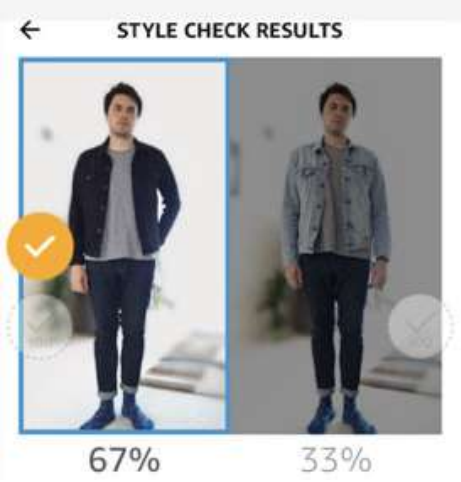


Internet of Things Sensors & Actuators

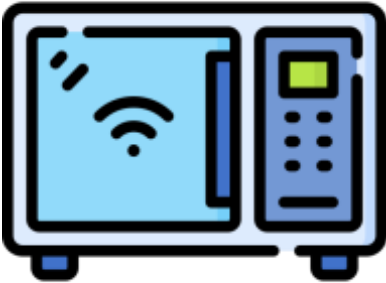
Abdallah El Ghamry



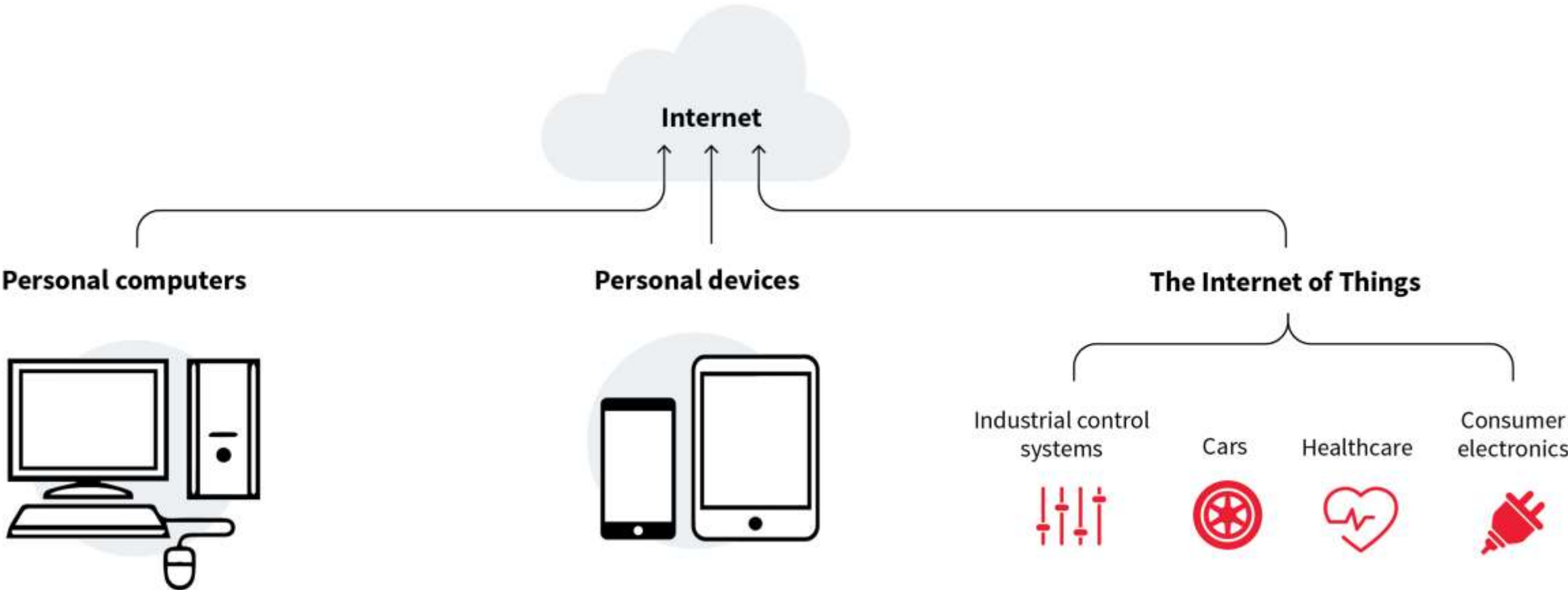
IoT Applications



Things



Internet of Things



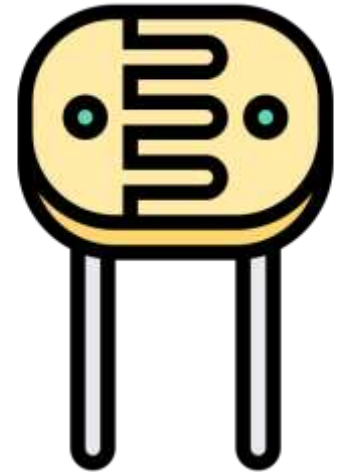
Internet of Things

- The Internet of Things (IoT) represents the **network of physical objects** “**Things**” that are integrated with sensors, software and other technologies for the purpose of exchanging data with other devices on **the Internet**.

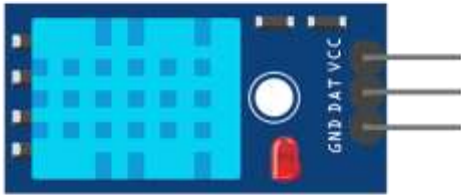


Sensors

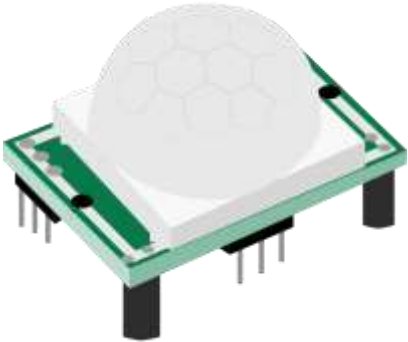
- A **sensor** is a device that detects some type of input from the **physical environment**.
