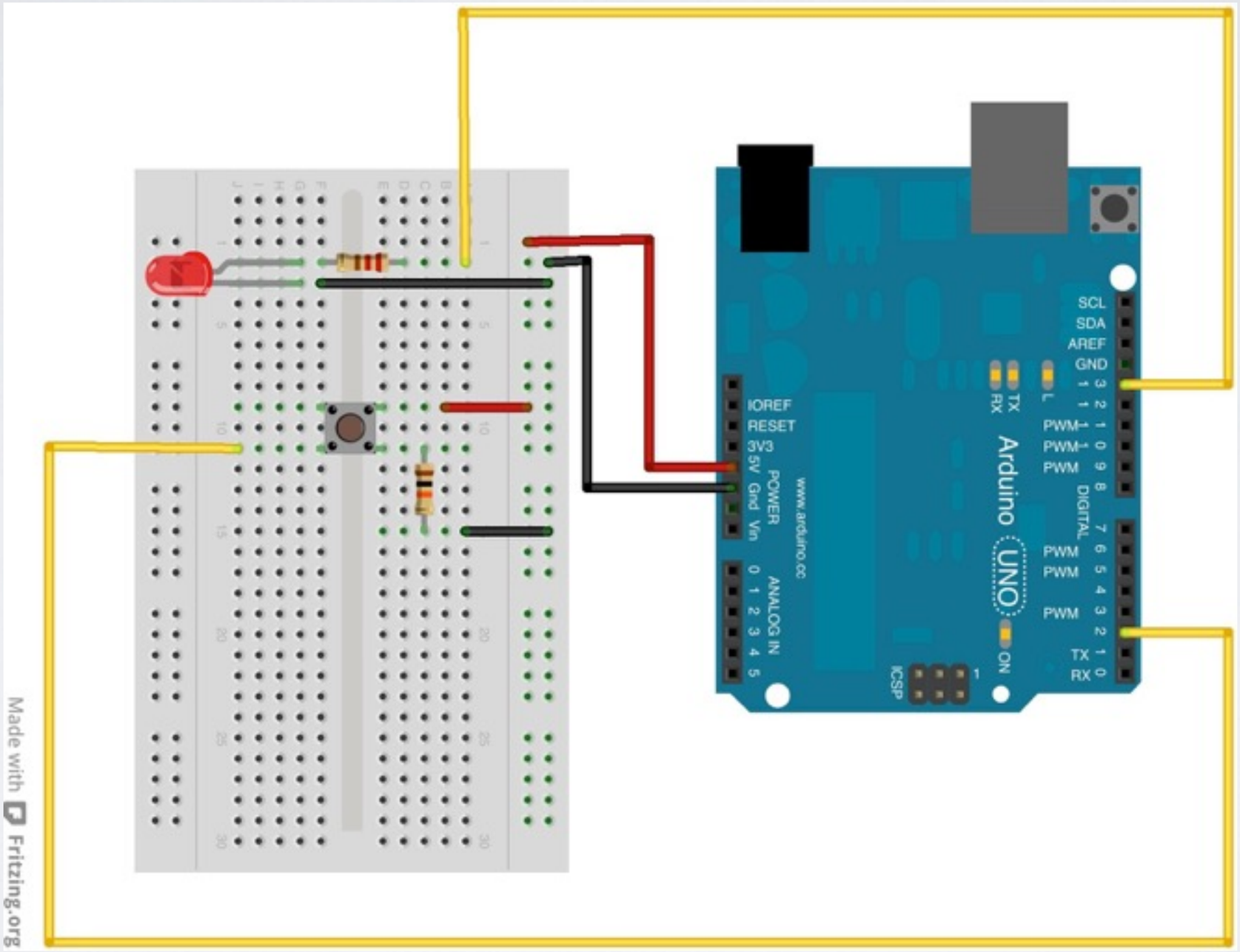
ARDUINO:

Intro to Microcontrollers - Week 2

Digital Input



ART

Nils Volker

http://www.nilsvoelker.com/

North Pitney

https://www.youtube.com/watch?v=V9KPgUzNcNk

ART

Zimoun

http://www.zimoun.net/2013-43.html

Survival Research Labs

https://www.youtube.com/watch?v=Ahj5-zV80c0&index=8&list=PLrG8Ed4_R2aAEiQa0xGAdDWBtj6GRiHsz

The Palace at 4am John Kessler

https://www.youtube.com/watch?v=k-uBggfiMGo

The Palace at 4 a.m.
Alberto Giacometti

https://www.moma.org/learn/moma_learning/alberto-giacometti-the-palace-at-4-a-m-1932

ARDUINO WITH OTHER SOFTWARE

http://www.creativeapplications.net/objects/solar-sinter-objects/

ARDUINO WITH OTHER SOFTWARE

“The two most important introductions for art in the past 20 years have been the Arduino and Processing,” Paola Antonelli, senior curator in the Department of Architecture and Design at the Museum of Modern Art.

http://www.nytimes.com/2011/03/17/arts/design/arduinos-provide-interactive-exhibits-for-about-30.html?_r=0

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)

