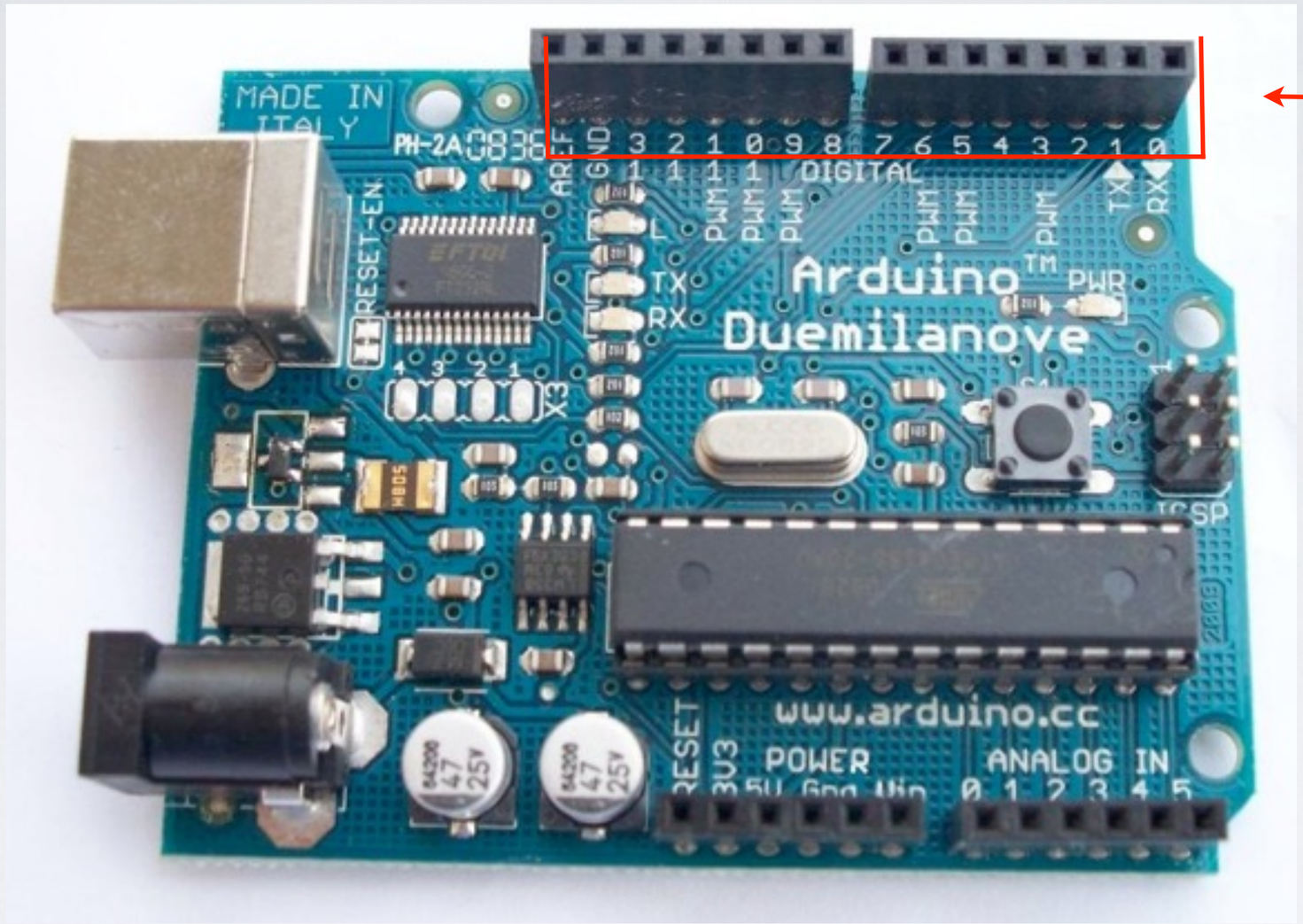
THE ARDUINO HARDWARE

The Arduino board is a microcontroller board, which is a small circuit (the board) that contains a whole computer on a small chip (the micro- controller). This computer is at least a thousand times less powerful than the MacBook I'm using to write this, but it's a lot cheaper and very useful to build interesting devices.

THE ARDUINO HARDWARE

The Arduino design team have placed on this board all of the components that are required for this micro controller to work properly and to communicate with your computer. There are many versions of this board; the one we'll use throughout this book is the Arduino Uno, which is the simplest one to use and the best one for learning on.

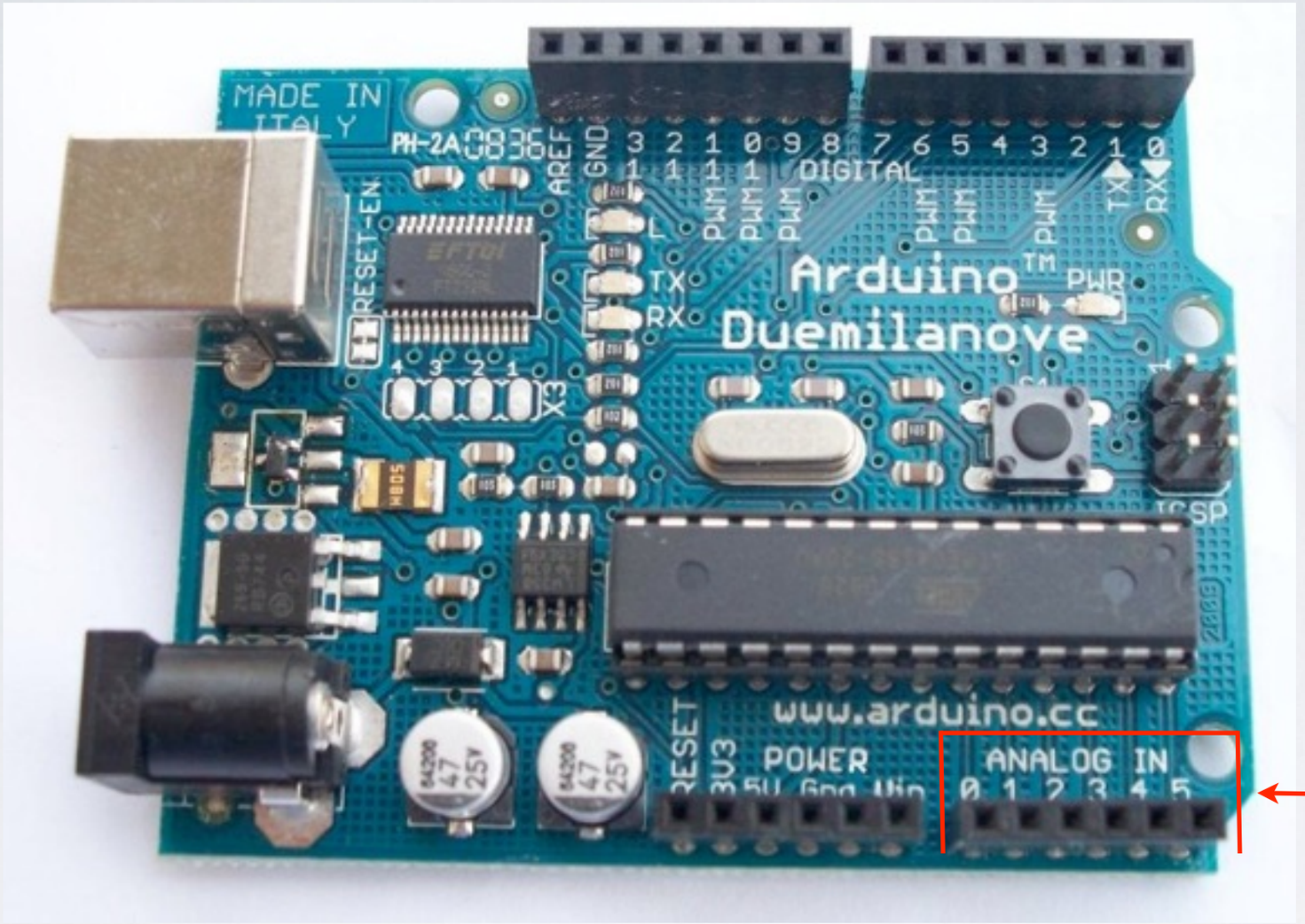
THE ARDUINO HARDWARE



14 Digital IO pins (pins 0–13)

These can be inputs or outputs, which is specified by the sketch you create in the IDE.

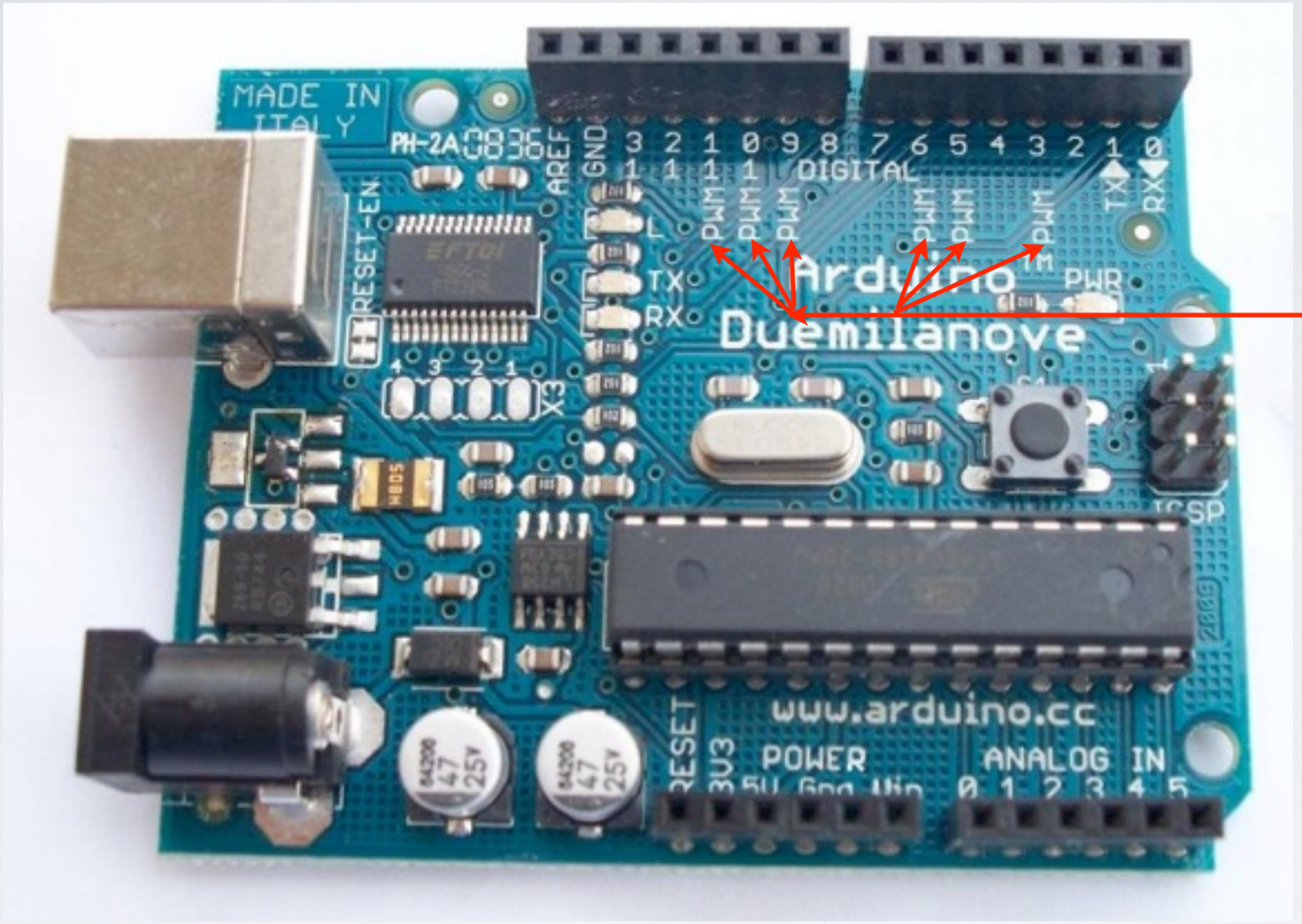
THE ARDUINO HARDWARE



Analogue In pins (pins 0–5)

These dedicated analogue input pins take analog values (i.e., voltage readings from a sensor) and convert them into a number between 0 and 1023 we'll look at this Thursday

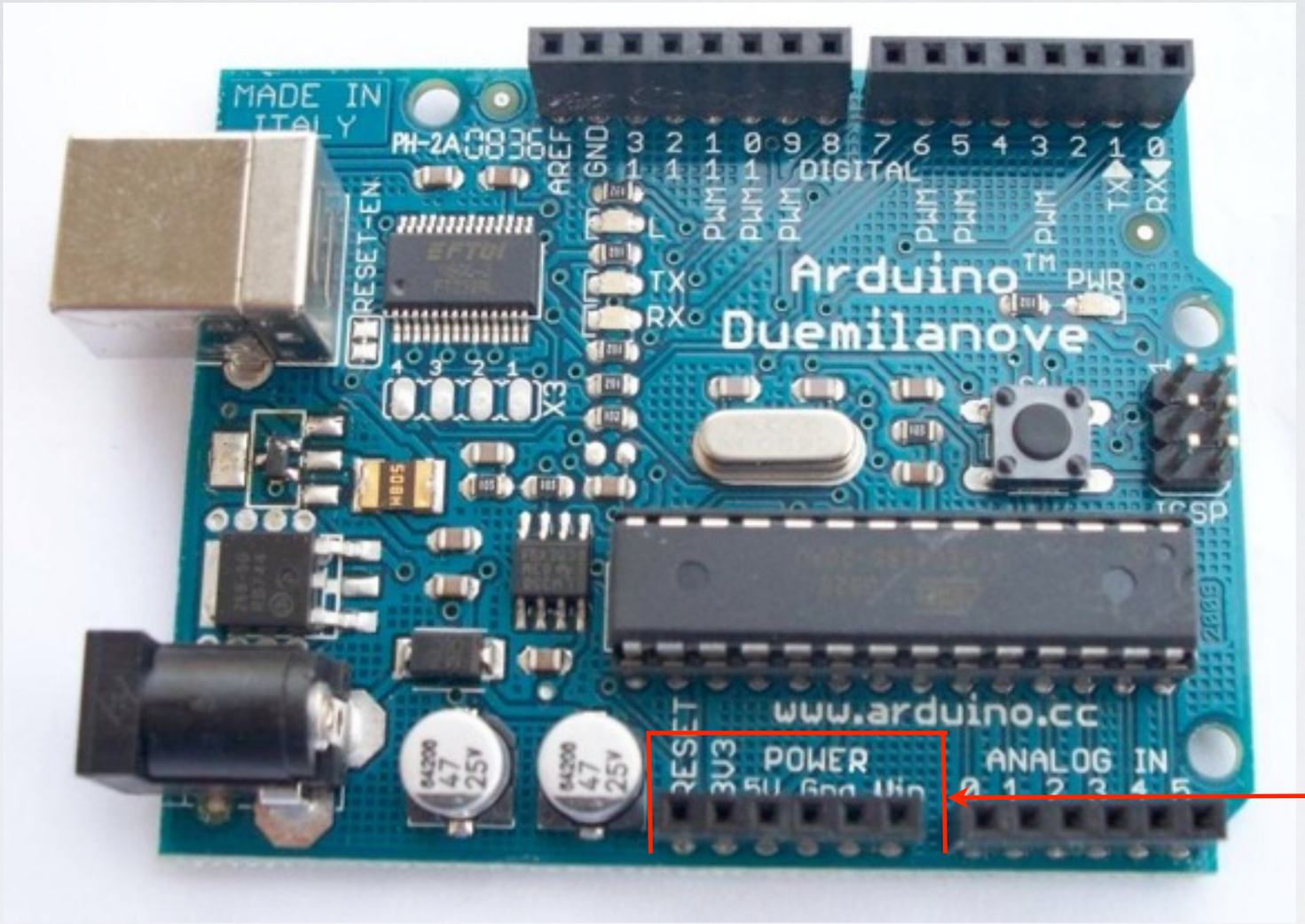
THE ARDUINO HARDWARE



Analogue Out pins (pins 3, 5, 6, 9, 10, and 11)

These are actually six of the digital pins that can be reprogrammed for analog output we'll look at this Thursday

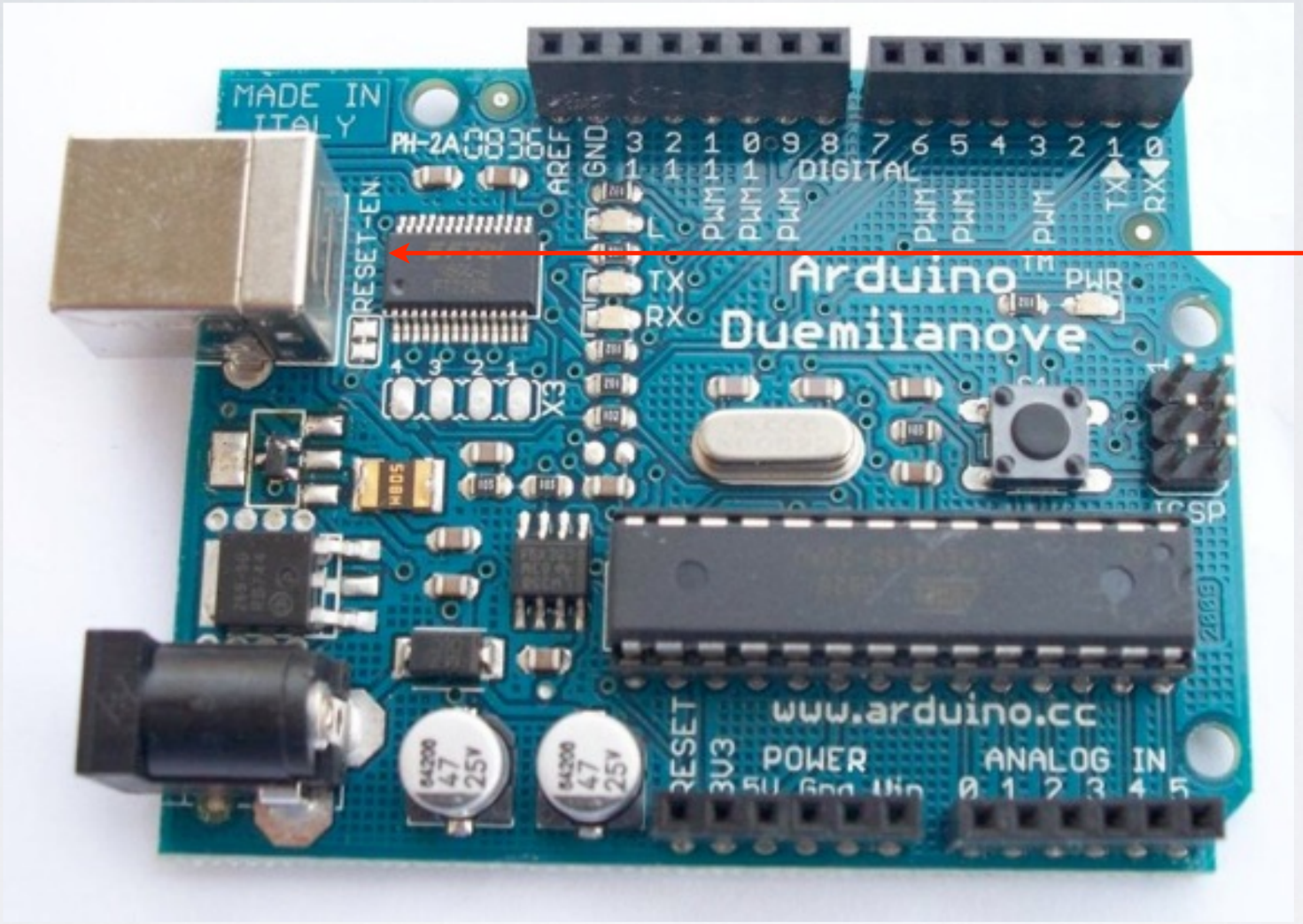
THE ARDUINO HARDWARE



These power pins are capable or providing 5.5 volts or 3.3 volts of power. Also there are two pins that are capable of providing ground any components that you attach to the controller.

*We'll cover Power, Ground and Voltage in just a bit.