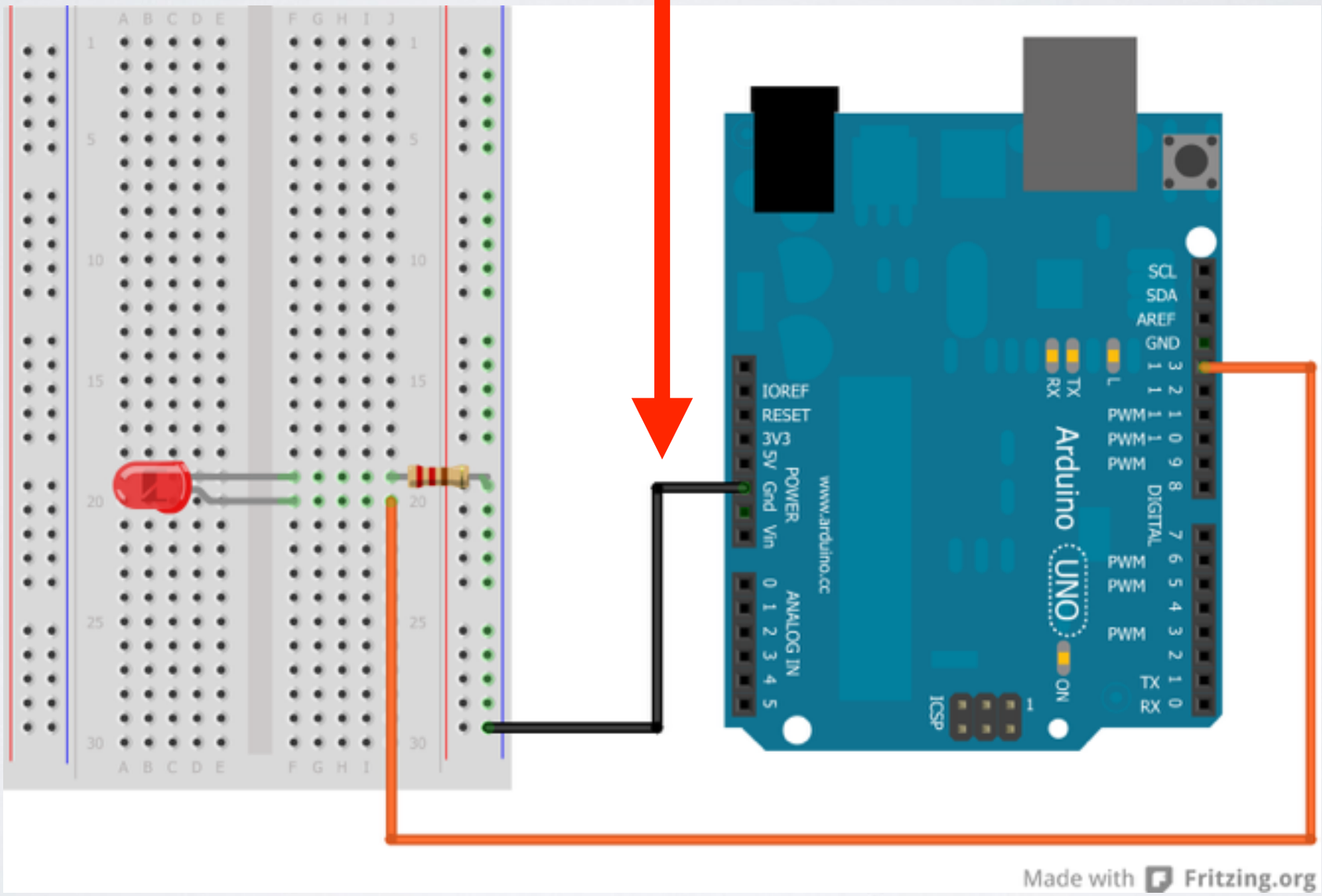
1 Breadboard

1 Arduino and USB cable

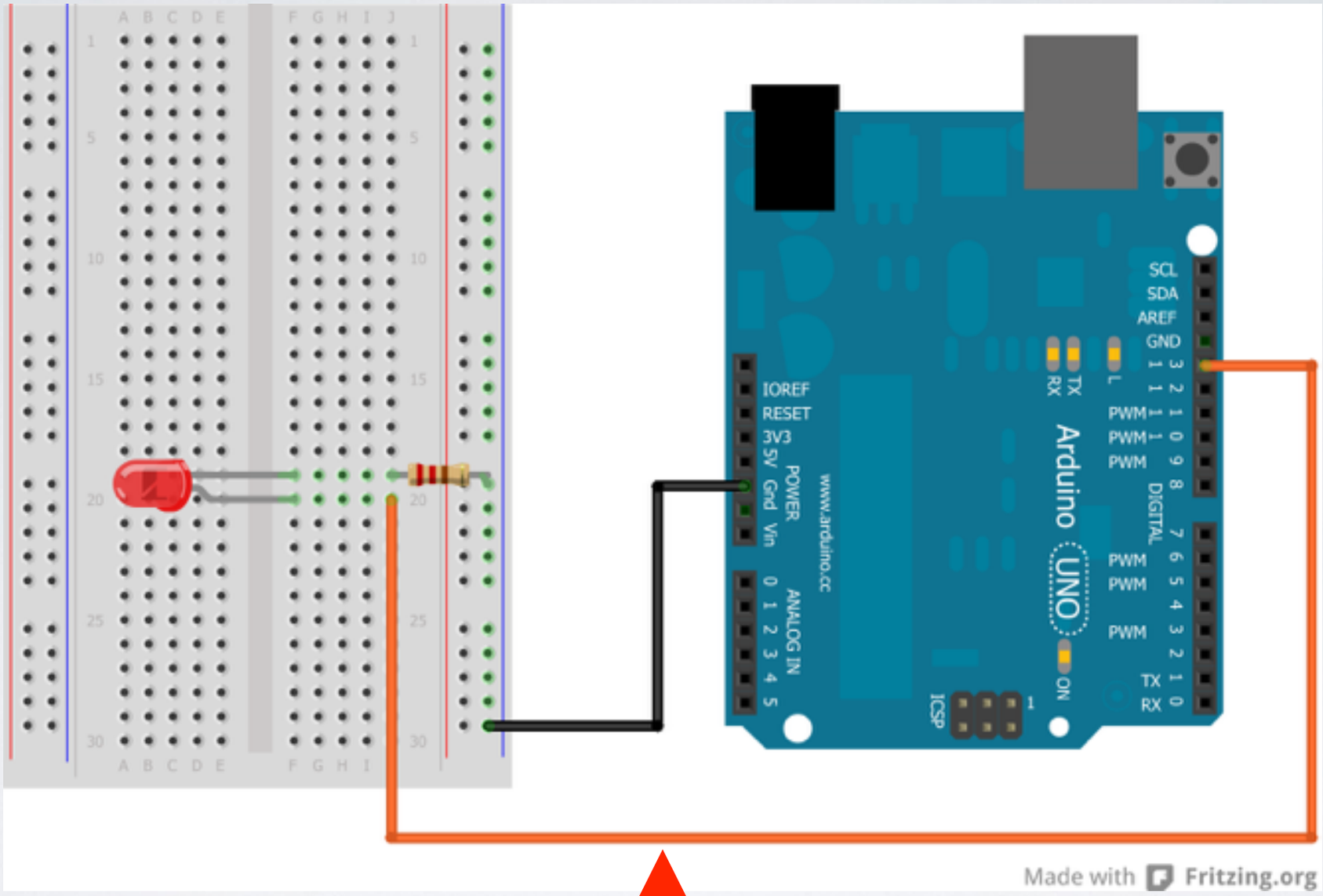
*Most of the parts from exercise 1

BLINKING AN LED

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

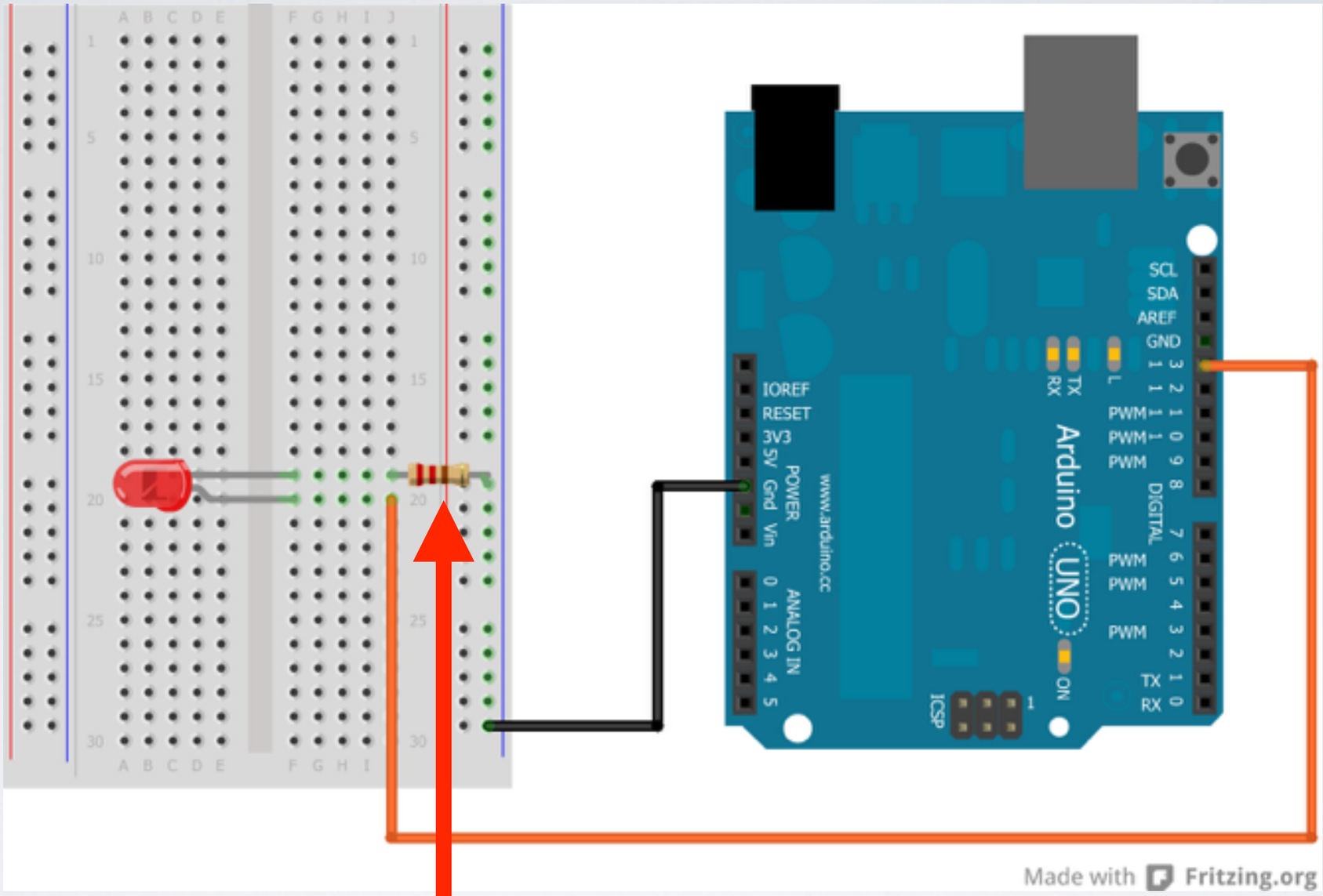


BLINKING AN LED



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.

KNIGHT RIDER

KNIGHT RIDER

Parts:

1 long green wire

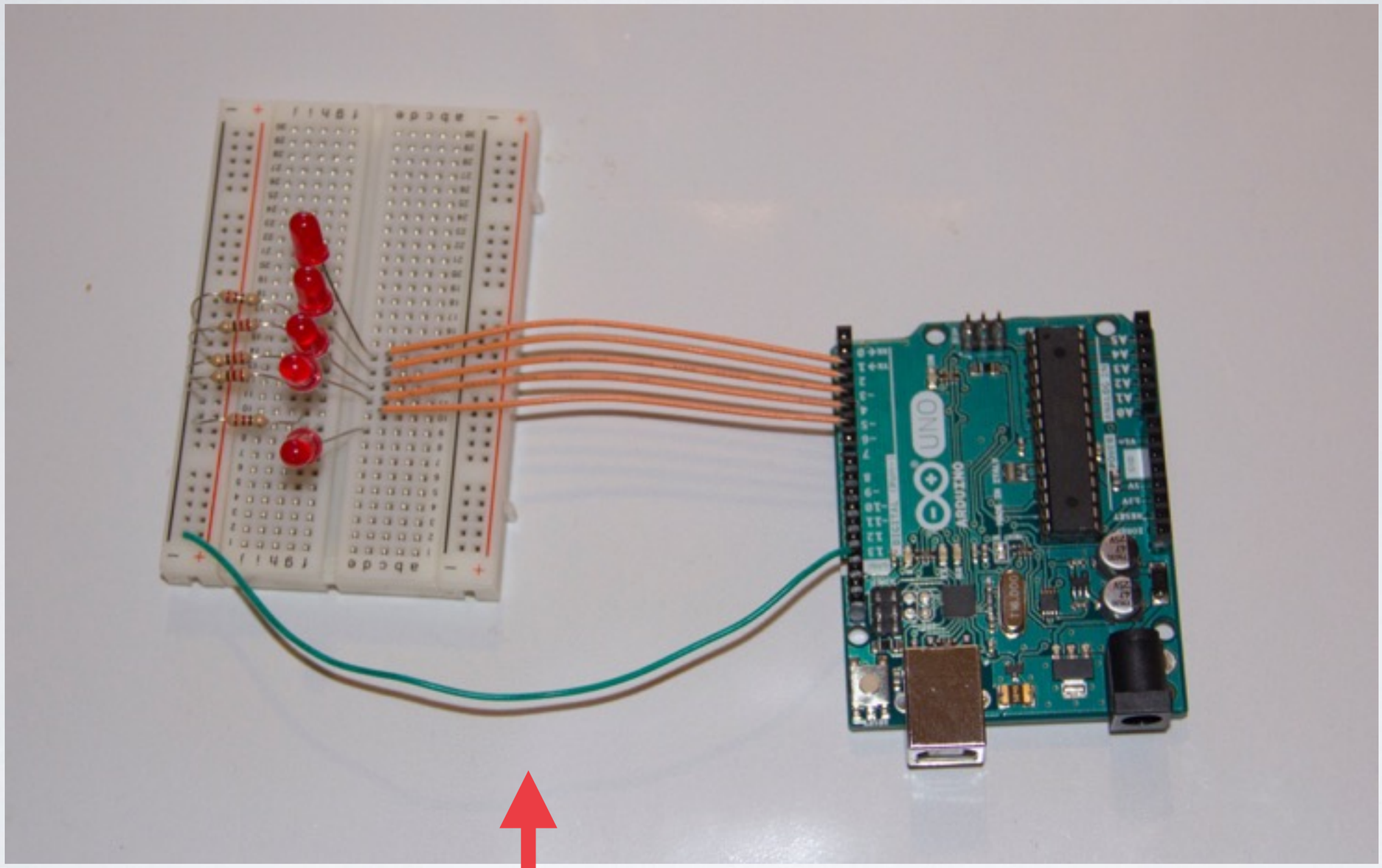
5 long orange wires

5 red Leds

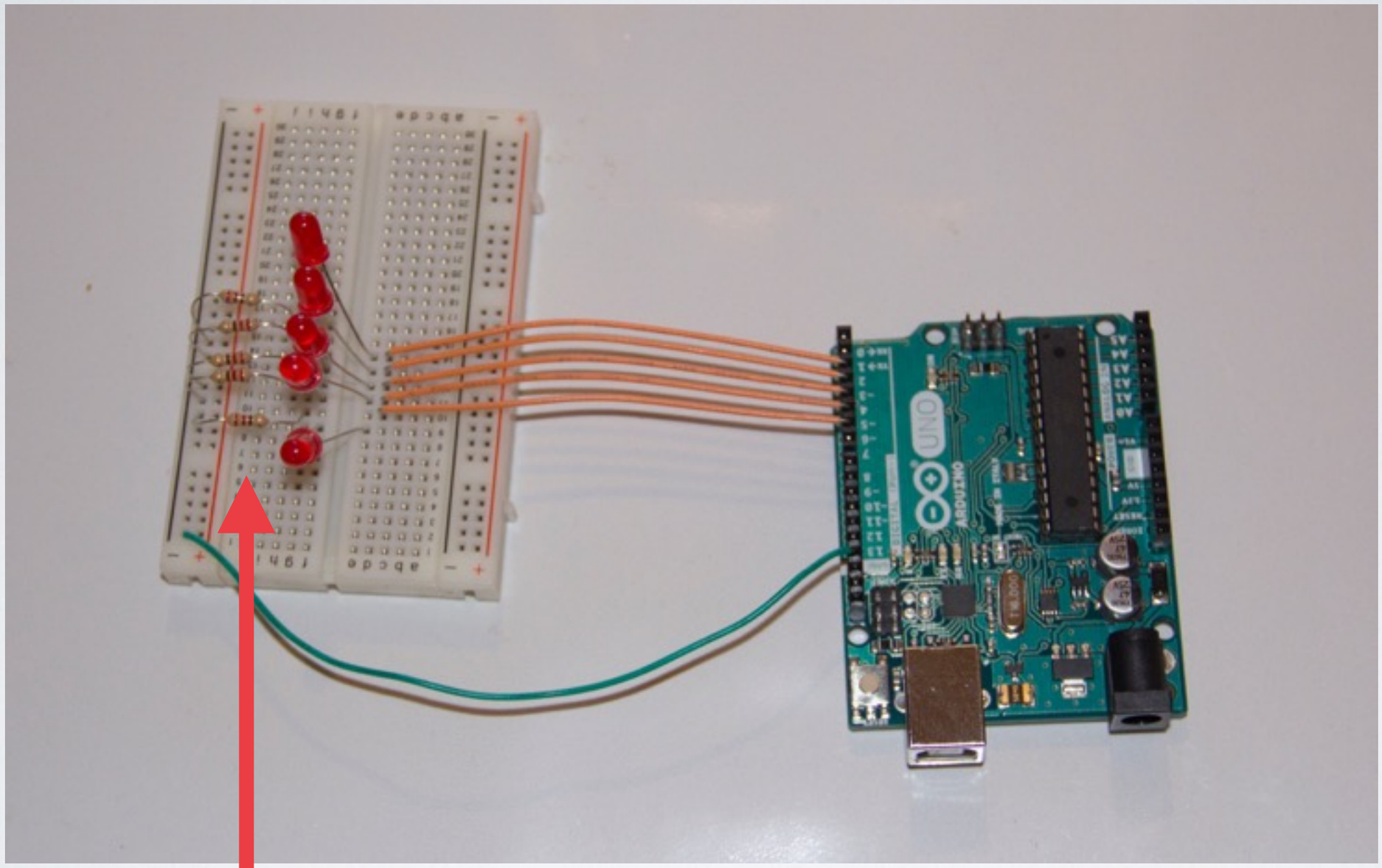
5 resistors (bands are red,red,violet, gold)