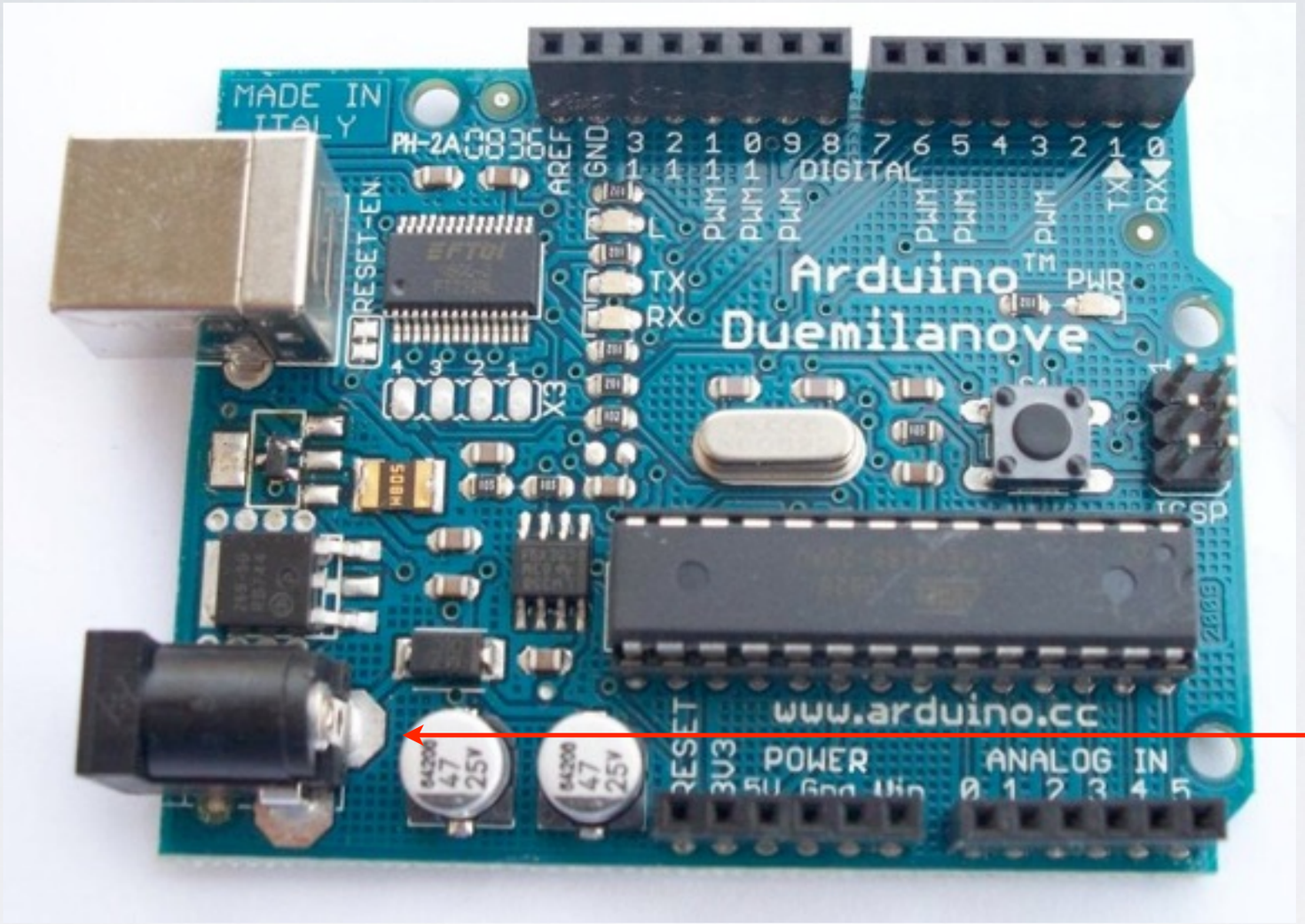
THE ARDUINO HARDWARE



USB Connector

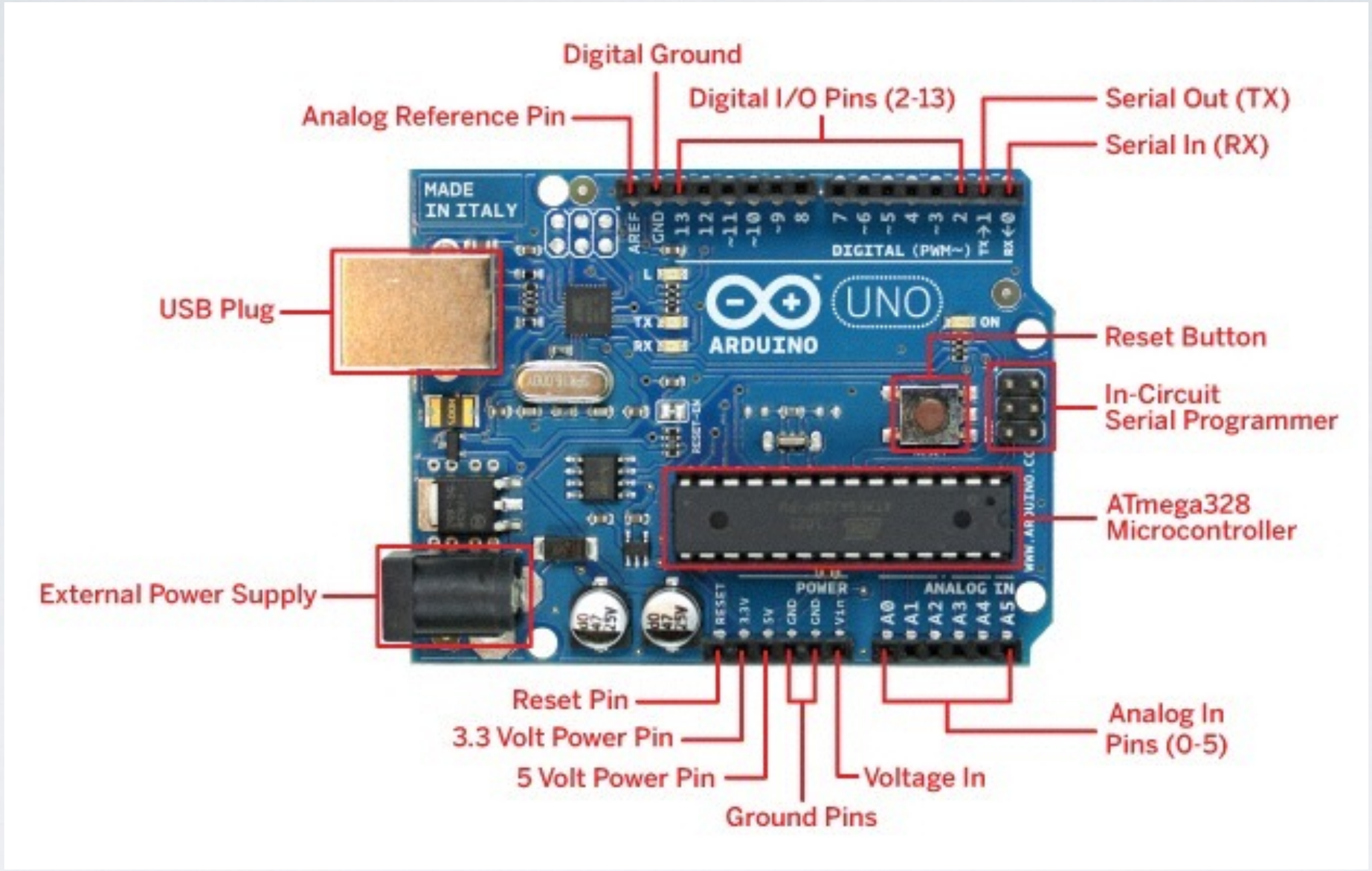
The USB connector connects the controller to you computer allowing you to program it. The USB connection also provides power.

THE ARDUINO HARDWARE

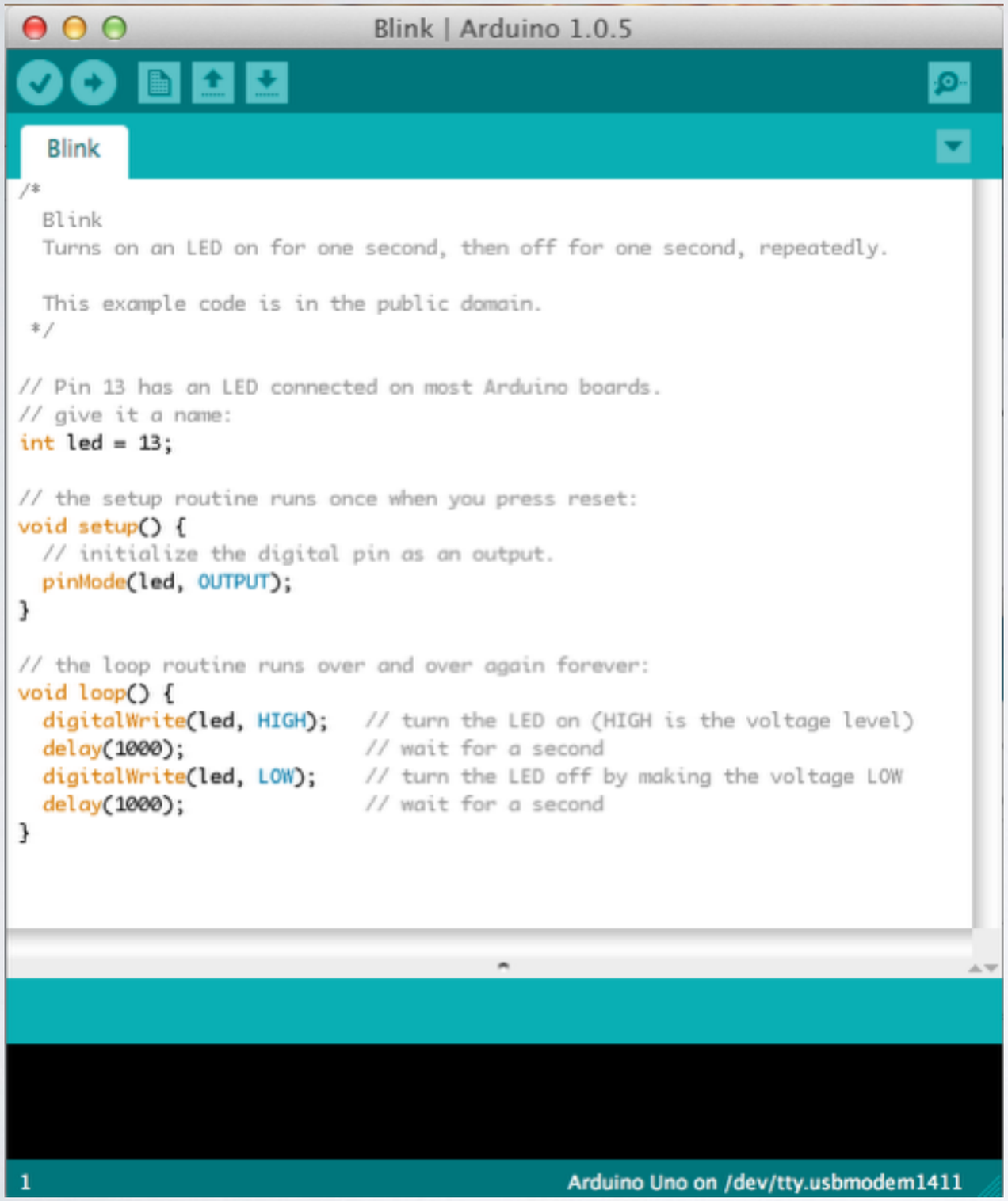


The power connector allows you to power your controller independently either with a battery or a power supply.

THE ARDUINO HARDWARE



THE ARDUINO SOFTWARE (IDE)



-Like a text editor

-View/write/edit sketches

-But then you program them into hardware

THE ARDUINO SOFTWARE (IDE)

The IDE (Integrated Development Environment) is a program running on your computer that allows you to write sketches for the Arduino board in a simple language modeled after the Processing (www.processing.org) language.

THE ARDUINO SOFTWARE (IDE)

The magic happens when you press the button that uploads the sketch to the board: the code that you have written is translated into the C language (which is generally quite hard for a beginner to use), and is compiled into a language that the micro controller understands.

This last step is quite important, because it's where Arduino makes your life easy by hiding away as much as possible of the complexities of programming microcontrollers.

THE ARDUINO SOFTWARE (IDE)

The programming cycle on Arduino is basically as follows:

» Plug your board into a USB port on your computer.

» Write a sketch that will bring the board to life.

» Upload this sketch to the board through the USB connection and wait a couple of seconds for the board to restart.

» The board executes the sketch that you wrote.

LESSON 1 - BASIC HOUSEKEEPING

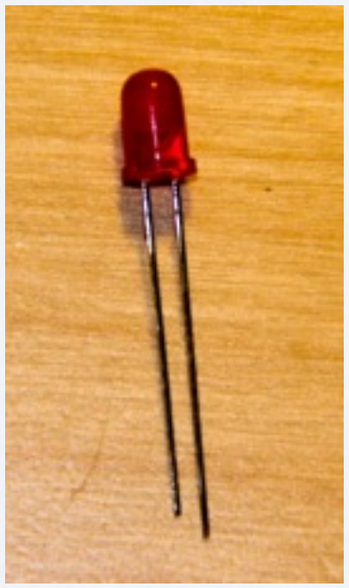
This first lesson won't really teach programming the Arduino. It's meant to introduce you to building a basic electrical circuit.

It will also get us prepped for building the rest of the exercises throughout the class.

LESSON 1 - GATHER THE PARTS

1 220 ohm resistor (bands are red,red,violet, gold)

1 red Light Emitting Diode(LED)