1 Breadboard

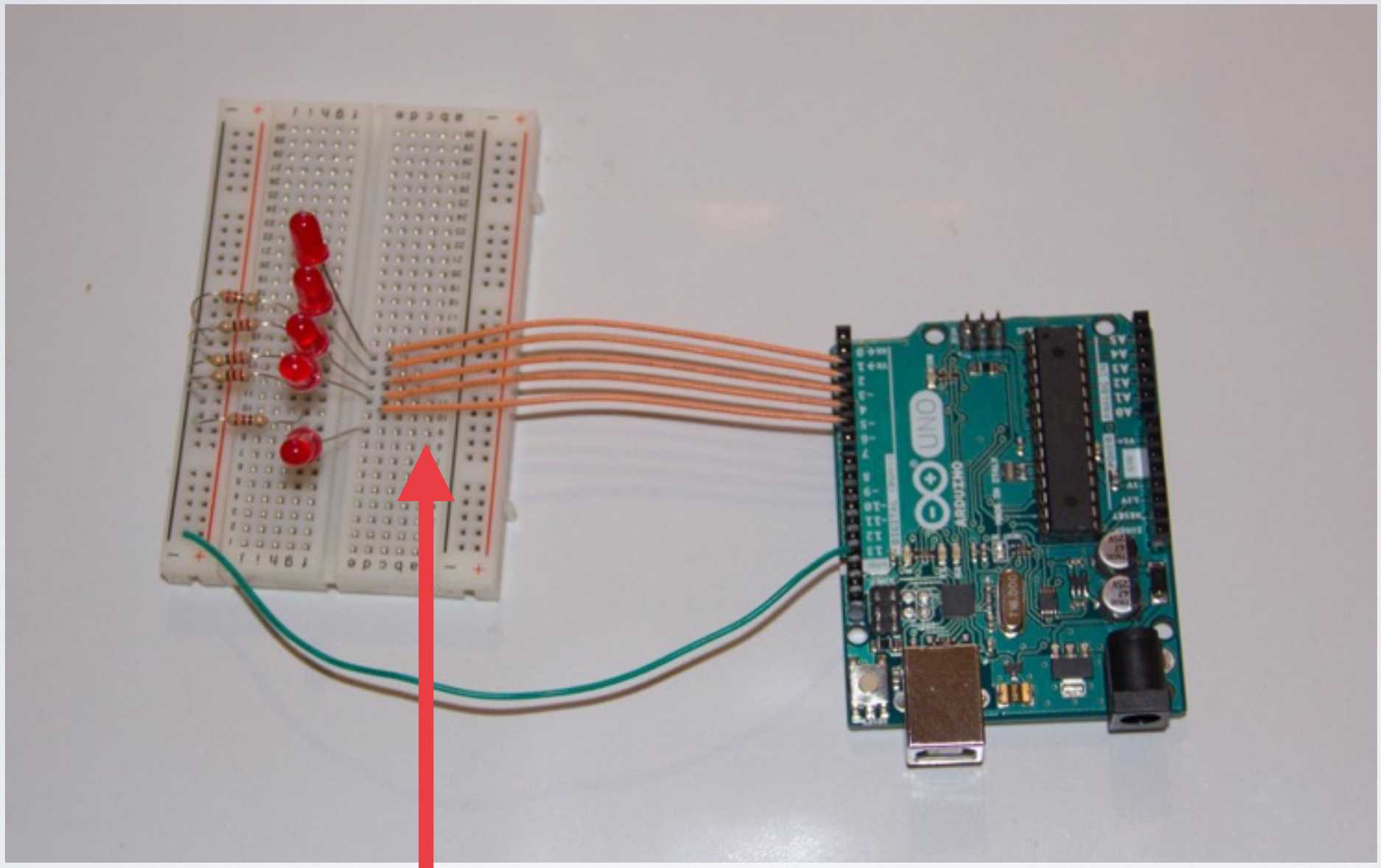
1 Arduino and USB cable



Connect GND from the Arduino to the - strip on breadboard



Connect 5 resistors from the - strip
to the short wire on the led



Attach the long wire on the LED
to pins 2 - 6 on the arduino

KNIGHT RIDER 1

Don’t type this in!

It’s in KnightRider 1 folder

```
/* 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);
```

KNIGHT RIDER WITH A FOR LOOP

```
/* Knight Rider 2

int pinArray[] = {2, 3, 4, 5, 6};
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);
  }
}
```

DIGITAL INPUT (BUTTON)

Parts for this project

- Solderless Breadboard

- 7 x Flexible Wire Jumpers

- 1 x LEDs (any color)

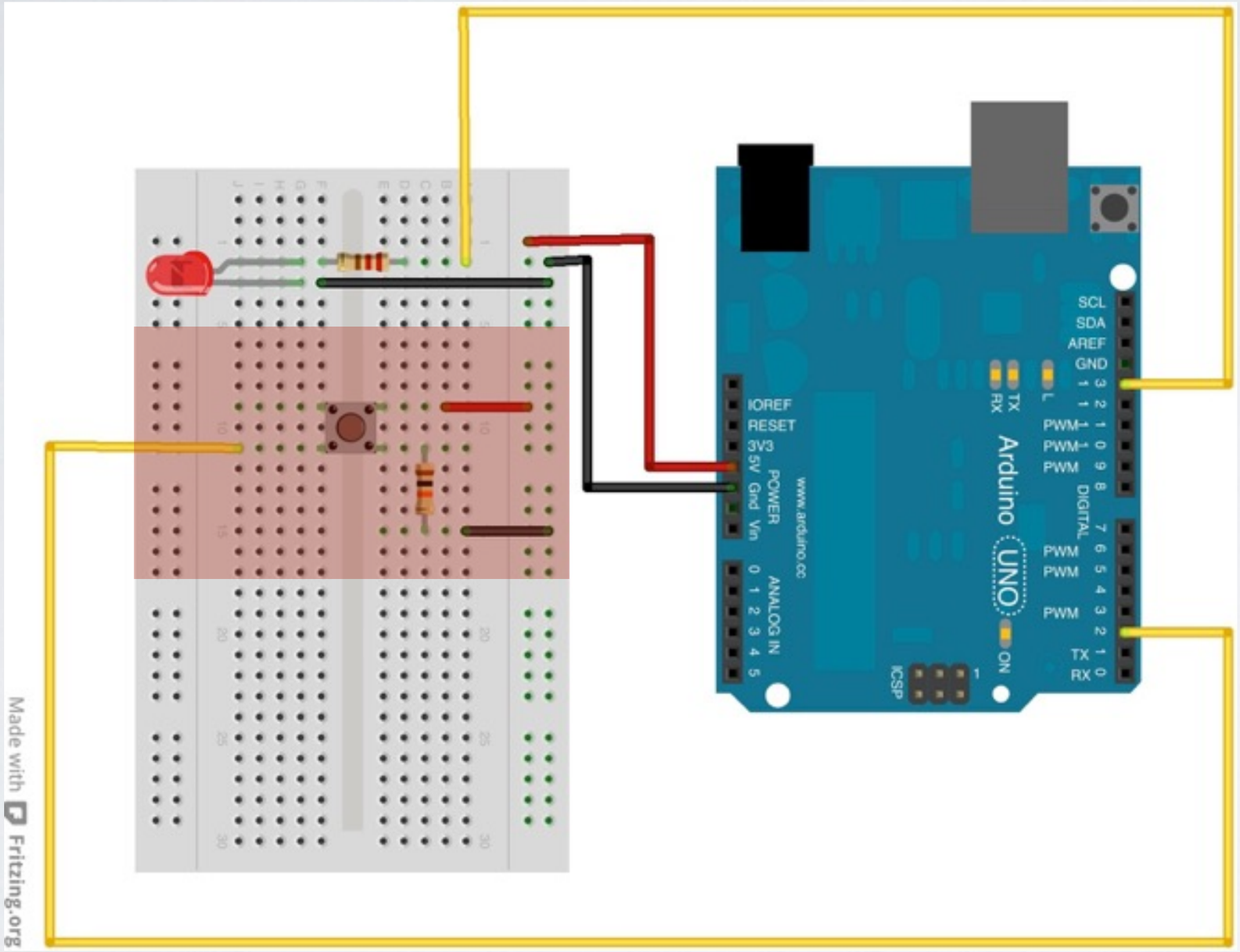
- 2 x 220 Ohm Resistors

- 1 x Tactile Pushbutton

- Arduino Duo board

- USB Cable

DIGITAL INPUT (BUTTON)