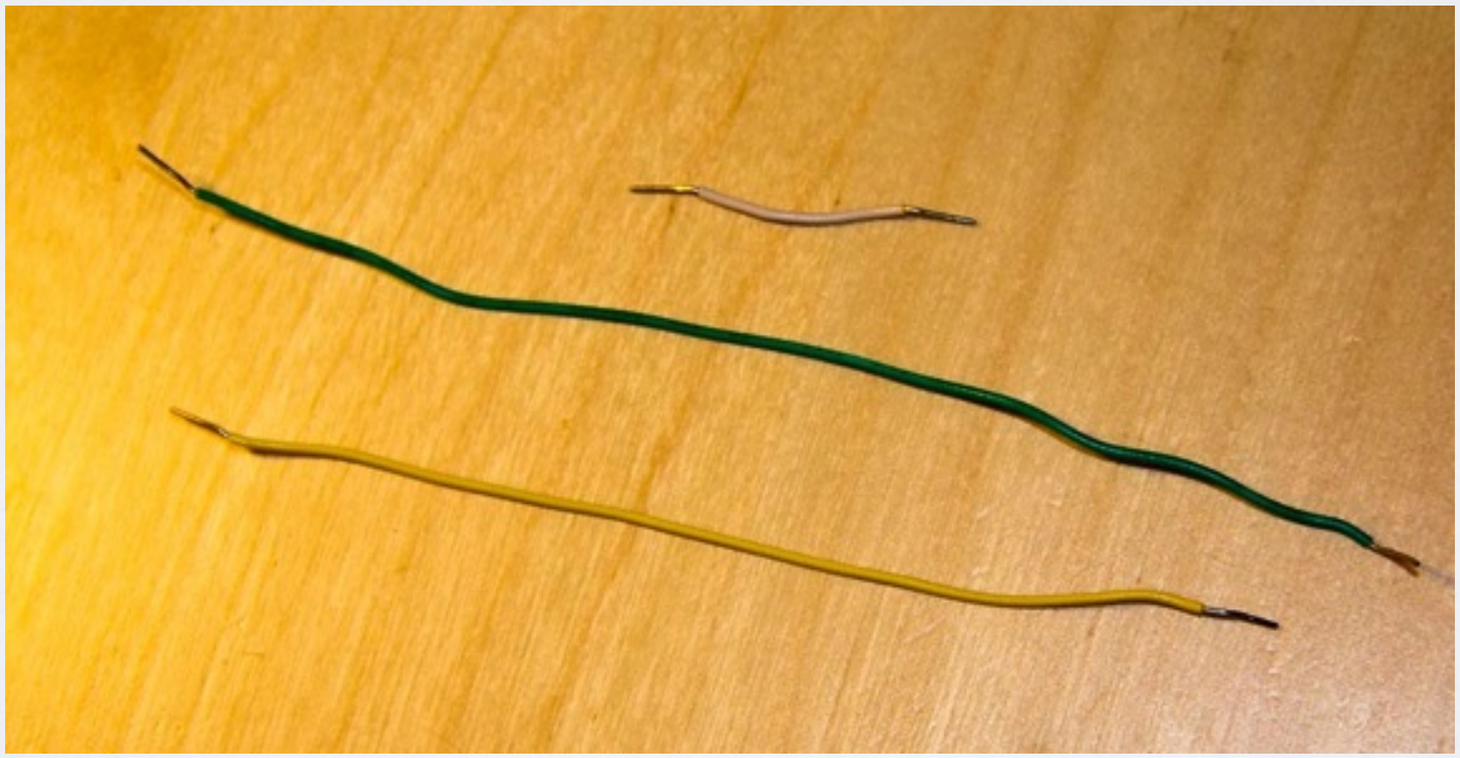


LESSON 1 - GATHER THE PARTS

1 long yellow wire

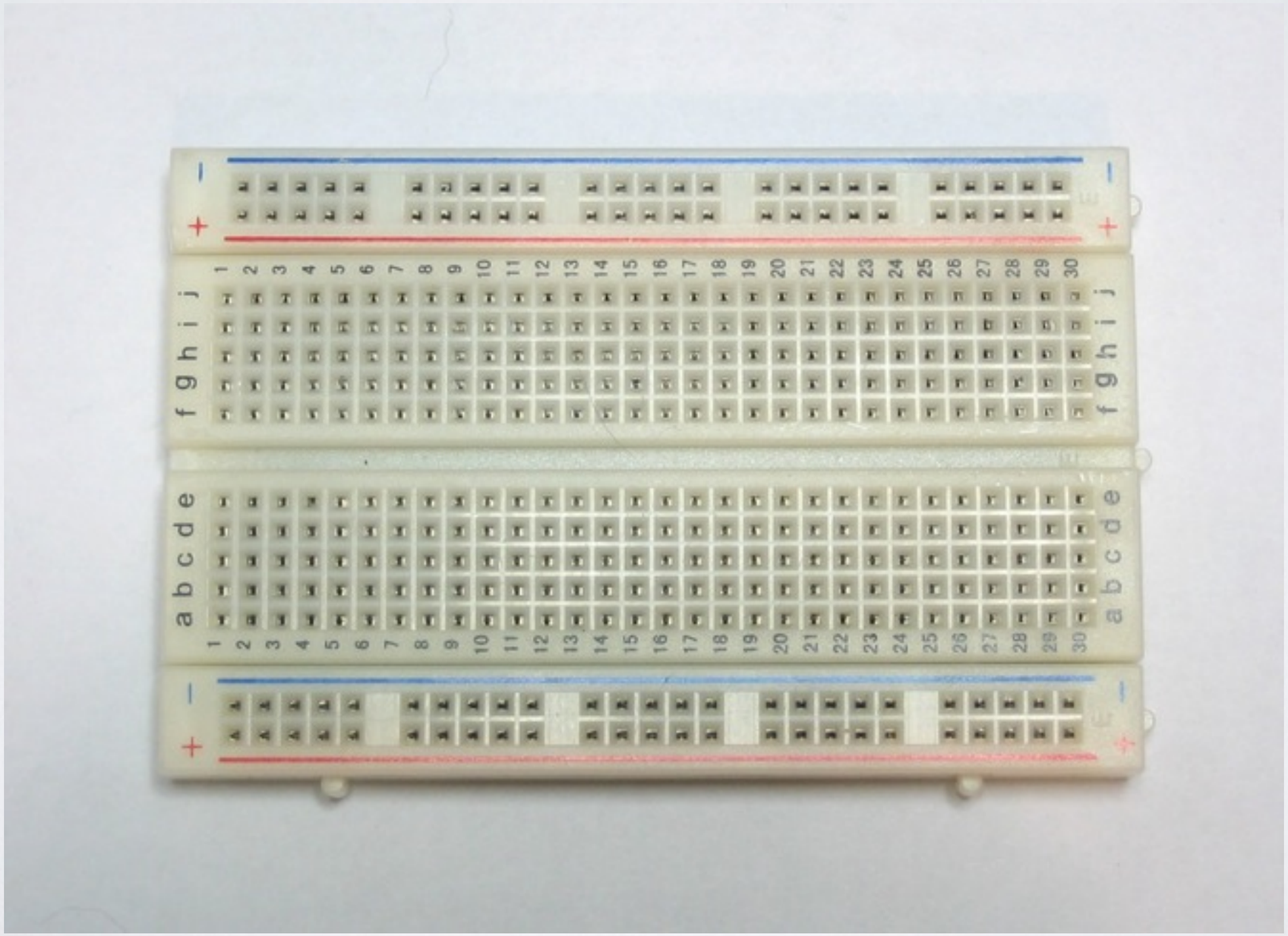
1 long green wire

1 short white wire



LESSON 1 - GATHER THE PARTS

1 breadboard

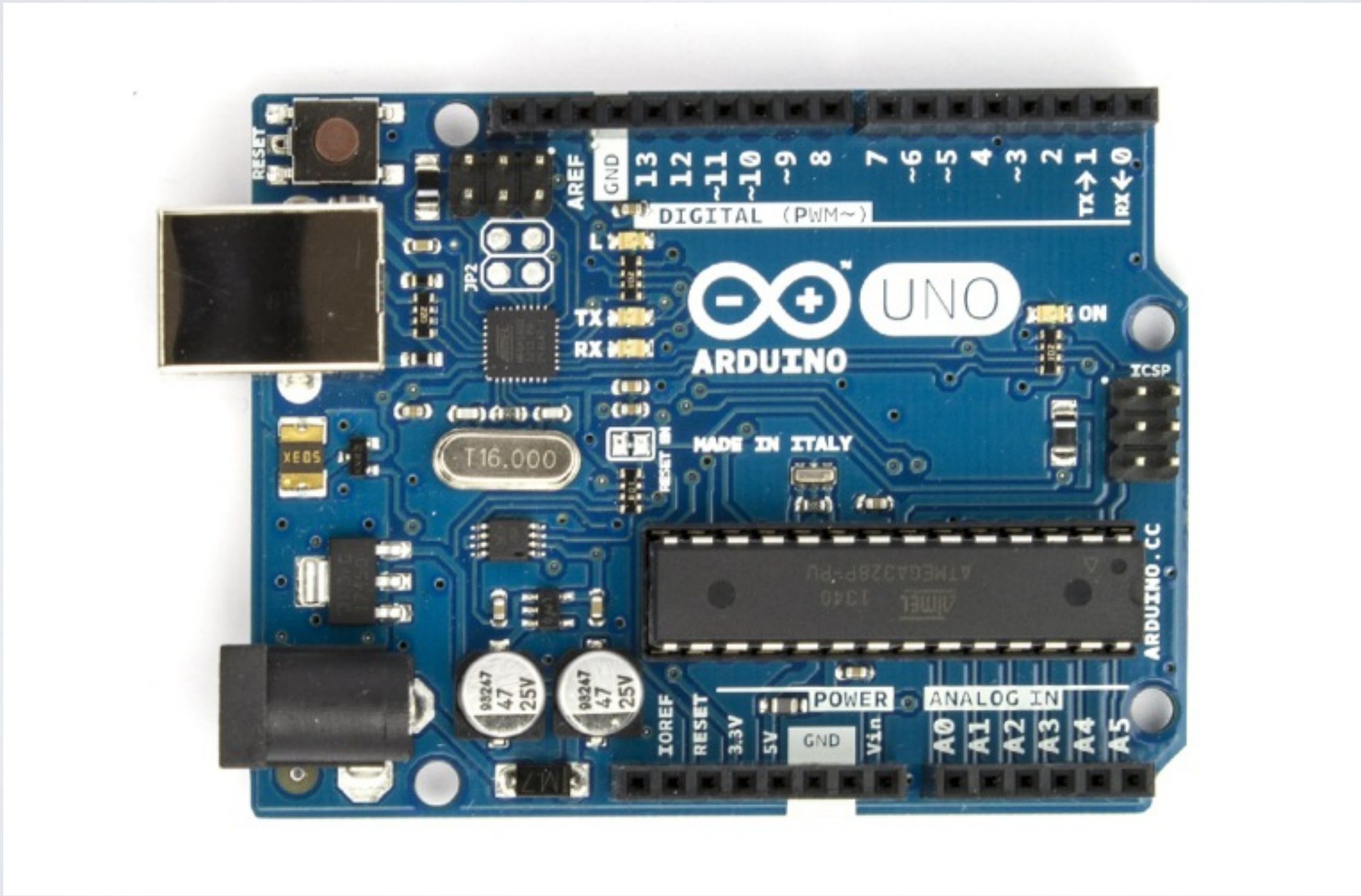


LESSON 1 - GATHER THE PARTS

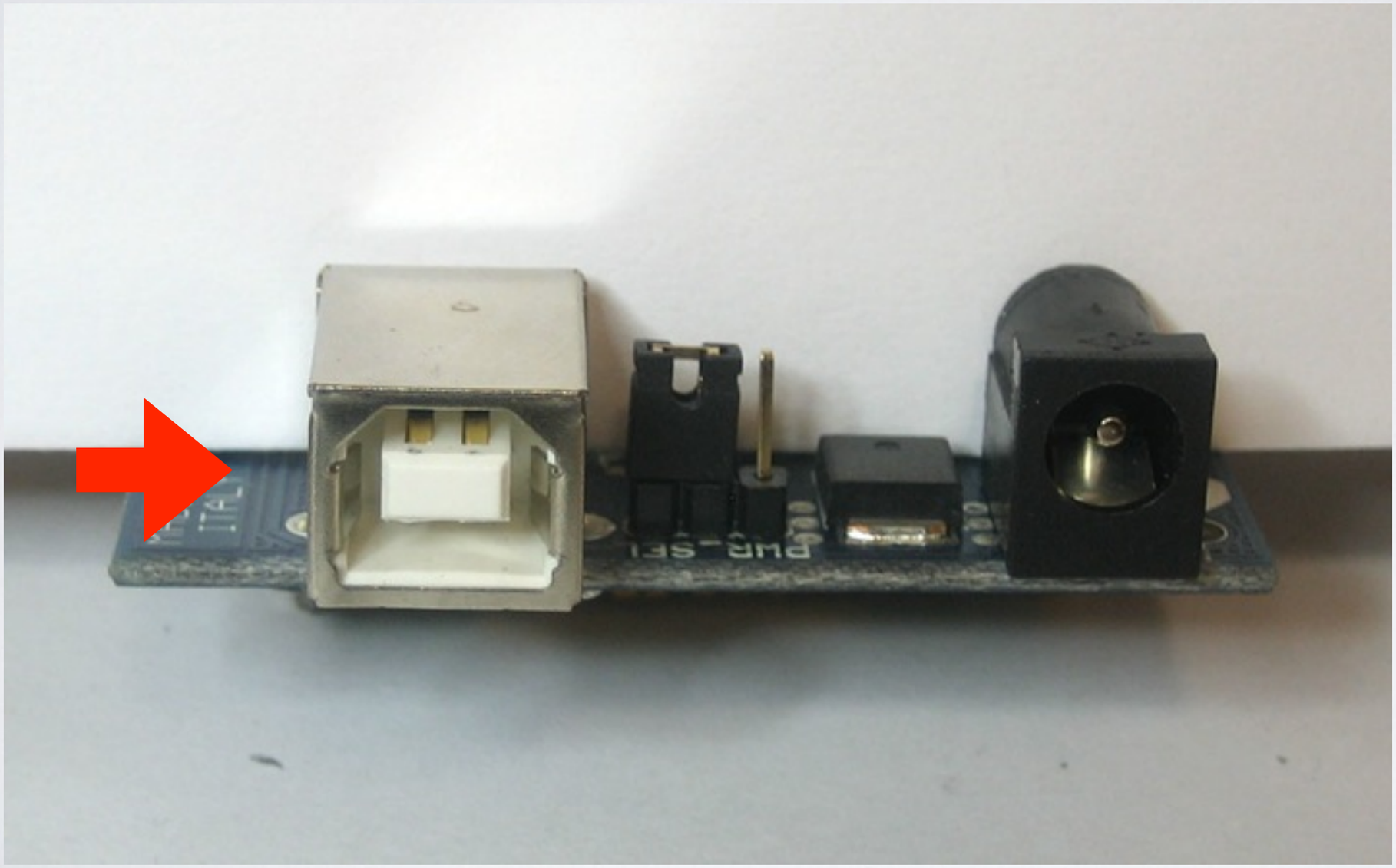
1 USB cable

LESSON 1 - GATHER THE PARTS

1 Arduino



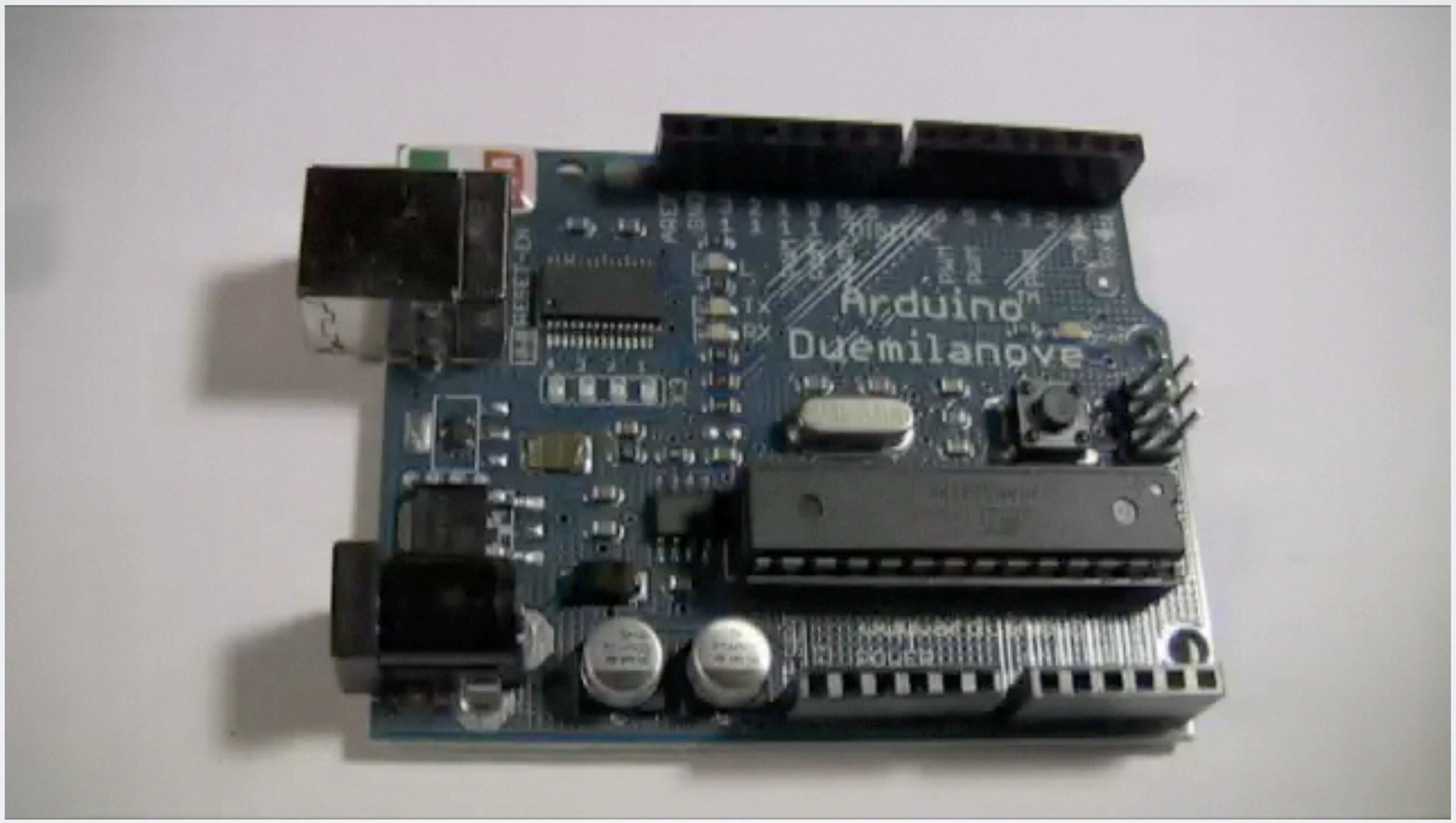
CONNECTING TO YOUR COMPUTER



USB Port

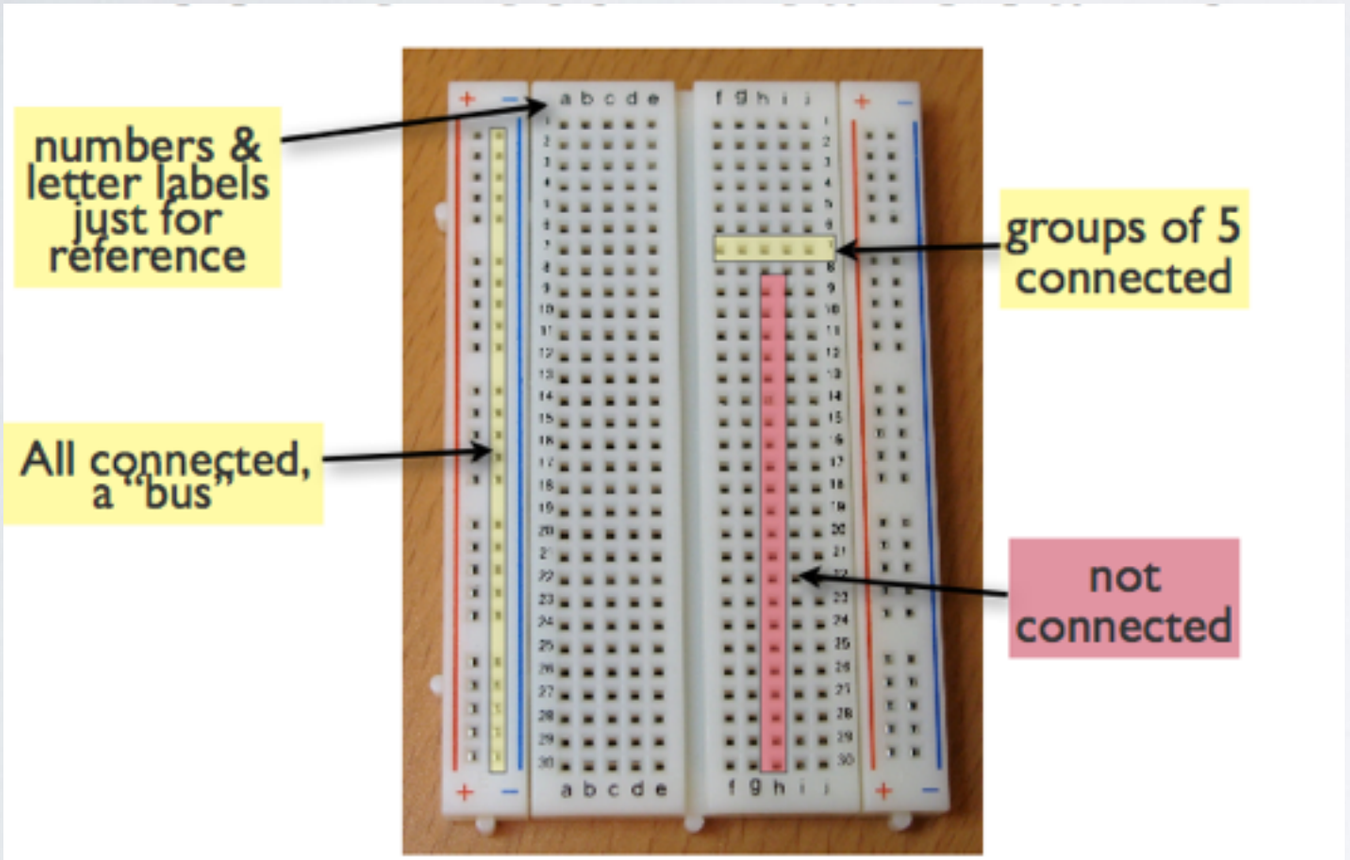
CONNECTING TO YOUR COMPUTER

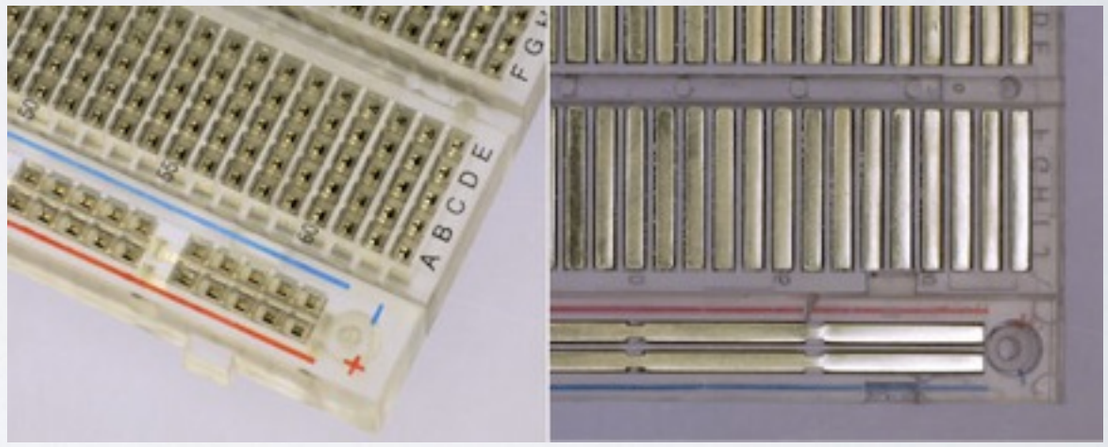
Connect the flat end of the USB cable to your computer, and the square end to the Arduino. A green led on the right should come on.



LET MAKE SOME CIRCUITS!

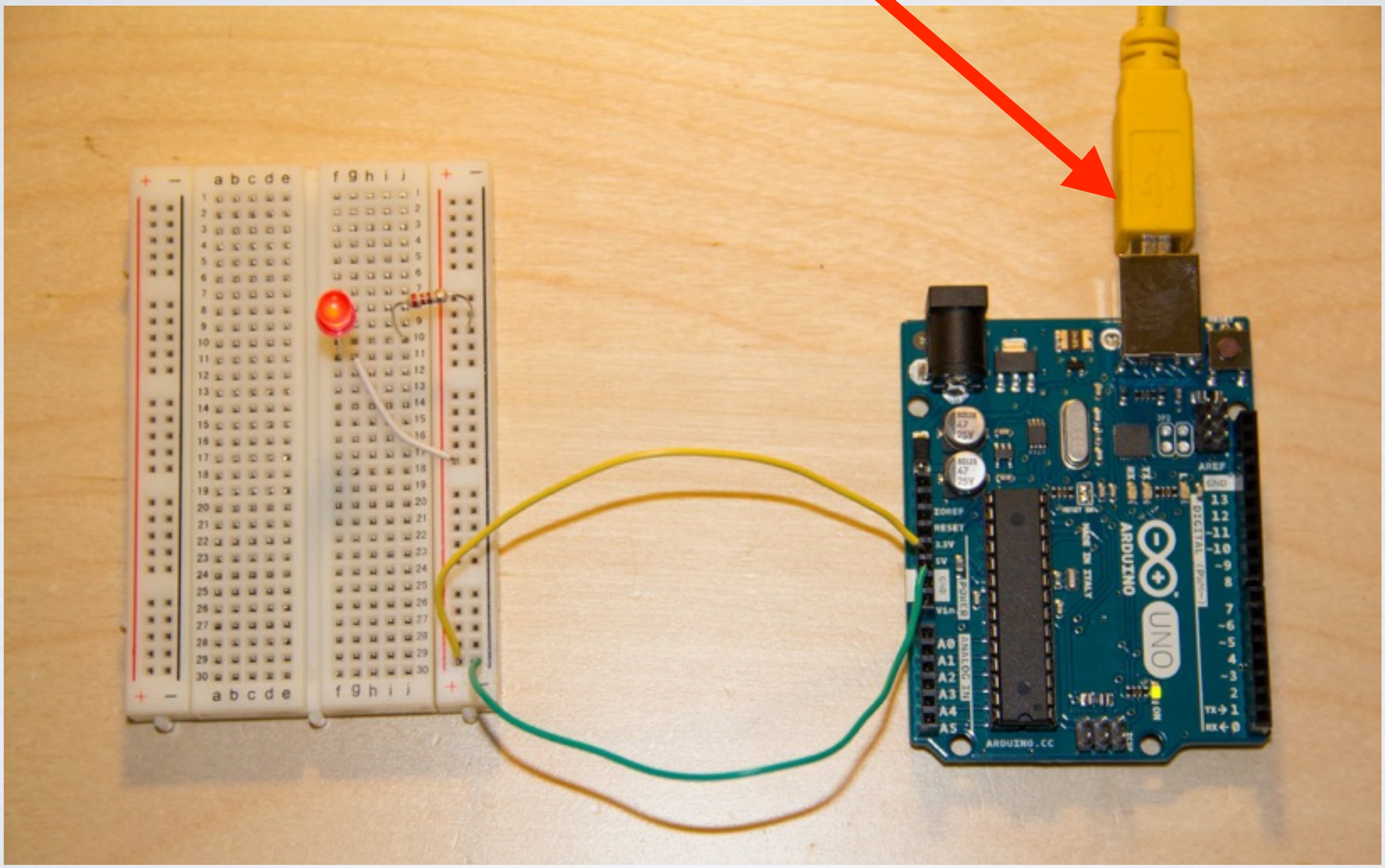
Solderless Breadboard





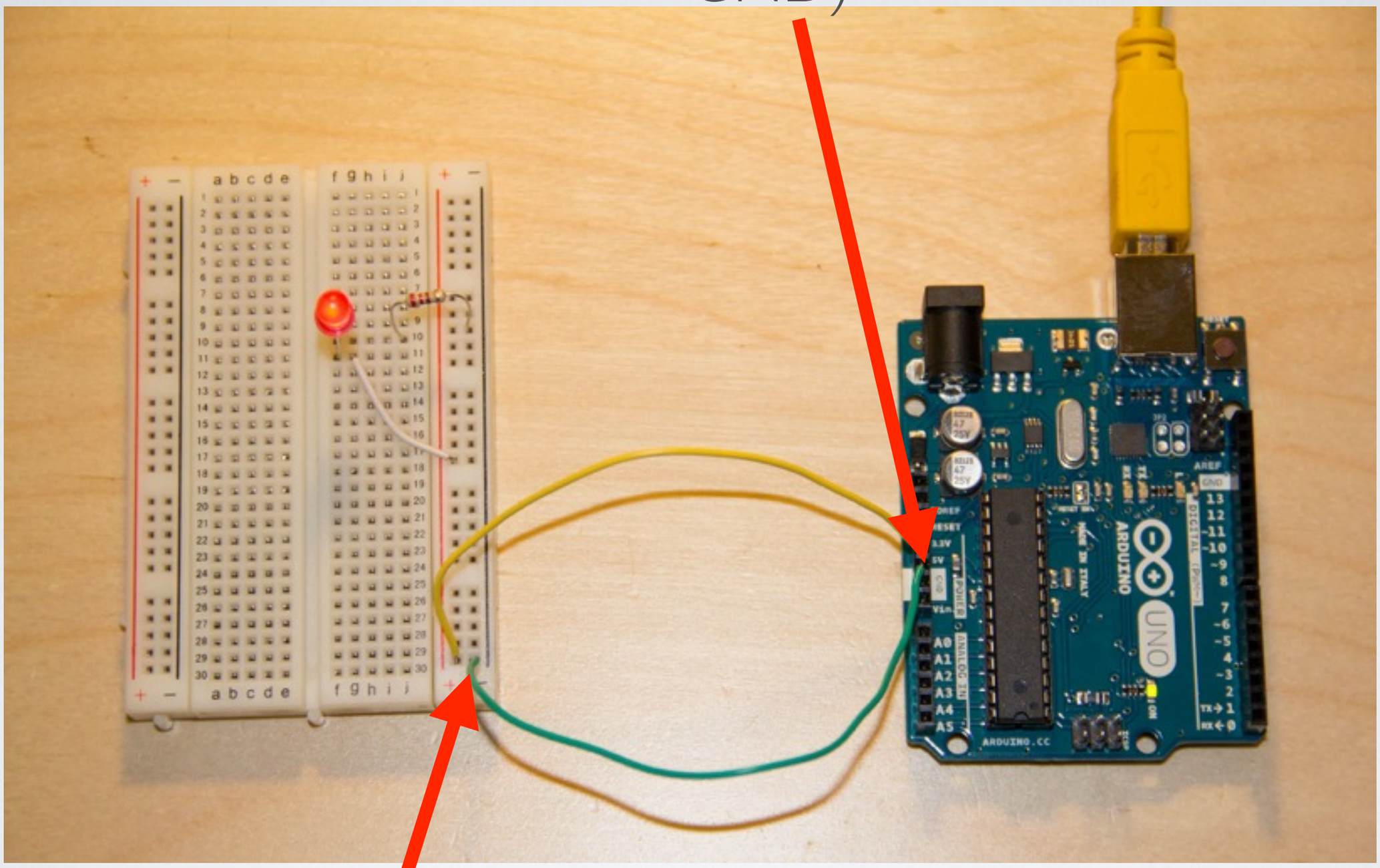
LETS MAKE SOME CIRCUITS!

Step 1 connect Ardunio to computer with USB cable



LETS MAKE SOME CIRCUITS!

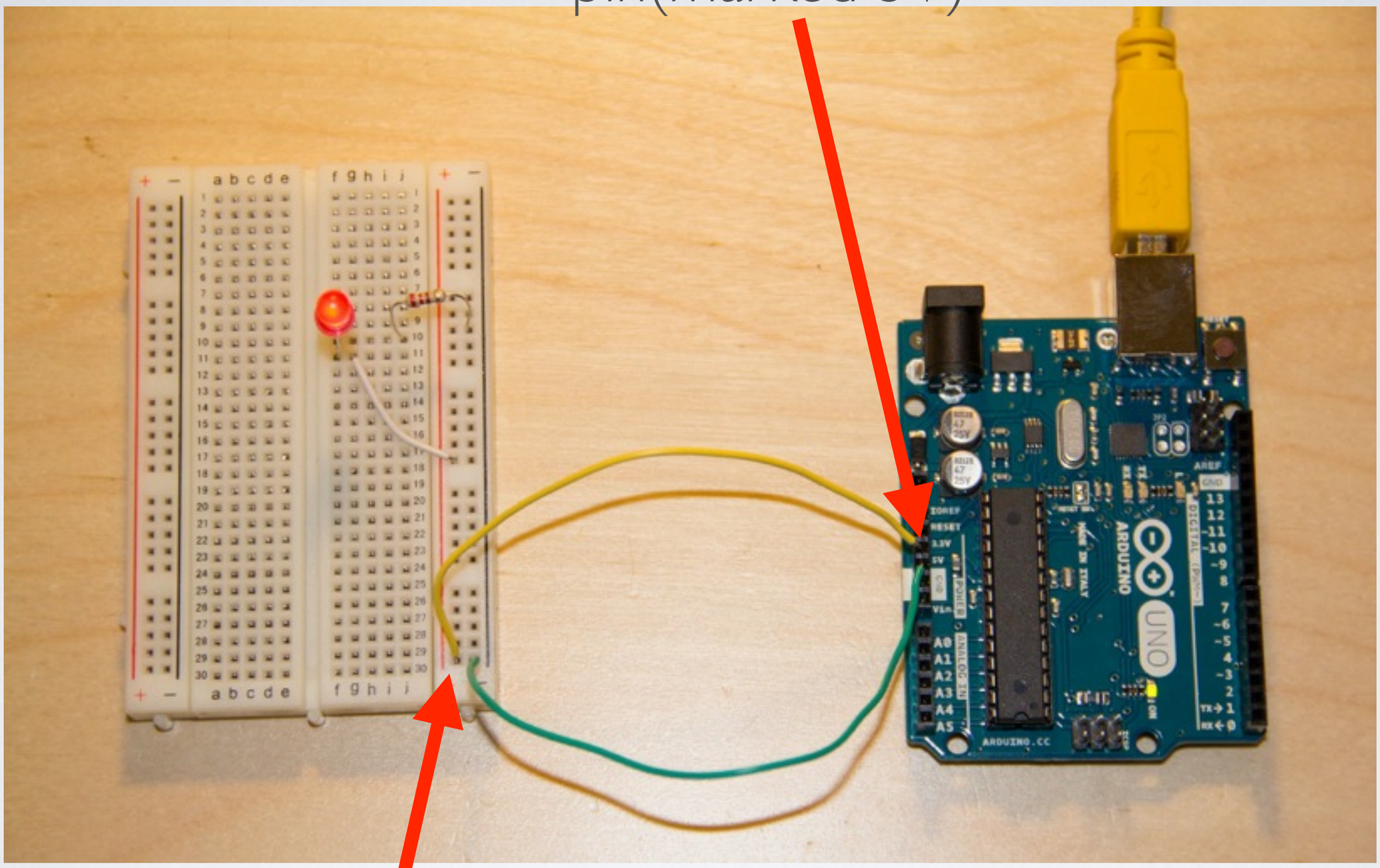
Step 2 connect one side green wire to the ground pin(marked GND)



Connect the other side of the green wire to the ground strip on the breadboard market with a minus sign -

LETS MAKE SOME CIRCUITS!

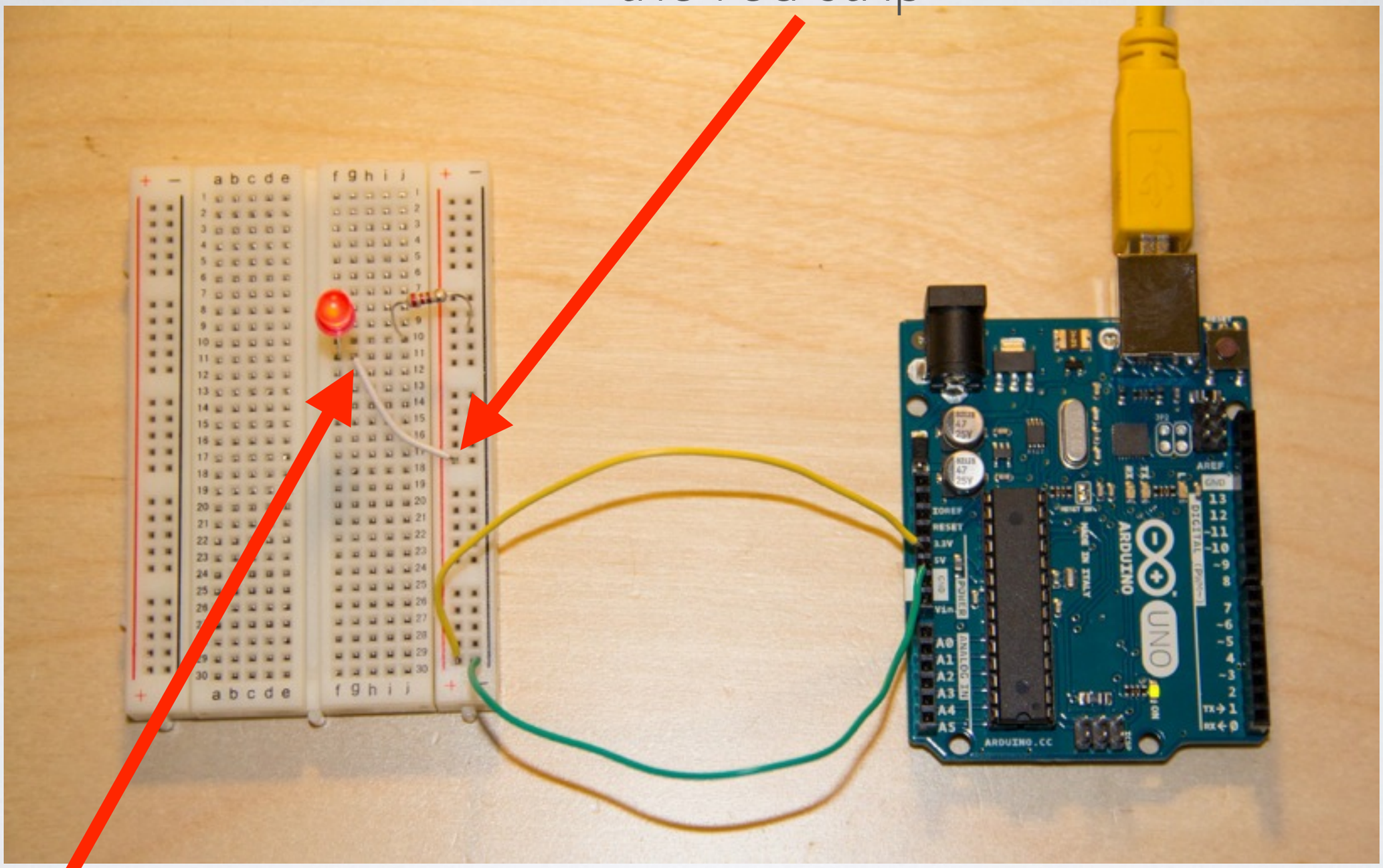
Step 3 connect one side of the yellow wire to the 5 volt pin(marked 5V)



Connect the other side of the green wire to the ground strip on the breadboard market with a plus sign +

LETS MAKE SOME CIRCUITS!

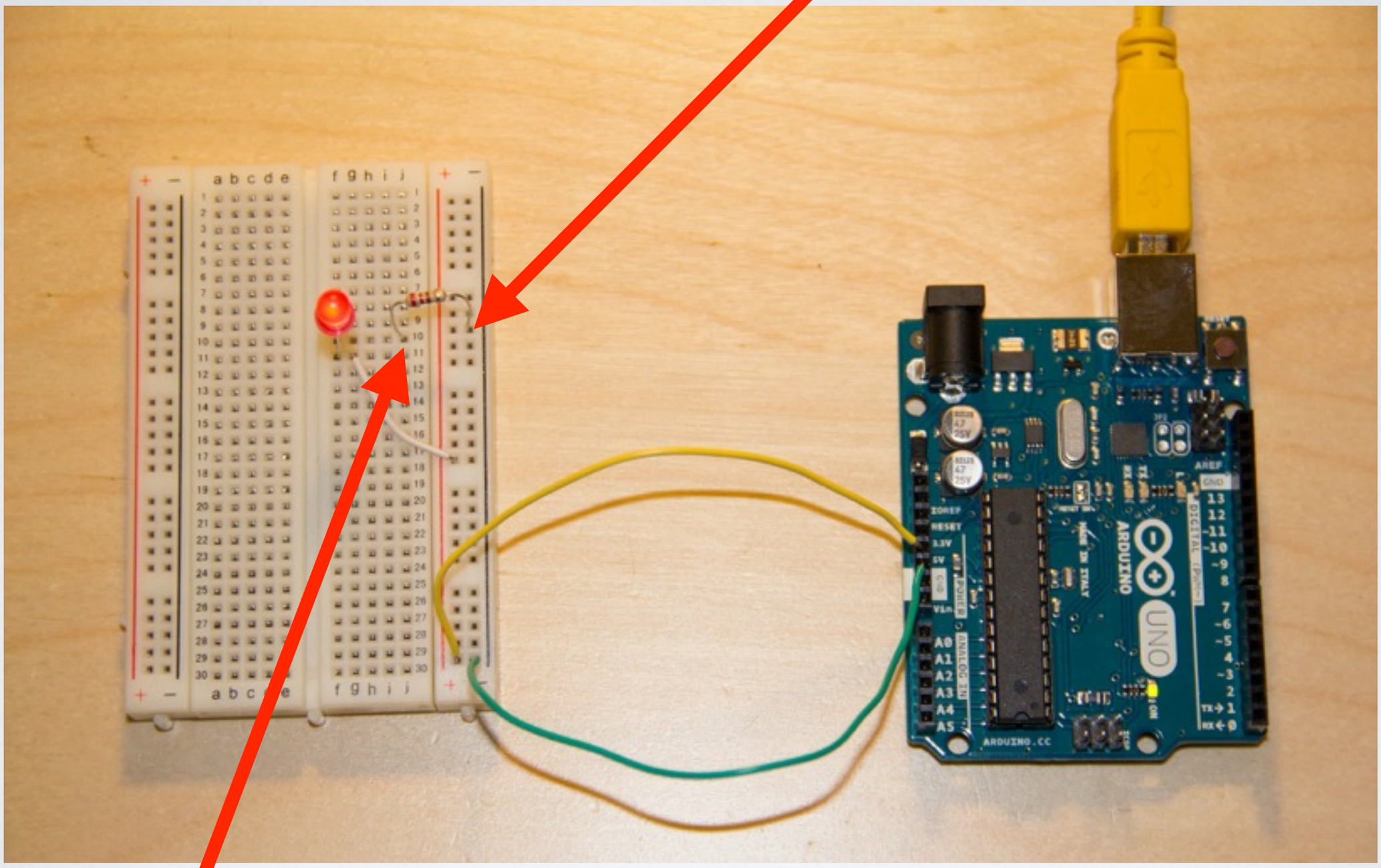
Step 4 connect one side of the short white to any socket along the red strip



Connect the other side of the short wire to any socket in the row numbered 11

LETS MAKE SOME CIRCUITS!

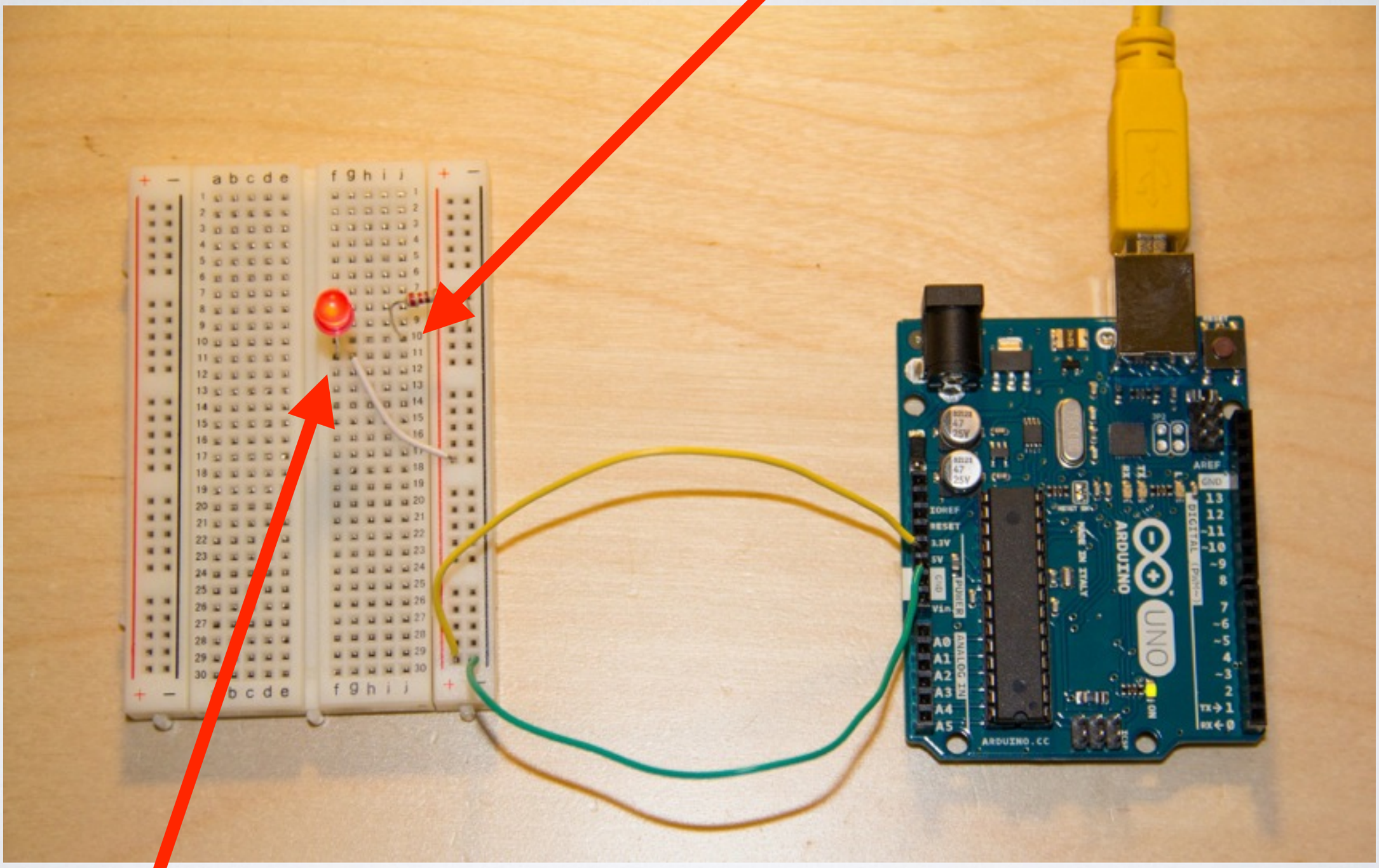
Step 5:Place the resistor on one socket in the strip marked with a negative sign -



Place the other end of the resistor in row 11

LETS MAKE SOME CIRCUITS!