Connect 5V and GND to the side strips

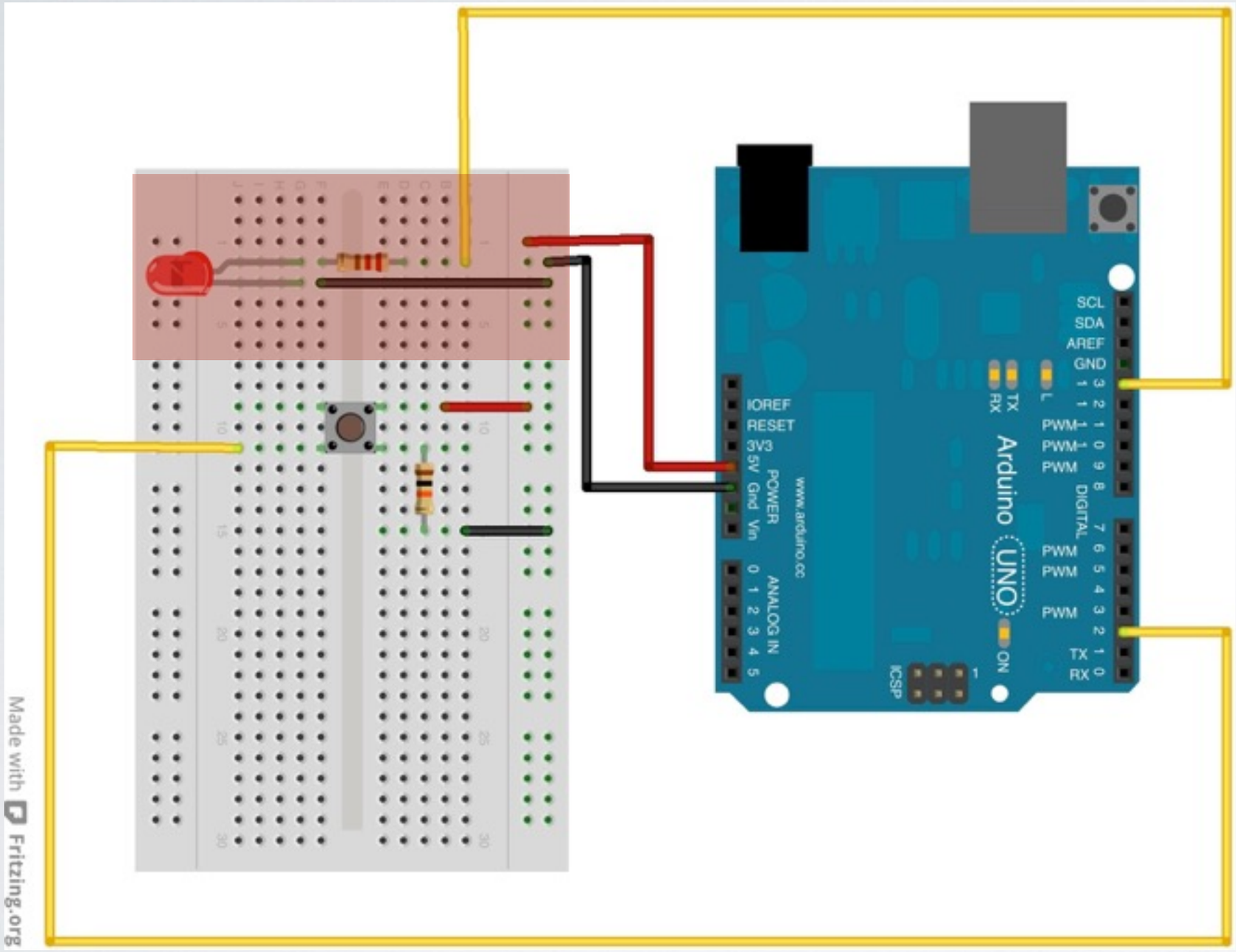
Add a pushbutton across the center vertical row.

Connect the top pin to 5V.

Connect lower side to ground in series with a resistor.

Connect the opposite lower pin of the switch to pin 2

DIGITAL INPUT (BUTTON)



Connect an LED, cathode to ground pin and anode in upper row.

Place a resistor in series with the anode(long lead of LED)

Connect the resistor to Pin 13

DIGITAL INPUT (BUTTON)

```
// 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);
  }
}
```

DIGITAL INPUT (BUTTON)

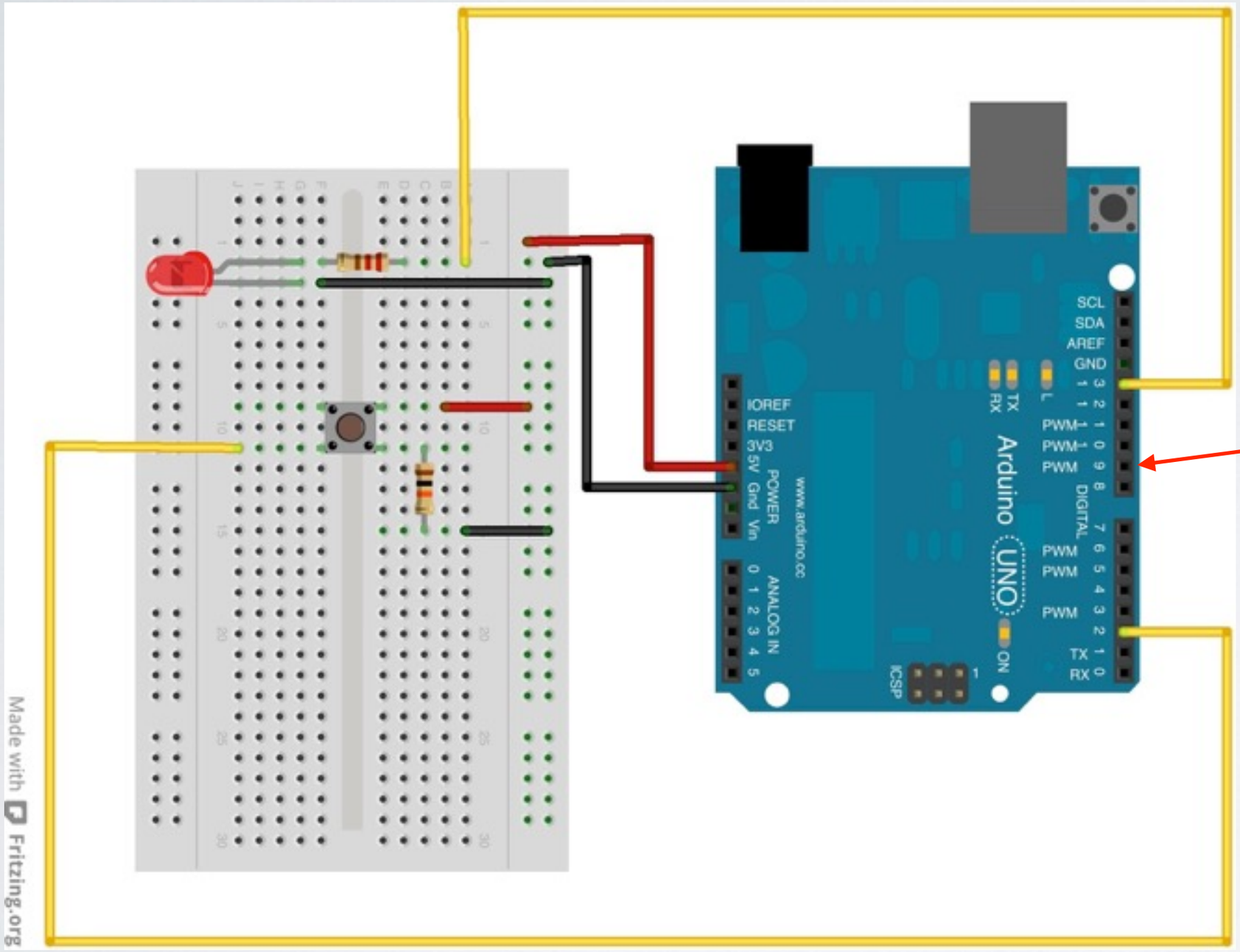
digitalRead()