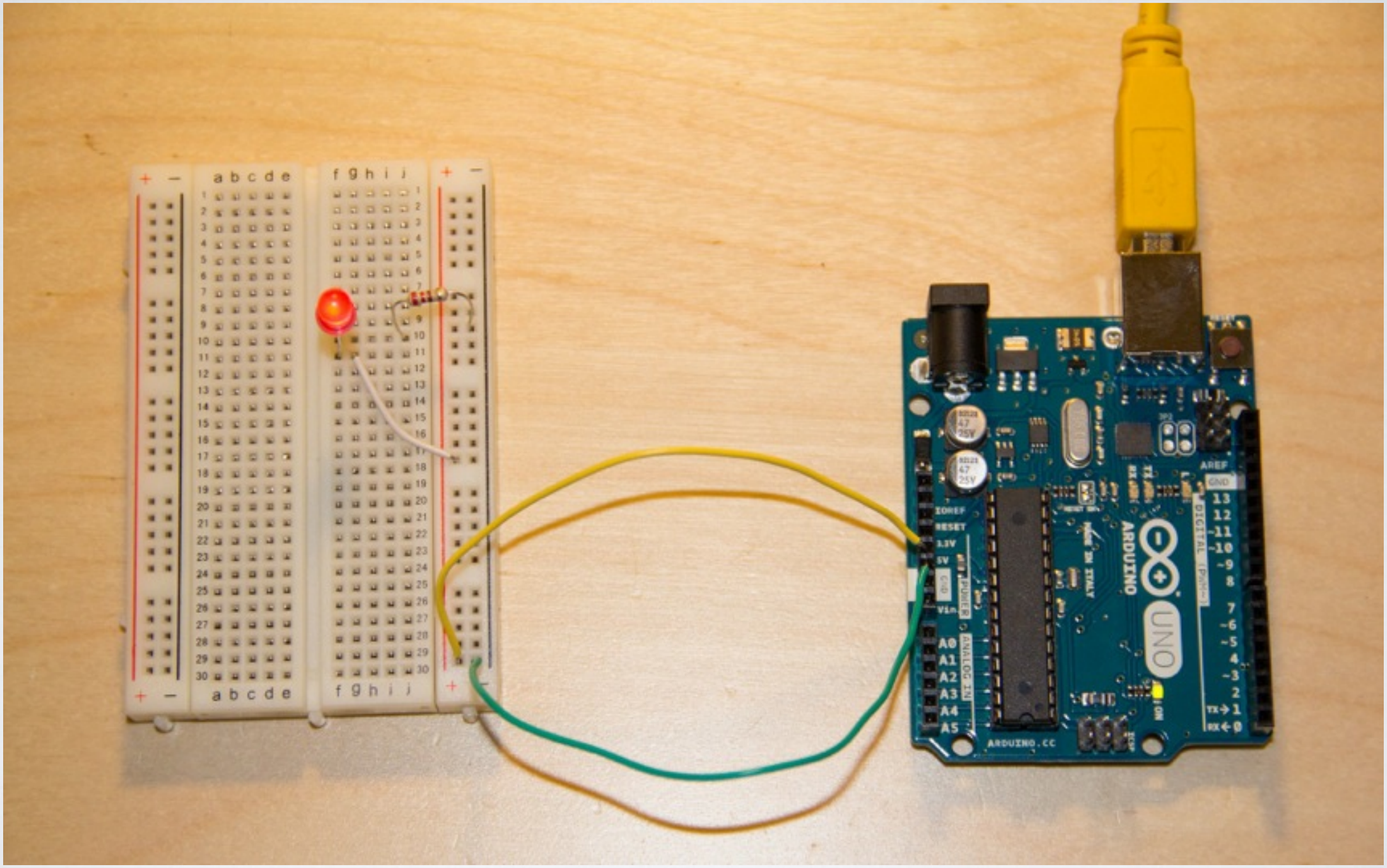
Step 6: The LED has a long wire and a short wire place that in the same row as the short white wire #11



Place short end of the LED in the same row as the resistor #10

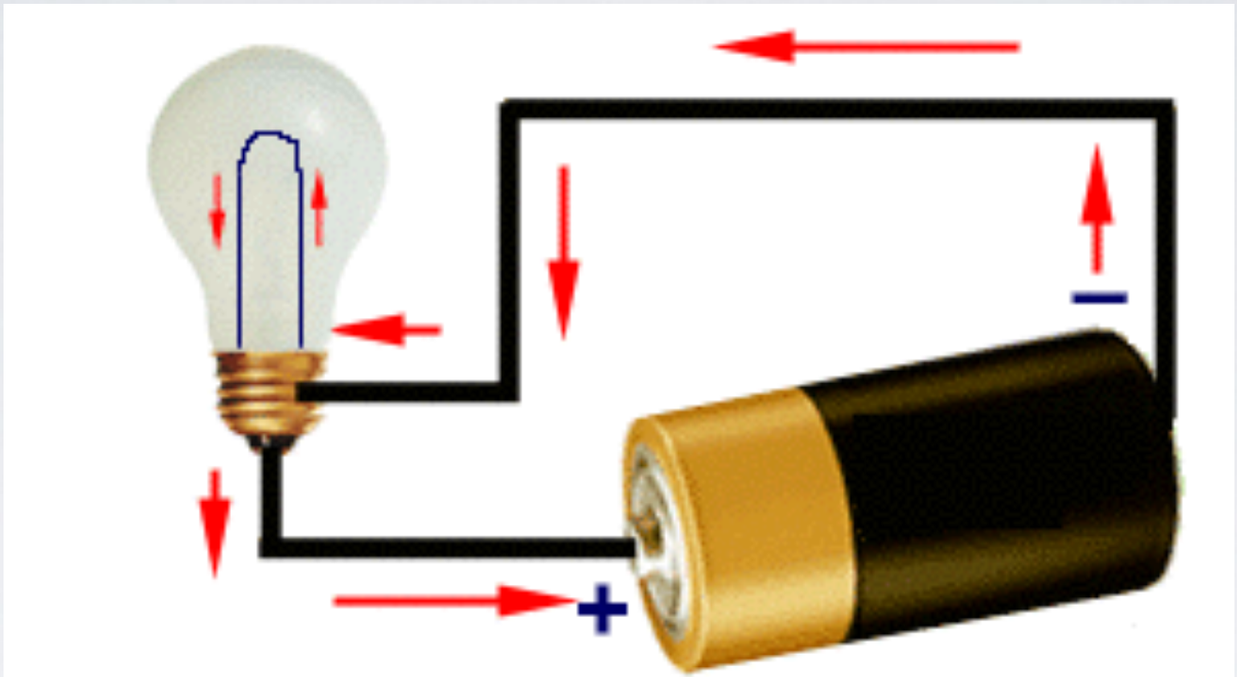
LETS MAKE SOME CIRCUITS!

You should have Light!



LETS MAKE SOME CIRCUITS!

What's going on here?

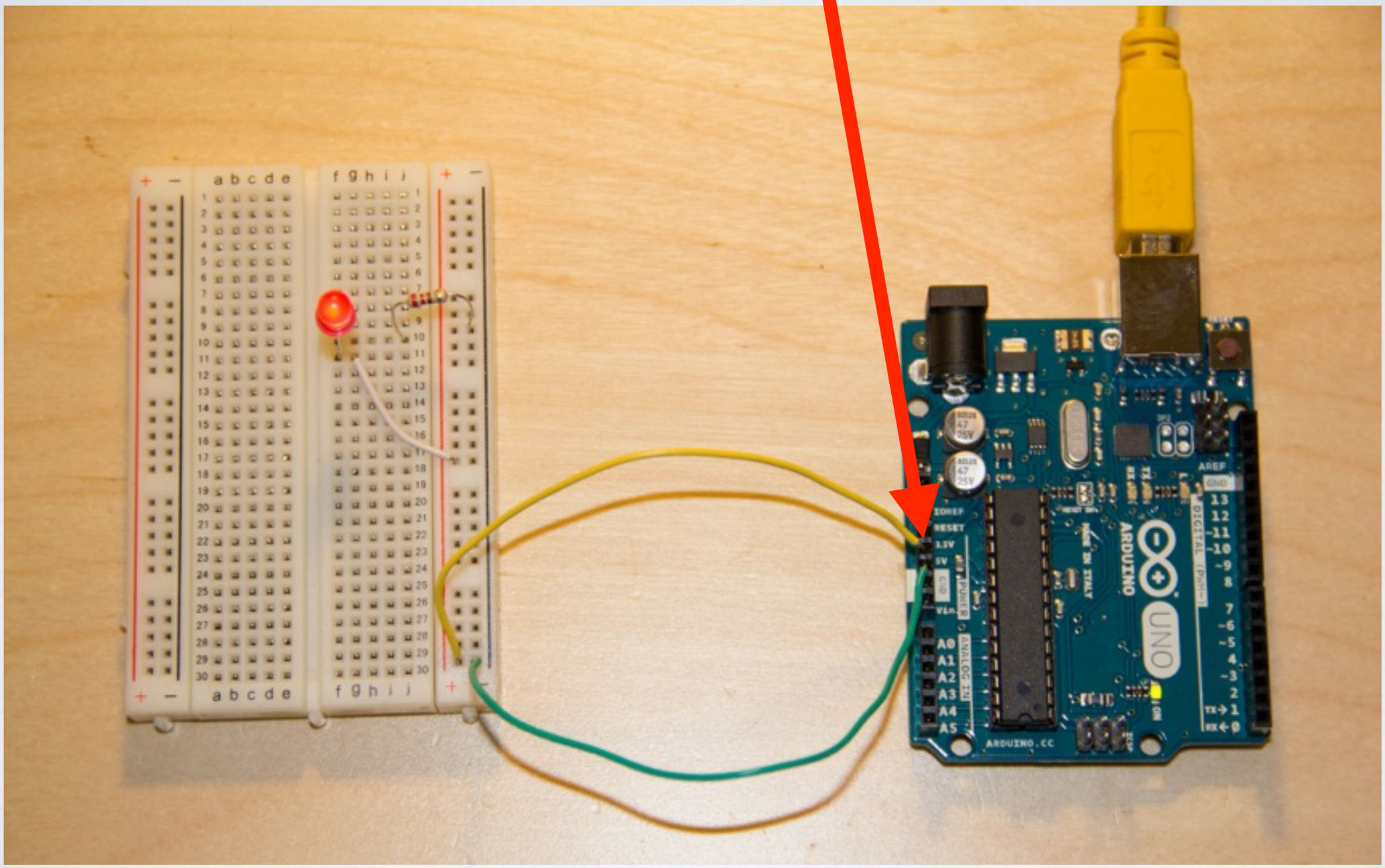


An electric circuit is a closed loop where electric current can flow. Electricity flows like water from Ground to Positive.

As the current passes though a component such as an LED it causes it to light up.

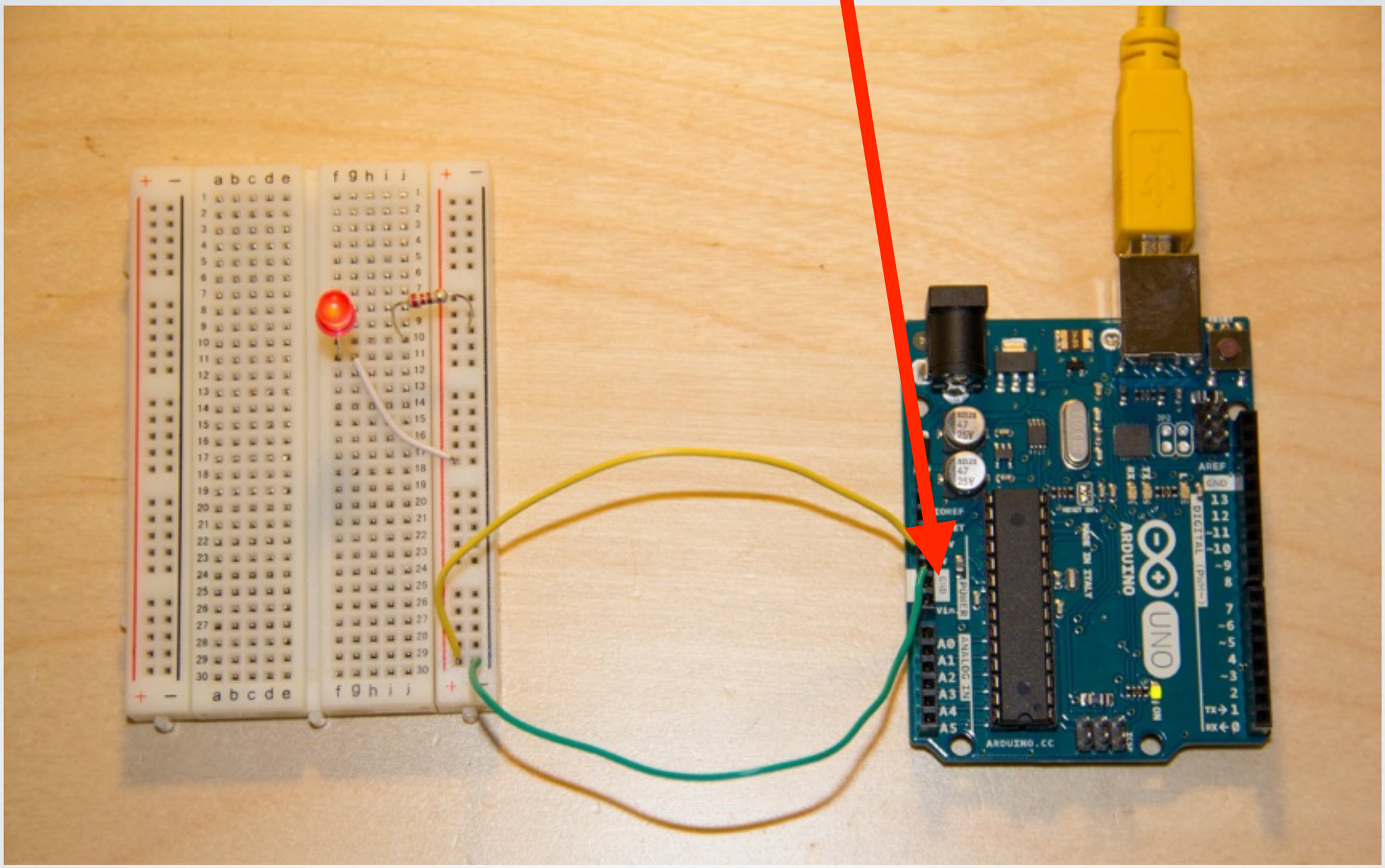
LETS MAKE SOME CIRCUITS!

The Arduino is acting as our battery providing 5 volts of electricity.



LETS MAKE SOME CIRCUITS!

The Arduino is acting as our battery and is closing the loop with the ground connection.



LETS MAKE SOME CIRCUITS!

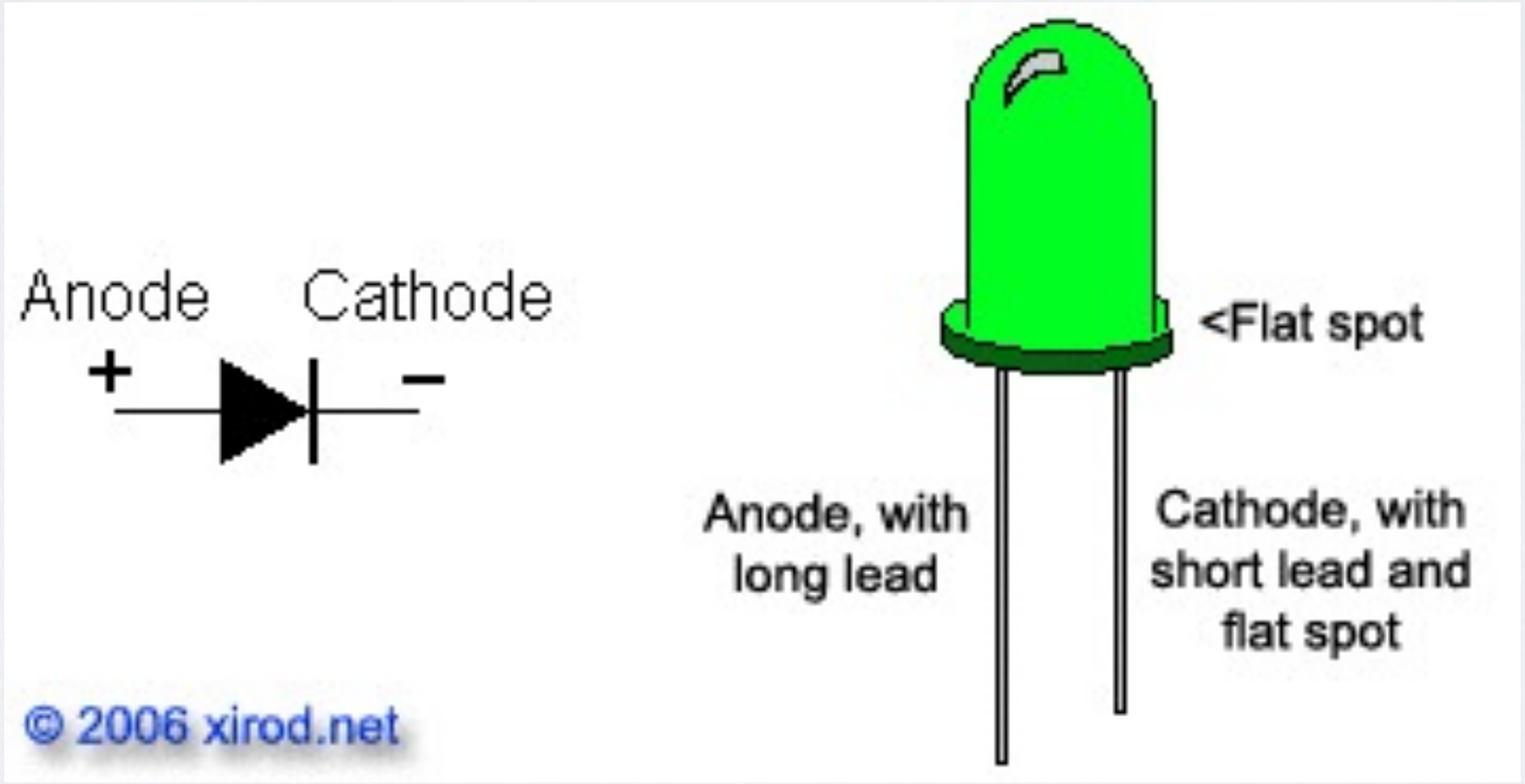
What's going on here?

An led is a directional component.

-The long Lead is called the Anode is connected to the +5Volts

-The short lead is called the Cathode and is connected to ground

-As the electrical current passes through the LED produces light.

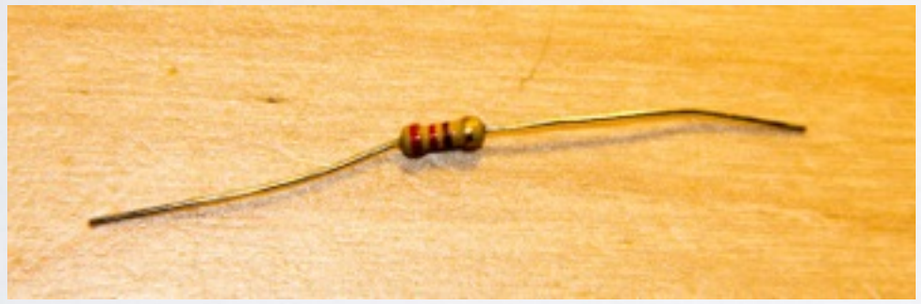


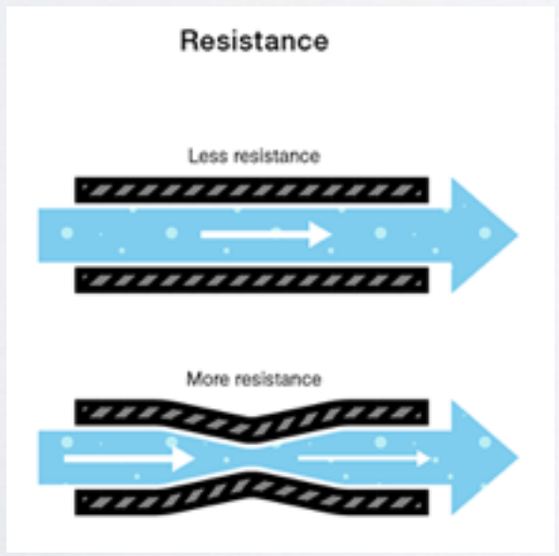
LETS MAKE SOME CIRCUITS!

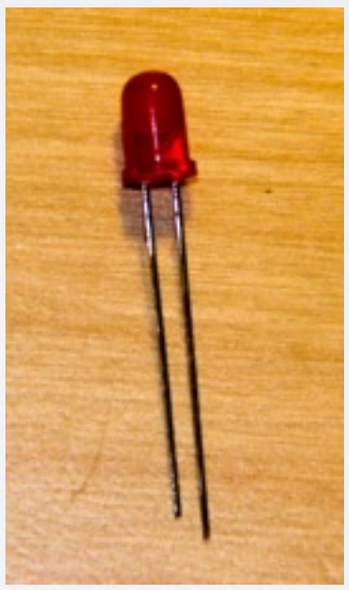
What's going on here?

But, 5 Volts is too much electricity for the LED and will burn it out!

So before we send electricity to it we need to place a resistor in front of it to limit the electrical current.



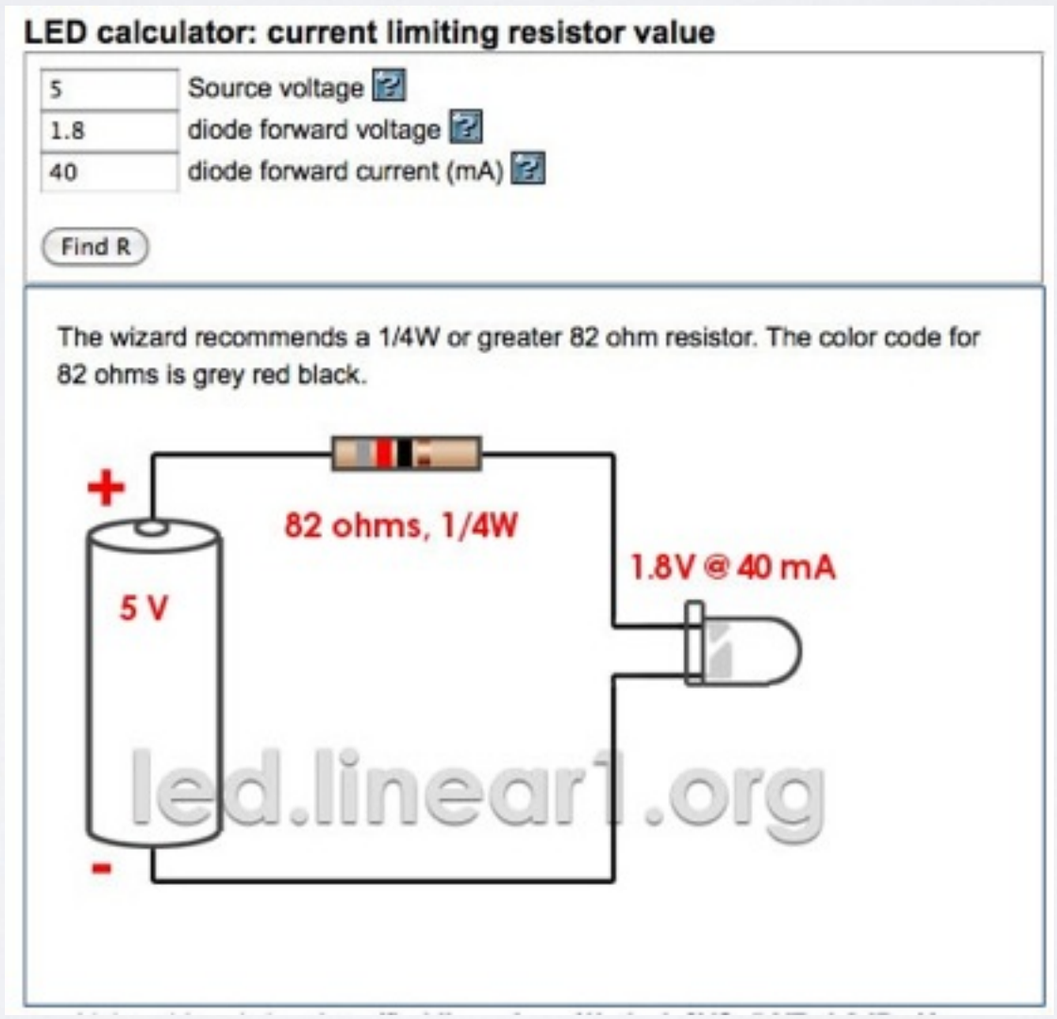




LETS MAKE SOME CIRCUITS!

More on basic electrical theory here:
https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law

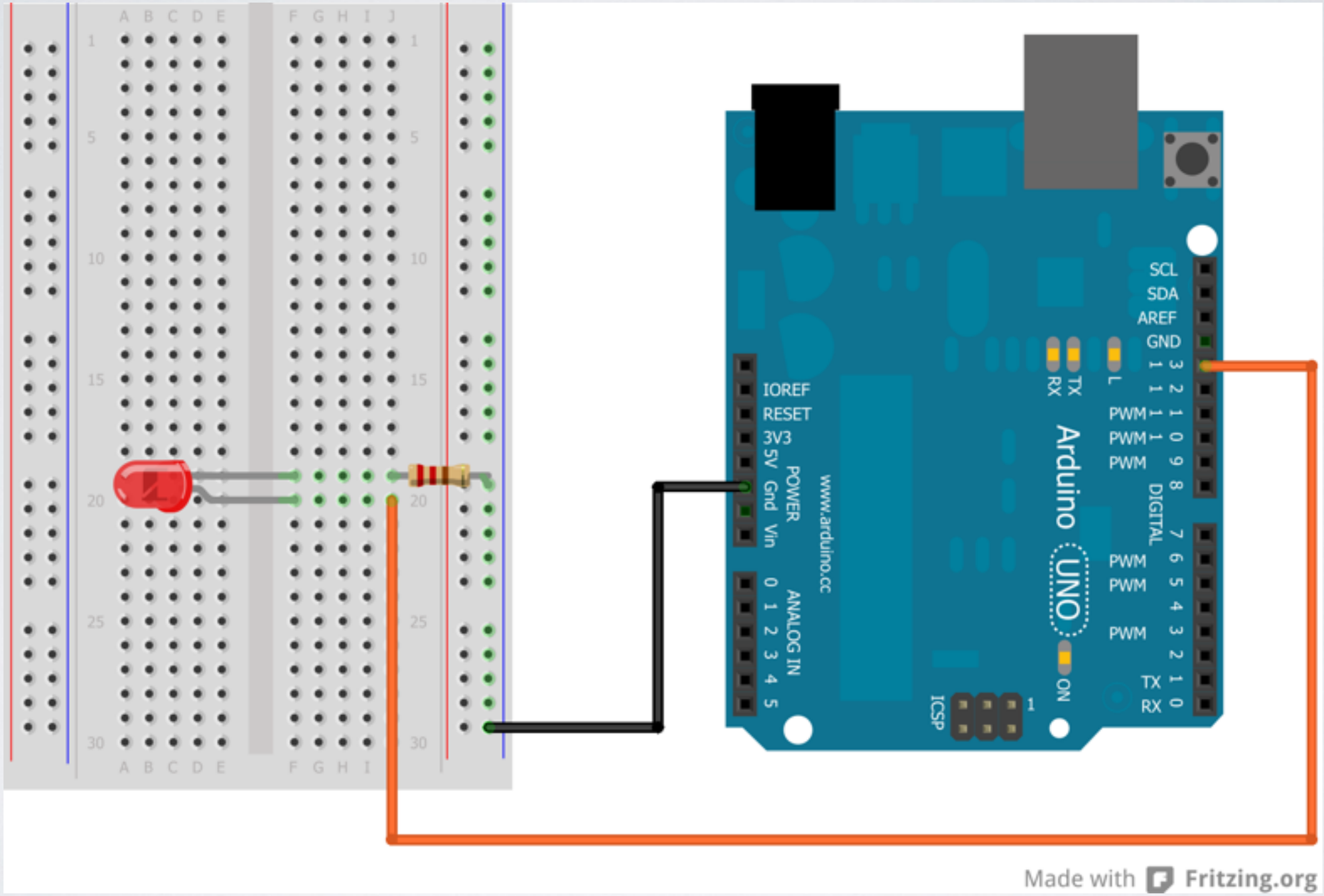
LED Resistor calculator: http://led.linear1.org/1led.wiz



LESSON 2 -REALLY GETTING STARTED WITH ARDUINO

This lesson will basically get you up and running using the Arduino software and uploading a sketch to the Arduino board. Once you've completed this step we can continue to the really interesting stuff, which is when we start writing our own sketches!

BLINKING AN LED



BLINKING AN LED

Parts:

1 long green wire

1 long yellow wire

1 Led(your choice of color)

1 resistor (bands are red,red,violet, gold)

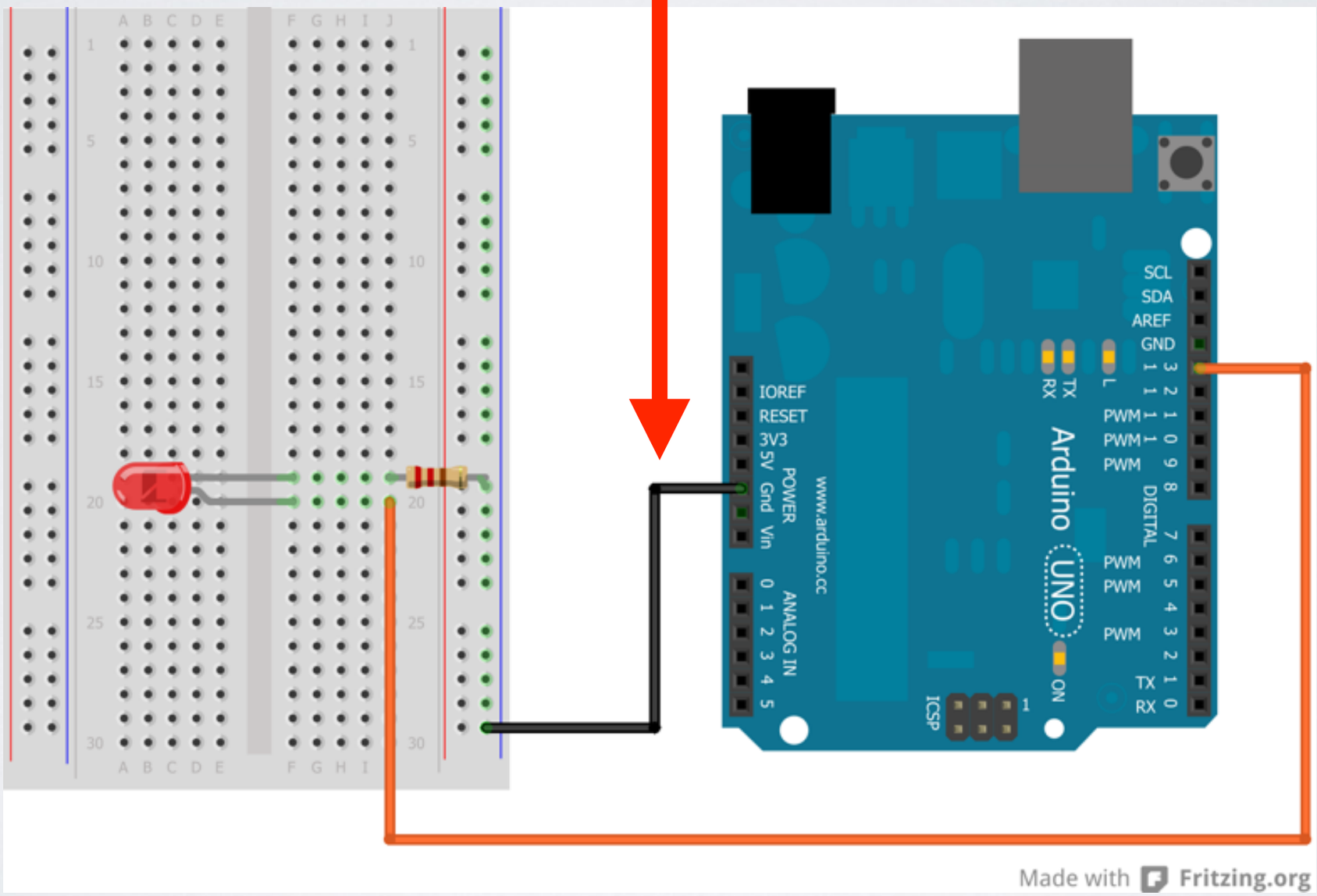
1 Breadboard

1 Arduino and USB cable

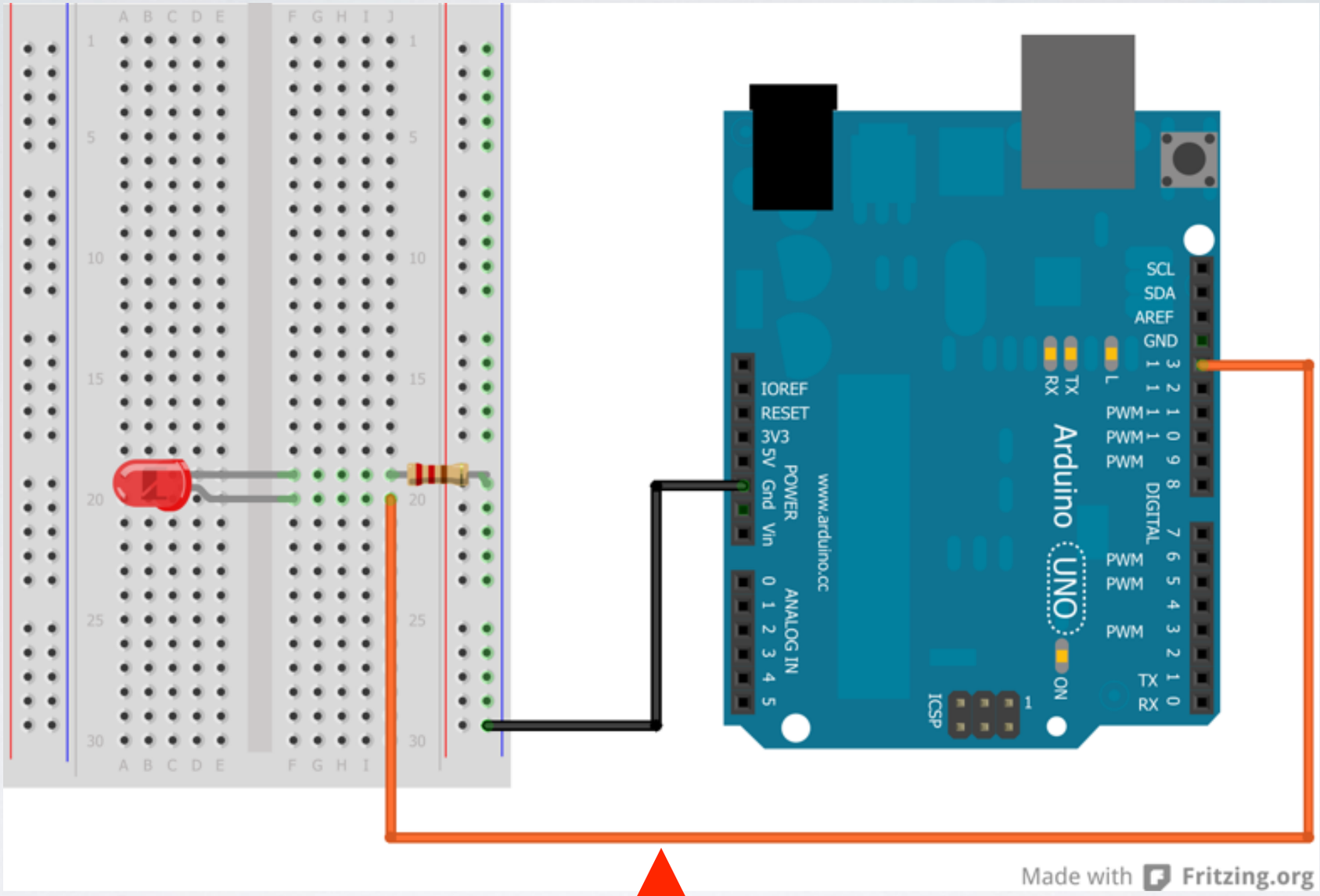
*Most of the parts from exercise 1

BLINKING AN LED

1 Connect GND on the arduino
to the ground terminal on the breadboard

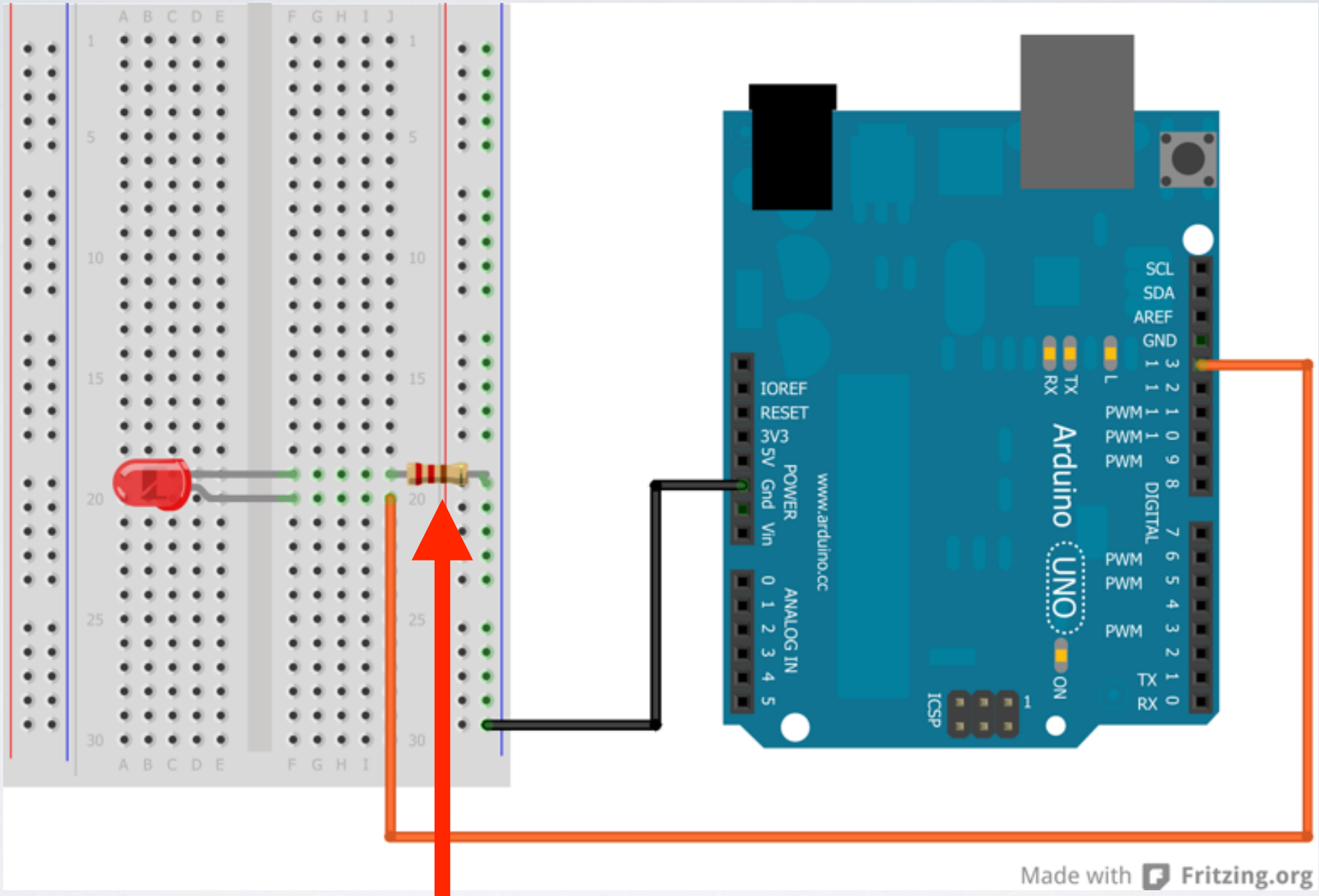


BLINKING AN LED



2 Connect **Pin 13** on the to to the same row as the long lead of the LED on the breadboard

BLINKING AN LED

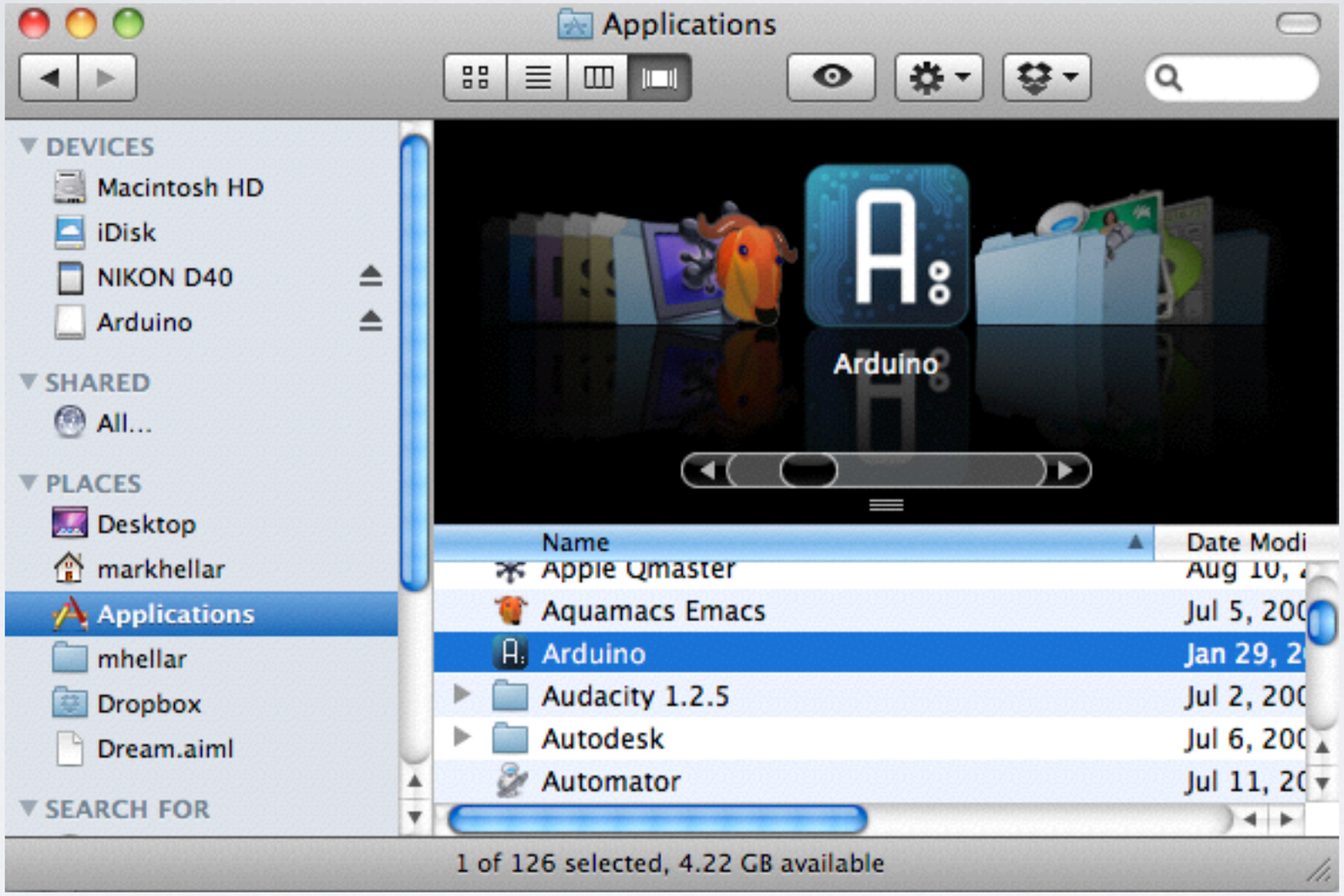


3 Make a connection between the ground strip and the short lead of the LED using the resistor.

BLINKING AN LED

Once the LED is connected, you need to tell Arduino what to do. This is done through code - that is, a list of instructions that we give the micro- controller to make it do what we want.