Syntax:
digitalRead(pin)

Parameters
pin: the number of the digital pin you want to read (int)

Returns
HIGH or LOW

VARIATION 1



Keep Same Circuit

*Except, Move the LED pin from 13 to pin 9

FADE OR BLINK

Upload Fade_Or_Blink from the code folder.

```
/*
 * FadeOrBlink
 *
 *
 */

int ledPin = 9;                // choose the pin for the LED
int inputPin = 2;              // choose the input pin (for a pushbutton)
int val = 0;                   // variable for reading the pin status
int fadeval = 0;

void setup() {
  pinMode(ledPin, OUTPUT);     // declare LED as output
  pinMode(inputPin, INPUT);    // declare pushbutton as input
}

void loop(){
  val = digitalRead(inputPin);  // read input value
  if (val == HIGH) {            // pushed button means do blinking
    digitalWrite(ledPin, LOW);  // turn LED OFF
    delay(50);
    digitalWrite(ledPin, HIGH); // turn LED ON
    delay(50);
  }
  else {  // else button isn't pressed so do fading
    for(fadeval = 0 ; fadeval <= 255; fadeval+=5) { // fade in (from min to max)
      analogWrite(ledPin, fadeval);          // sets the value (range from 0-255)
      delay(10);
    }
    for(fadeval = 255; fadeval >=0; fadeval-=5) { // fade out (from max to min)
      analogWrite(ledPin, fadeval);
      delay(10);
    }
  }
}
```

FADE OR BLINK

Syntax:
analogWrite(pin, value)

Parameters:
pin: the pin to write to.

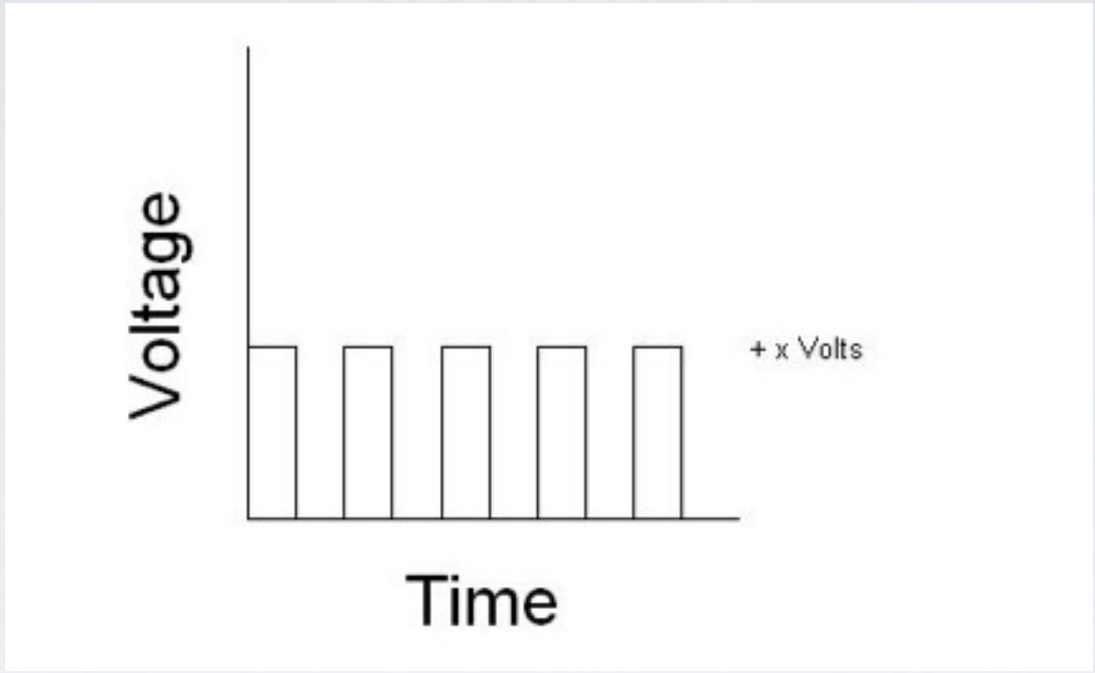
value: the duty cycle: between 0 (always off) and 255 (always on).

PULSE WIDTH MODULATION

-PWM stands for Pulse Width Modulation.

-Rather than limiting or controlling the voltage or current going through your LED, the voltage is fixed. When the full voltage is sent through the LED, it will emit the maximum of light it can (basically the same as connecting directly to a fully charged battery)