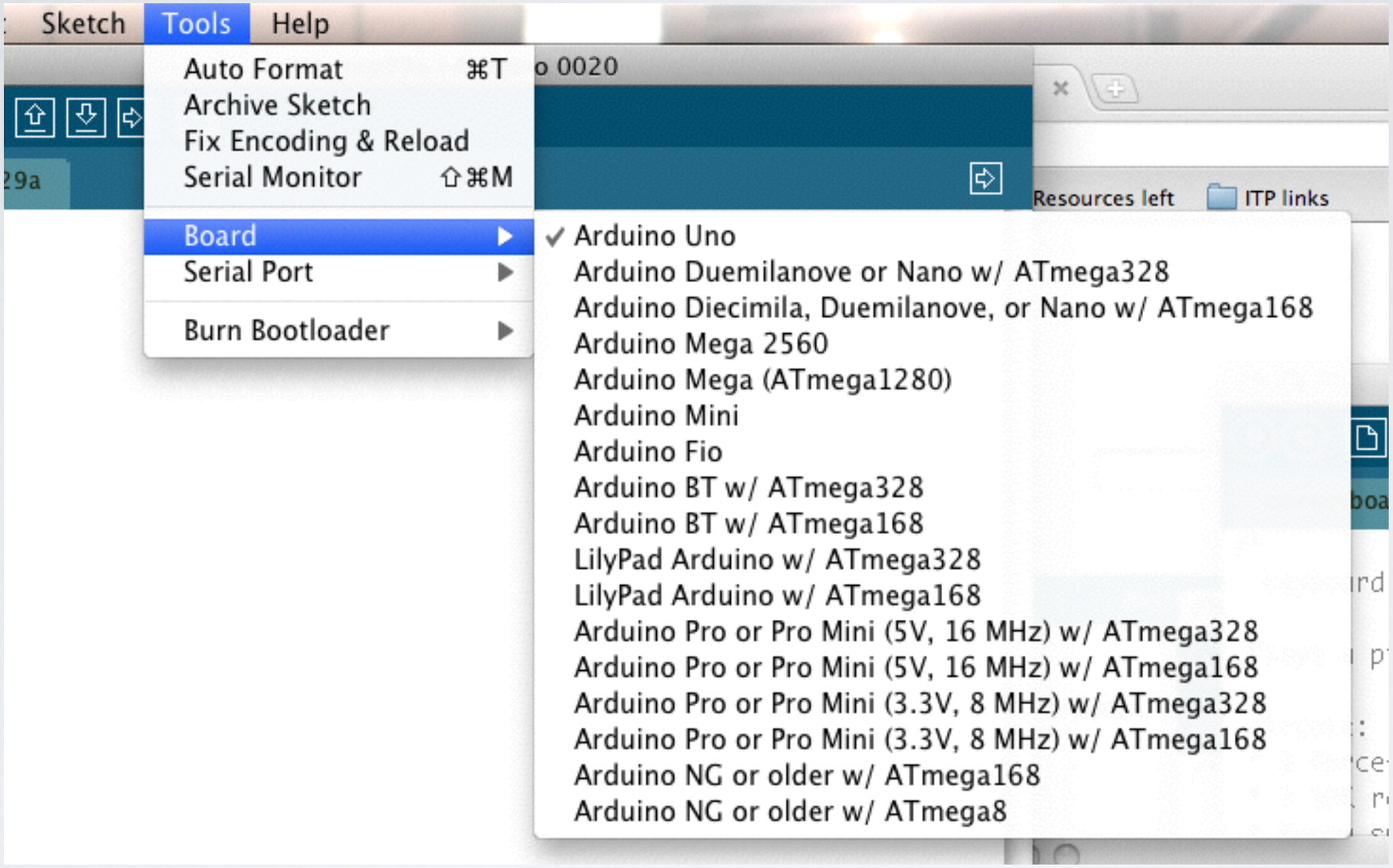
BLINKING AN LED



Fire up the Arduino IDE from the Applications folder.

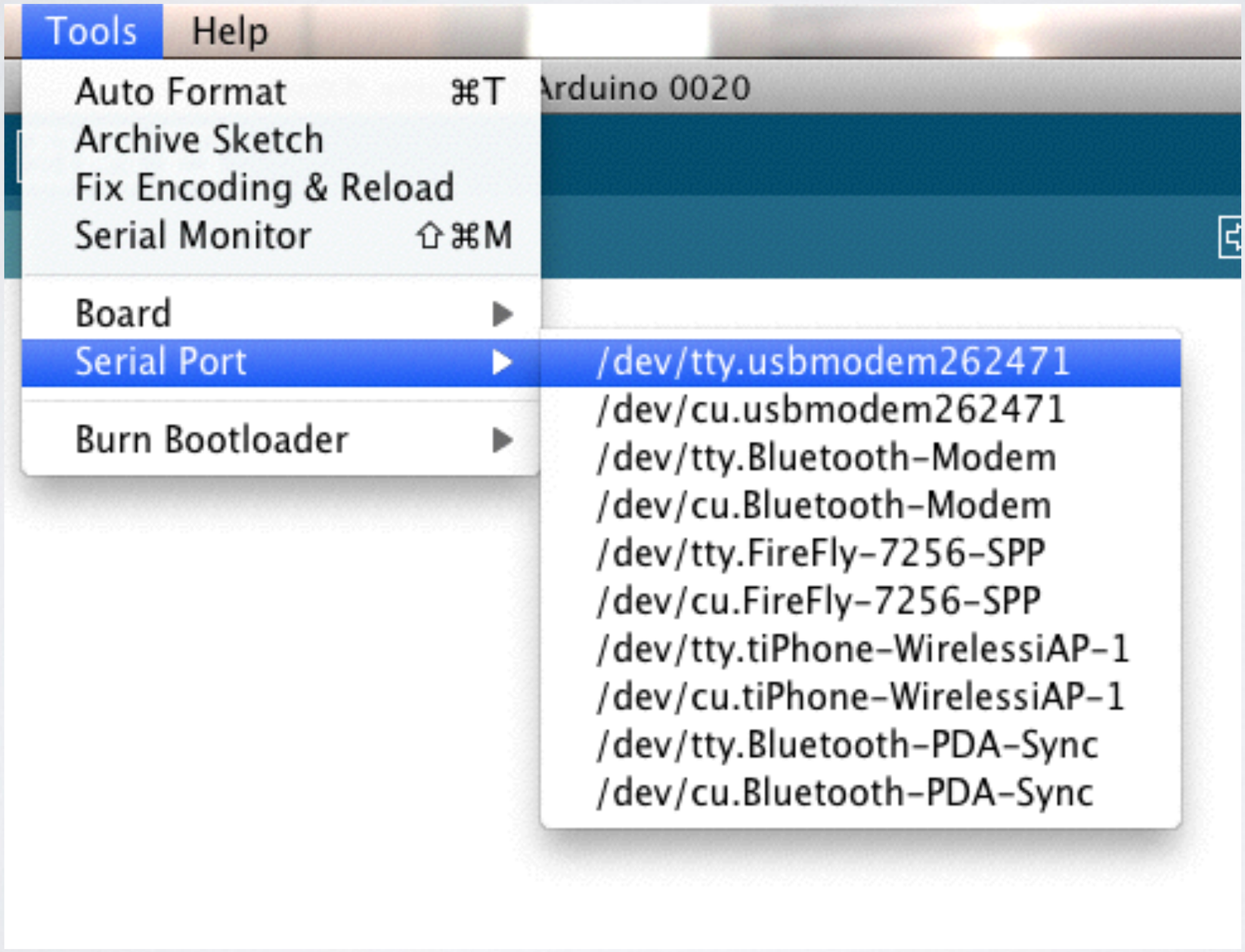
BLINKING AN LED

From the tools menu -> Board->Arduino Uno



BLINKING AN LED

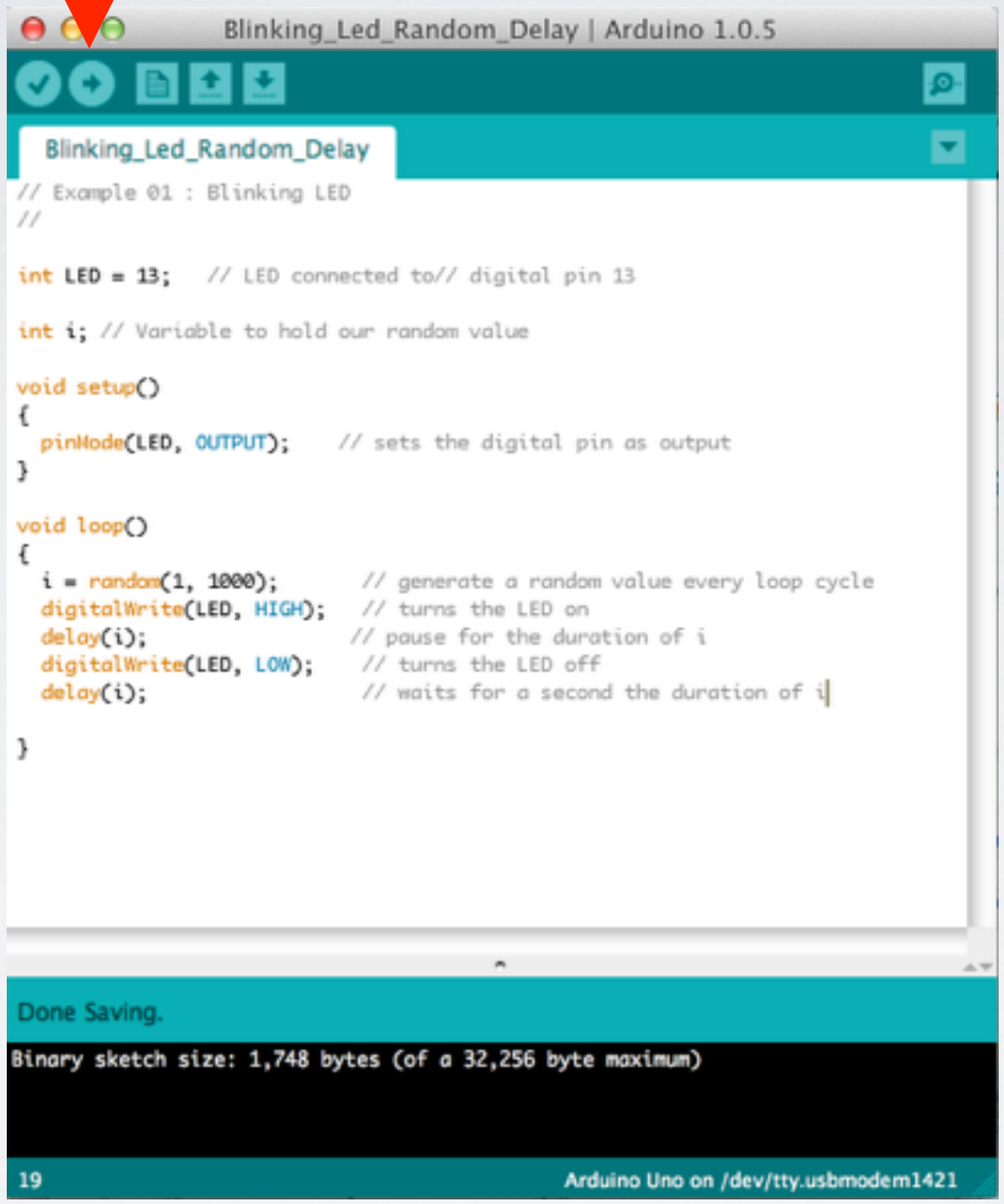
From the tools menu -> Serial Port >tty.usbmodem####



BLINKING AN LED

Upload code to board

Check code for errors



The Arduino IDE with your first sketch loaded

BLINKING AN LED

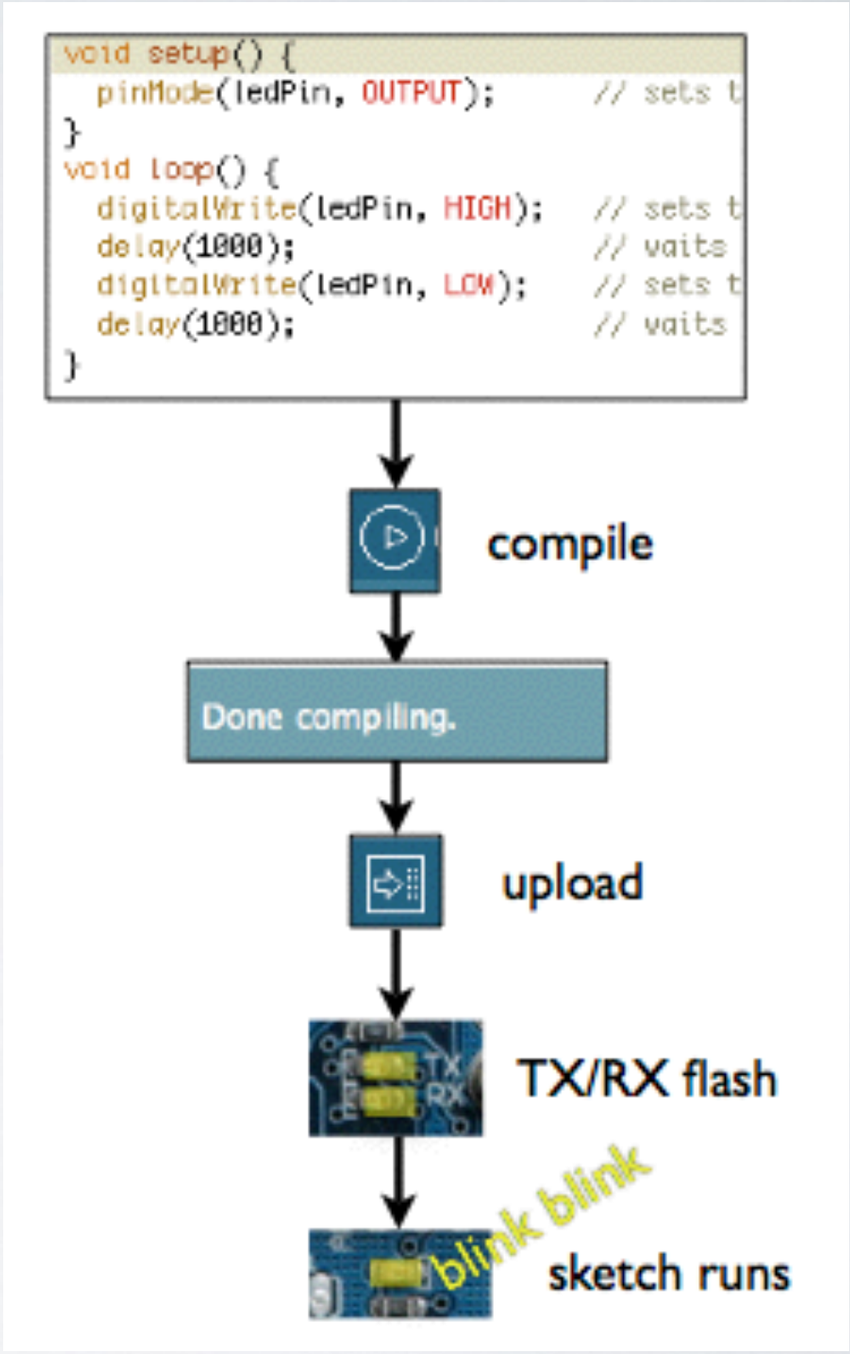
```
// Example 01 : Blinking LED

int LED = 13;   // LED connected to digital pin 13

void setup()
{

pinMode(LED, OUTPUT);    // sets the digital
                         // pin as output
}

void loop()
{
  digitalWrite(LED, HIGH);   // turns the LED on
  delay(1000);               // waits for a second
  digitalWrite(LED, LOW);    // turns the LED off
  delay(1000);               // waits for a second
}
```

USING ARDUINO

-Write your sketch

-Press Compile button (to check for errors)

-Press Upload button to program Arduino board with your sketch



CODE:STEP BY STEP

Arduino expects two functions to exists—one called **setup()** and one called **loop()**.

setup() is where you put all the code that you want to execute once at the beginning of your program

loop() contains the core of your program, which is executed over and over again.

When you power up the board, the code runs; when you want to stop, you just turn it off.

CODE:STEP BY STEP

// Example 01 : Blinking LED

A comment is a useful way for us to write little notes. The preceding title comment just reminds us that this program, Example 01, blinks an LED.

CODE:STEP BY STEP

int LED = 13; // LED connected to // digital pin 13

We are defining a integer variable called LED as Arduino pin 13.

CODE:STEP BY STEP

void setup()

This line tells Arduino that the next function will be called setup().

CODE:STEP BY STEP

```
void setup()
{
pinMode(LED, OUTPUT); // sets the digital // pin as output
}
```

pinMode tells Arduino how to configure a certain pin. Digital pins can be used either as INPUT or OUTPUT.

In this case, we need an output pin to control our LED, so we place the number of the pin and its mode inside the parentheses.

pinMode is a function, and the words (or numbers) specified inside the parentheses are arguments. INPUT and OUTPUT are constants in the Arduino language.

CODE:STEP BY STEP

```
void loop()
 {
```

loop() is where you specify the main behavior of your interactive device. It will be repeated over and over again until you switch the board off.

CODE:STEP BY STEP

digitalWrite(LED, HIGH); // turns the LED on

As the comment says, digitalWrite() is able to turn on (or off) any pin that has been configured as an OUTPUT.

The first argument (in this case, LED) specifies which pin should be turned on or off (remember that LED is a constant value that refers to pin 13, so this is the pin that's switched).

The second argument can turn the pin on (HIGH) or off (LOW).

CODE:STEP BY STEP

Imagine that every output pin is a tiny power socket, like the ones you have on the walls of your apartment. American ones are 110 V, and Arduino works at a more modest 5 V.

The magic here is when software becomes hardware. When you write digitalWrite(LED, HIGH), it turns the output pin to 5 V, and if you connect an LED, it will light up.

So at this point in your code, an instruction in software makes something happen in the physical world by controlling the flow of electricity to the pin. Turning on and off the pin at will now let us translate these into something more visible for a human being; the LED is our actuator.

CODE:STEP BY STEP

delay(1000); // wait for a second

Arduino has a very basic structure. Therefore, if you want things to happen with a certain regularity, you tell it to sit quietly and do nothing until it is time to go to the next step. delay() basically makes the processor sit there and do nothing for the amount of milliseconds that you pass as an argument. Milliseconds are thousands of seconds; therefore, 1000 milliseconds equals 1 second. So the LED stays on for one second here.