PULSE WIDTH MODULATION

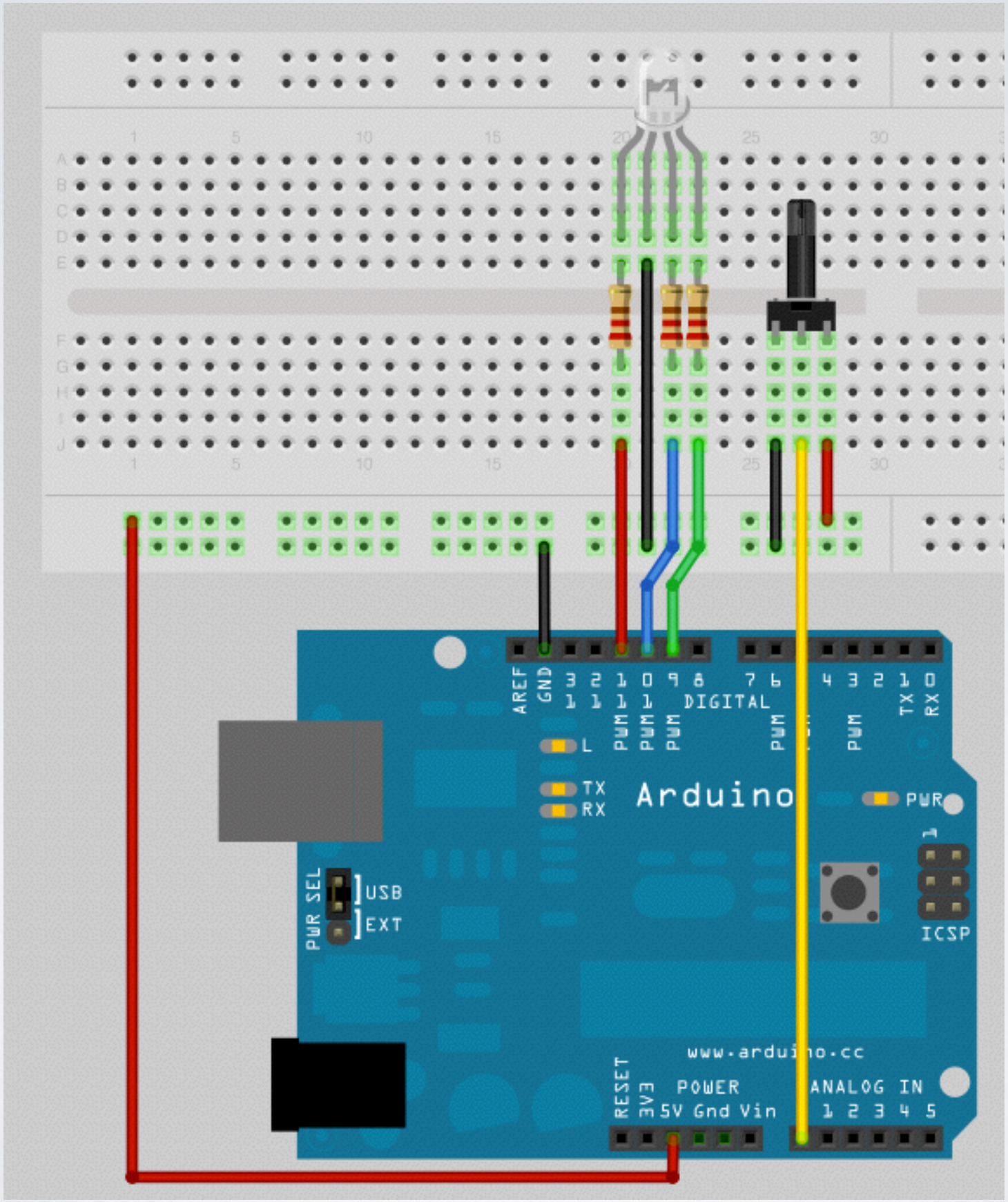


As you can see, we have now changed our constant voltage input for a square wave. In this graphic, our voltage is on half the time and off half the time.

The technical term here would be that we have a 50% duty cycle (50% of the time on.) With such an input on our LED, the amount of light given off appears diminished by 50%.

The reason I say it appears, is because it gives off its maximum amount of light, but only 50% of the time, so our eyes perceive only half the amount of light (more on this on the next slide).

ANALOG-IN ANALOG-OUT



DATA SHEETS

https://www.arduino.cc/en/Main/ArduinoStarterKit

CONTROL STRUCTURES
IF Statement

`if`, which is used in conjunction with a comparison operator, tests whether a certain condition has been reached, such as an input being above a certain number. The format for an if test is:

```
if (someVariable > 50)
{
  // do something here
}
```

CONTROL STRUCTURES
IF Statement

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120){ digitalWrite(LEDpin, HIGH); }

if (x > 120){
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}                                    // all are correct
```

CONTROL STRUCTURES

IF Statement Comparison Operators:

x == y (x is equal to y)

x != y (x is not equal to y)

x < y (x is less than y)

x > y (x is greater than y)

x <= y (x is less than or equal to y)

x >= y (x is greater than or equal to y)

CONTROL STRUCTURES
IF / ELSE

if/else allows greater control over the flow of code than the basic if statement, by allowing multiple tests to be grouped together. For example, an analog input could be tested and one action taken if the input was less than 500, and another action taken if the input was 500 or greater. The code would look like this:

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

CONTROL STRUCTURES
IF / ELSE

if/elseif Each test will proceed to the next one until a true test is encountered. When a true test is found, its associated block of code is run, and the program then skips to the line following the entire if/else construction. If no test proves to be true, the default else block is executed, if one is present, and sets the default behavior.

```
if (pinFiveInput < 500)
{
  // do Thing A
}
else if (pinFiveInput >= 1000)
{
  // do Thing B
}
else
{
  // do Thing C
}
```

CONTROL STRUCTURES
Switch Case

else Like if statements, switch...case controls the flow of programs by allowing programmers to specify different code that should be executed in various conditions. In particular, a switch statement compares the value of a variable to the values specified in case statements. When a case statement is found whose value matches that of the variable, the code in that case statement is run.

The break keyword exits the switch statement, and is typically used at the end of each case. Without a break statement, the switch statement will continue executing the following expressions ("falling-through") until a break, or the end of the switch statement is reached.

```
switch (var) {
    case 1:
      //do something when var equals 1
      break;
    case 2:
      //do something when var equals 2
      break;
    default:
      // if nothing else matches, do the default
      // default is optional
  }
}
```

CONTROL STRUCTURES
while loops

while loops will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

```
while(expression){
  // statement(s)
}
```

EXAMPLE

```
var = 0;
while(var < 200){
  // do something repetitive 200 times
  var++;
}
```

CONTROL STRUCTURES
do - while

The **do** loop works in the same manner as the **while** loop, with the exception that the condition is tested at the end of the loop, so the **do** loop will *always* run at least once.

```
do
{
    // statement block
} while (test condition);
```

EXAMPLE

Example

```
do
{
  delay(50);              // wait for sensors to
stabilize
  x = readSensors();   // check the sensors

} while (x < 100);
```

CONTROL STRUCTURES
break

break is used to exit from a do, for, or while loop, bypassing the normal loop condition. It is also used to exit from a switch statement.

```
Example
for (x = 0; x < 255; x ++)
{
    digitalWrite(PWMpin, x);
    sens = analogRead(sensorPin);
    if (sens > threshold){       // bail out on sensor
detect
        x = 0;
        break;
    }
    delay(50);
}
```

CONTROL STRUCTURES
continue

The continue statement skips the rest of the current iteration of a loop (do, for, or while). It continues by checking the conditional expression of the loop, and proceeding with any subsequent iterations.

```
for (x = 0; x < 255; x ++)
{
    if (x > 40 && x < 120){          // create jump in
values
        continue;
    }

    digitalWrite(PWMpin, x);
    delay(50);
}
```

CONTROL STRUCTURES
return

Terminate a function and return a value from a function to the calling function, if desired

Example

```
int checkSensor(){
    if (analogRead(0) > 400) {
        return 1;
    else{
        return 0;
    }
}
```

ASSIGNMENT

Create a circuit

1 form of input - a button

1 form of output LED, digital or pwd

Write a paragraph explaining hot it works

Post code and image to the LMS