CODE:STEP BY STEP

digitalWrite(LED, LOW); // turns the LED off

This instruction now turns off the LED that we previously turned on. Why do we use HIGH and LOW? Well, it's an old convention in digital electronics. HIGH means that the pin is on, and in the case of Arduino, it will be set at 5 V. LOW means 0 V. You can also replace these arguments mentally with ON and OFF.

CODE:STEP BY STEP

delay(1000); // wait for a second

Here, we delay for another second. The LED will be off for one second.

}

This closing curly bracket marks end of the loop function.

CODE:STEP BY STEP

To sum up, this program does this:

- Turns pin 13 into an output (just once at the beginning)

- Enters a loop

- Switches on the LED connected to pin 13

- Waits for a second

- Switches off the LED connected to pin 13

- Waits for a second

- Goes back to beginning of the loop

CODE:STEP BY STEP

Before we move on to the next section, I want you to play with the code. For example, reduce the amount of delay, using different numbers for the on and off pulses so that you can see different blinking patterns.

In particular, you should see what happens when you make the delays very small, but use different delays for on and off . . . there is a moment when something strange happens; this "something" will be very useful when you learn about pulse-width modulation on Thursday.

RANDOM DELAY

```
//Random Delay
//

int LED = 13;   // LED connected to// digital pin 13

int i; // Variable to hold our random value

void setup()
{
  pinMode(LED, OUTPUT);    // sets the digital pin as
output
}

void loop()
{
  i = random(1, 1000);       // generate a random value every
loop cycle
  digitalWrite(LED, HIGH);   // turns the LED on
  delay(i);                  // pause for the duration of variable i
  digitalWrite(LED, LOW);    // turns the LED off
  delay(i);                  // waits for a second the duration of
variable  i

}
```

ITERATIVE DELAY

```
/// Blink with for loop
//

int LED = 13;   // LED connected to digital pin 13

int i;      // Variable to hold our random value
void setup()
{
  pinMode(LED, OUTPUT);    // sets the digital
// pin as output
}

void loop()
{
   // set I to 1; while i is less that 100 increment i by one

   for(int i = 1;i <=100; i++){
  digitalWrite(LED, HIGH);   // turns the LED on
  delay(i);                  // set delay based on the value of i
  digitalWrite(LED, LOW);    // turns the LED off
  delay(i);                   // set delay based on the value of i
  }
}
```

I MADE AN LED BLINK?

-Most actuators are switched on and off with a digital output

-The digitalWrite() command is the software portion of being able to control just about anything

-LEDs are easy, motors come in a bit

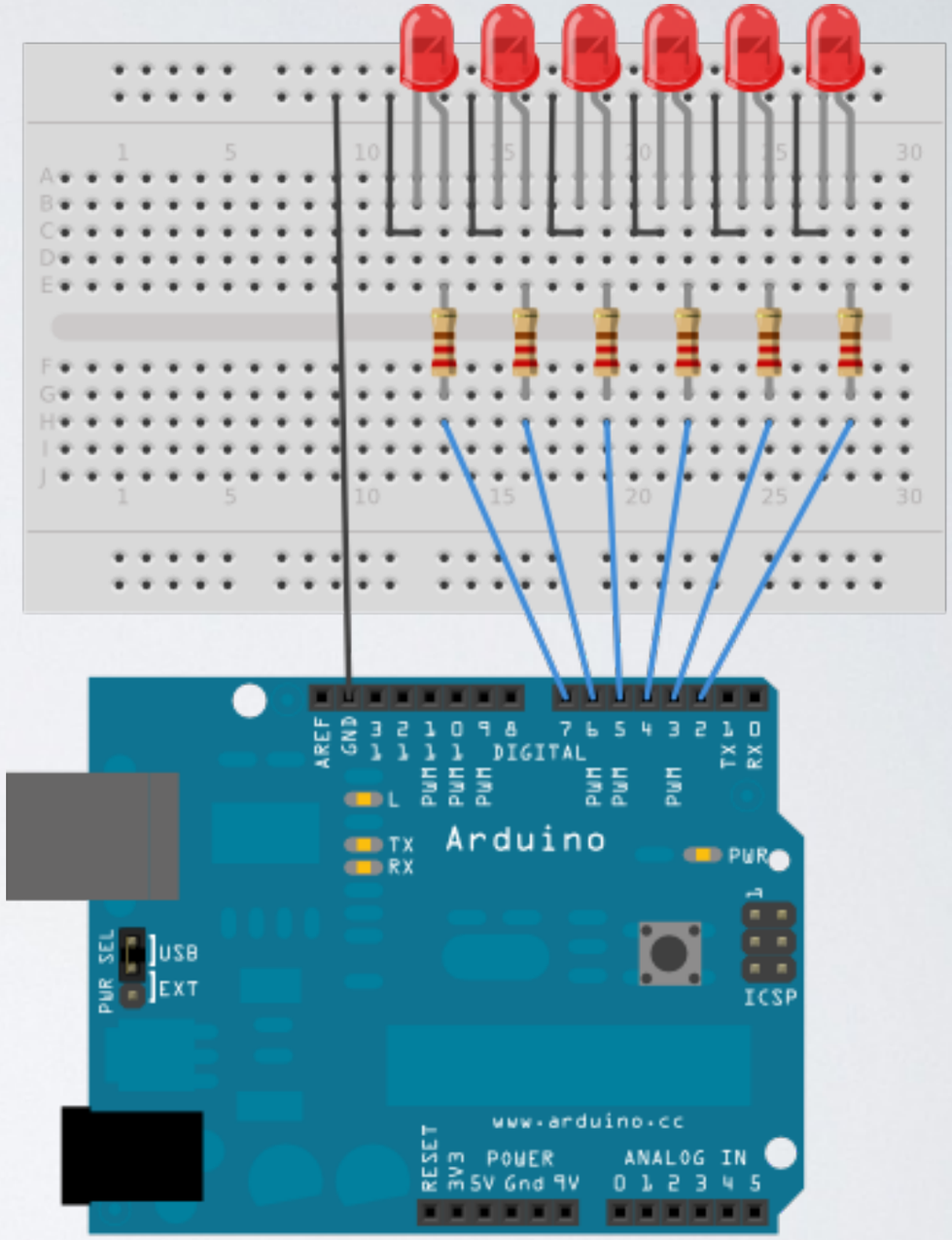
-Arduino has up to 13 digital outputs, and you easily can add more with helper chips

I MADE AN LED BLINK?

Cool Example!

http://vimeo.com/8196236

KNIGHT RIDER



KNIGHT RIDER

Parts for this project

- Solderless Breadboard

- 13 x Flexible Wire Jumpers

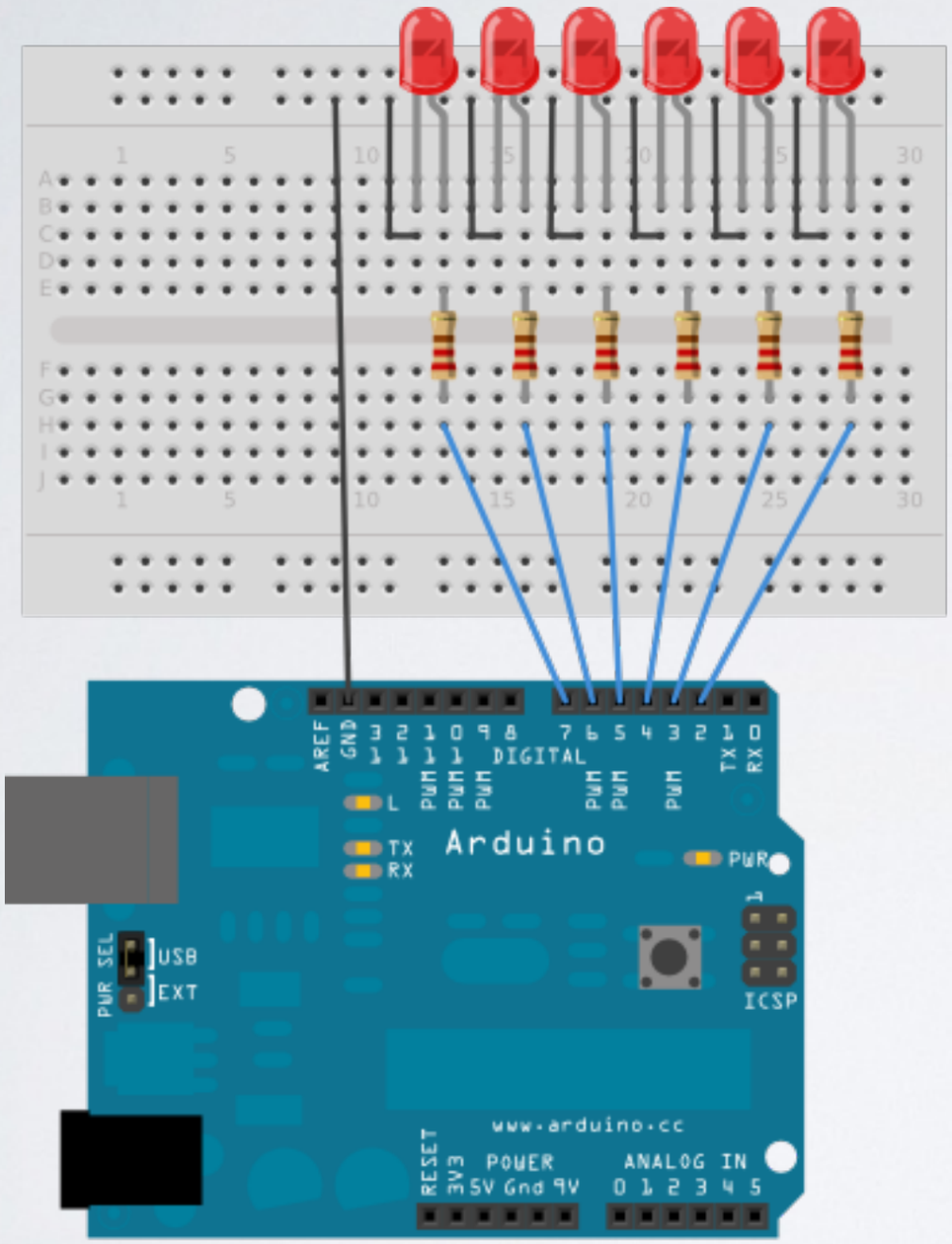
- 6 x LEDs (any color)

- 6 x 220 Ohm Resistors

- Arduino Duo board

- USB Cable

KNIGHT RIDER



Steps

-Connect the cathode of each LED to Ground

-Put a resistor in series with the Anode of each LED

-Connect the each resistor to pins 2-7

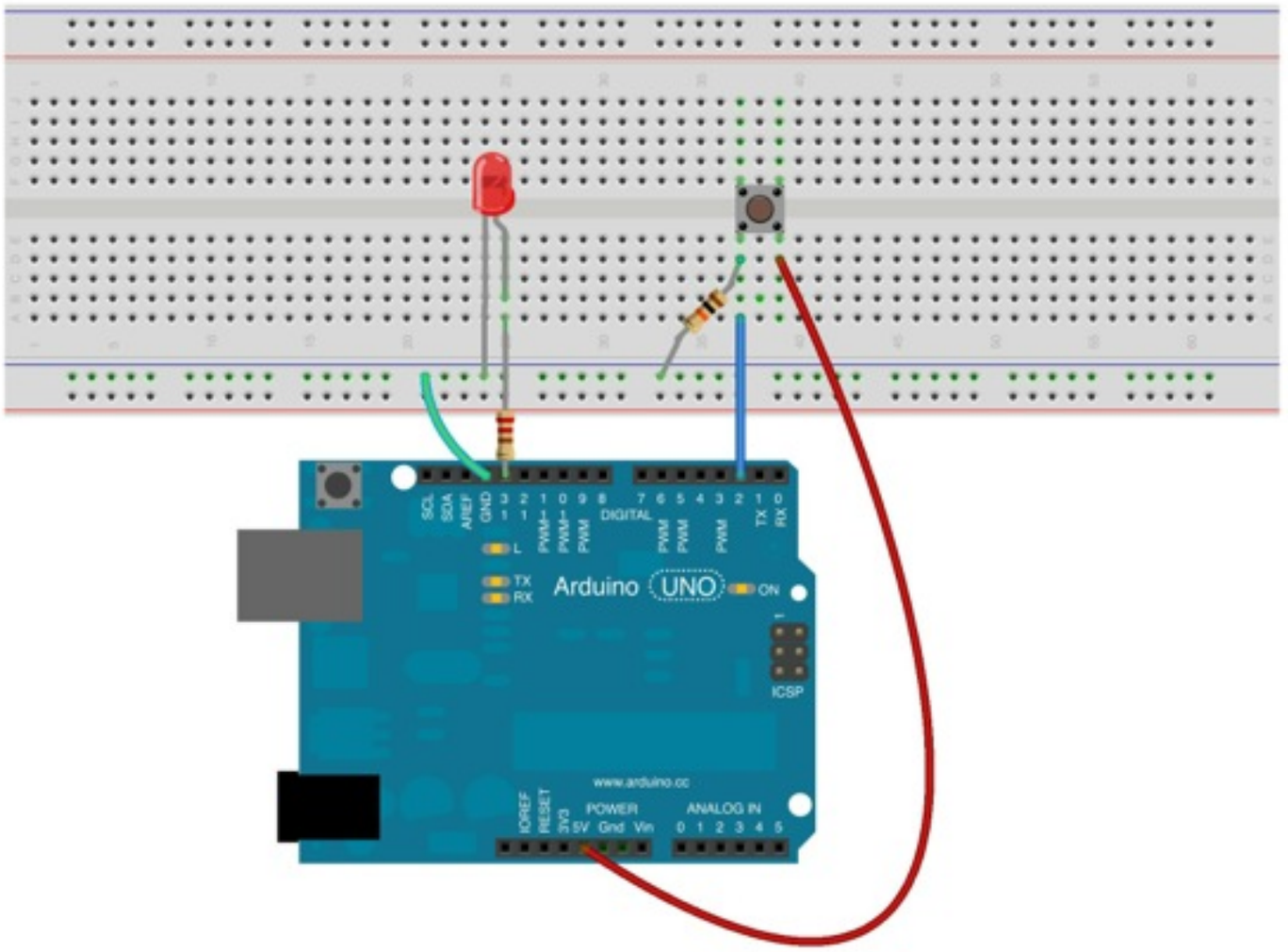
KNIGHT RIDER 1

```
/* Knight Rider 1
 * --------------


int pin2 = 2;
int pin3 = 3;
int pin4 = 4;
int pin5 = 5;
int pin6 = 6;
int pin7 = 7;
int timer = 100;

void setup(){
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);
  pinMode(pin5, OUTPUT);
  pinMode(pin6, OUTPUT);
  pinMode(pin7, OUTPUT);
}

void loop() {
   digitalWrite(pin2, HIGH);
   delay(timer);
   digitalWrite(pin2, LOW);
   delay(timer);

   digitalWrite(pin3, HIGH);
   delay(timer);
   digitalWrite(pin3, LOW);
   delay(timer);

   digitalWrite(pin4, HIGH);
   delay(timer);
   digitalWrite(pin4, LOW);
   delay(timer);

   digitalWrite(pin5, HIGH);
   delay(timer);
   digitalWrite(pin5, LOW);
   delay(timer);

   digitalWrite(pin6, HIGH);
   delay(timer);
   digitalWrite(pin6, LOW);
   delay(timer);

   digitalWrite(pin7, HIGH);
   delay(timer);
   digitalWrite(pin7, LOW);
   delay(timer);

   digitalWrite(pin6, HIGH);
   delay(timer);
   digitalWrite(pin6, LOW);
   delay(timer);

   digitalWrite(pin5, HIGH);
   delay(timer);
   digitalWrite(pin5, LOW);
   delay(timer);

   digitalWrite(pin4, HIGH);
   delay(timer);
   digitalWrite(pin4, LOW);
   delay(timer);
```

KNIGHT RIDER WITH A FOR LOOP

```
/* Knight Rider 2

int pinArray[] = {2, 3, 4, 5, 6, 7};
int count = 0;
int timer = 100;

void setup(){
  // we make all the declarations at once
  for (count=0;count<6;count++) {
    pinMode(pinArray[count], OUTPUT);
  }
}

void loop() {
  for (count=0;count<6;count++) {
   digitalWrite(pinArray[count], HIGH);
   delay(timer);
   digitalWrite(pinArray[count], LOW);
   delay(timer);
  }
  for (count=5;count>=0;count--) {
   digitalWrite(pinArray[count], HIGH);
   delay(timer);
   digitalWrite(pinArray[count], LOW);
   delay(timer);
  }
}
```

PUSHBUTTON


SCL
SDA
AREF
GND
DIGITAL
PWM~
PWM~
PWM
PWM
PWM
PWM
TX
RX
L
TX
RX
Arduino
UNO
ON
ICSP
www.arduino.cc
IOREF
RESET
3V3
5V
Gnd
Vin
POWER
ANALOG IN
0 1 2 3 4 5

PUSHBUTTON

Parts for this project

- Solderless Breadboard

- 3 x Flexible Wire Jumpers

- 1 x LEDs (any color)

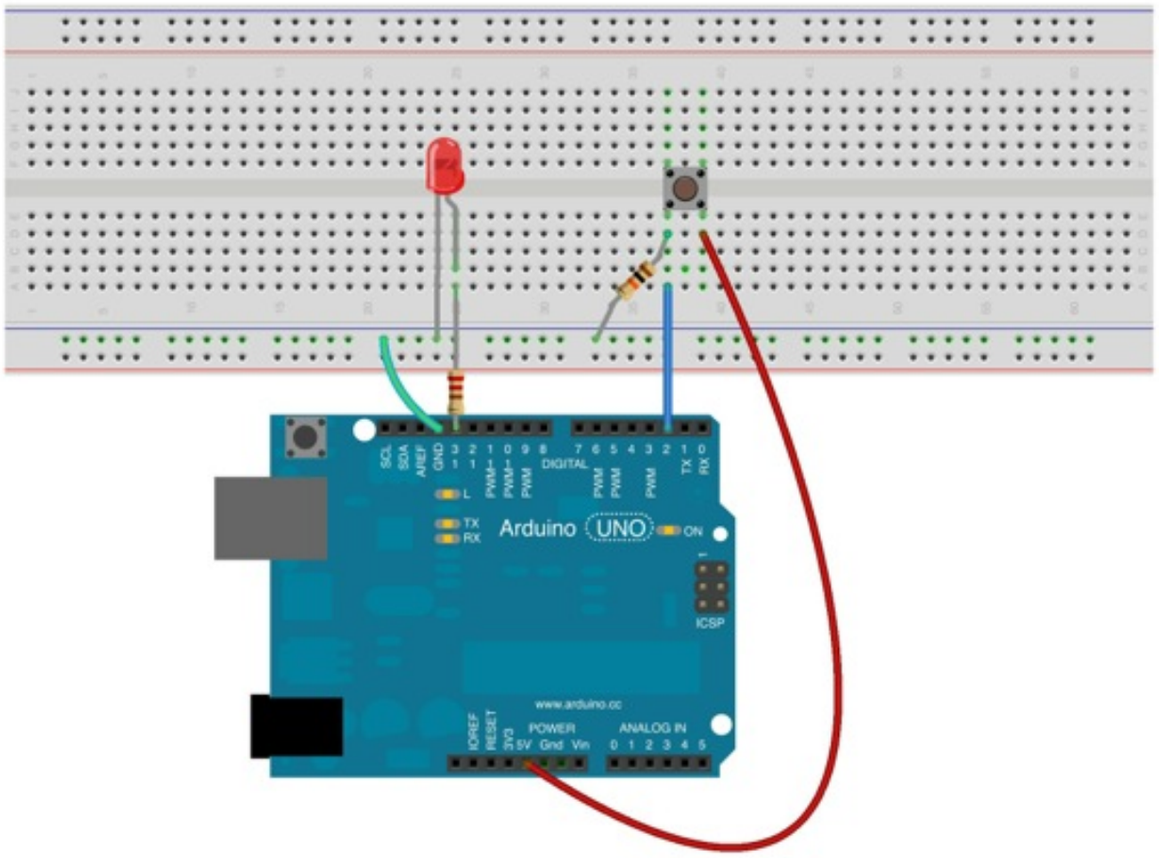
- 2 x 220 Ohm Resistors (bands are red,red,violet, gold)

- Arduino Duo board

- USB Cable

PUSHBUTTON

Steps



- Connect one side of the button to 5V

- On the other side of the button put a resistor to ground. In the same row connect a wire to pin 2

- Connect the cathode of the LED to ground.

- Connect the Anode of the LED to pin 13 in series with a resistor

PUSHBUTTON

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```