- The input can be **light, heat, motion, pressure** or any number of other environmental phenomena.



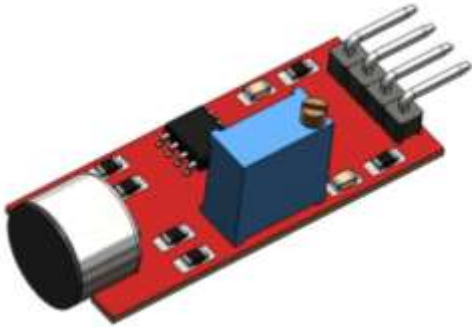
Sensors



Temperature and Humidity



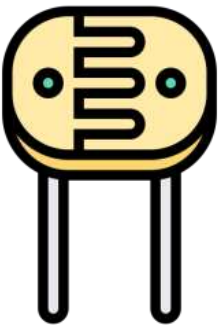
PIR Motion Detection



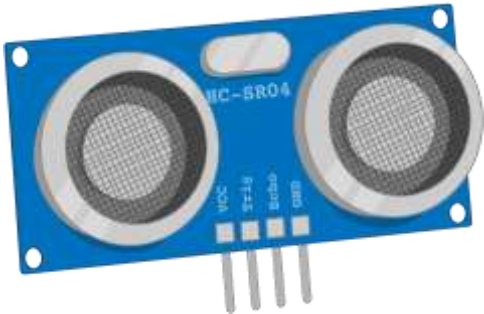
Microphone Sound Detection



Gas/Smoke Sensor



Photoresistor CdS Sensor



Ultrasonic Sensor



IR Obstacle Avoidance



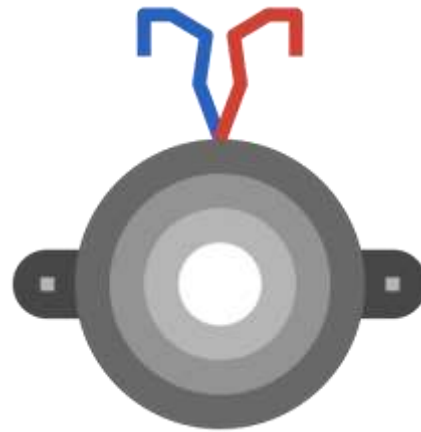
Heart Rate Sensor (ECG)

Actuators

- Sensors turn a **physical input** into an electrical output, while **actuators do the opposite**.
- Actuators take electrical signals from control modules and **turn them into physical outputs**.



LEDs



Buzzer



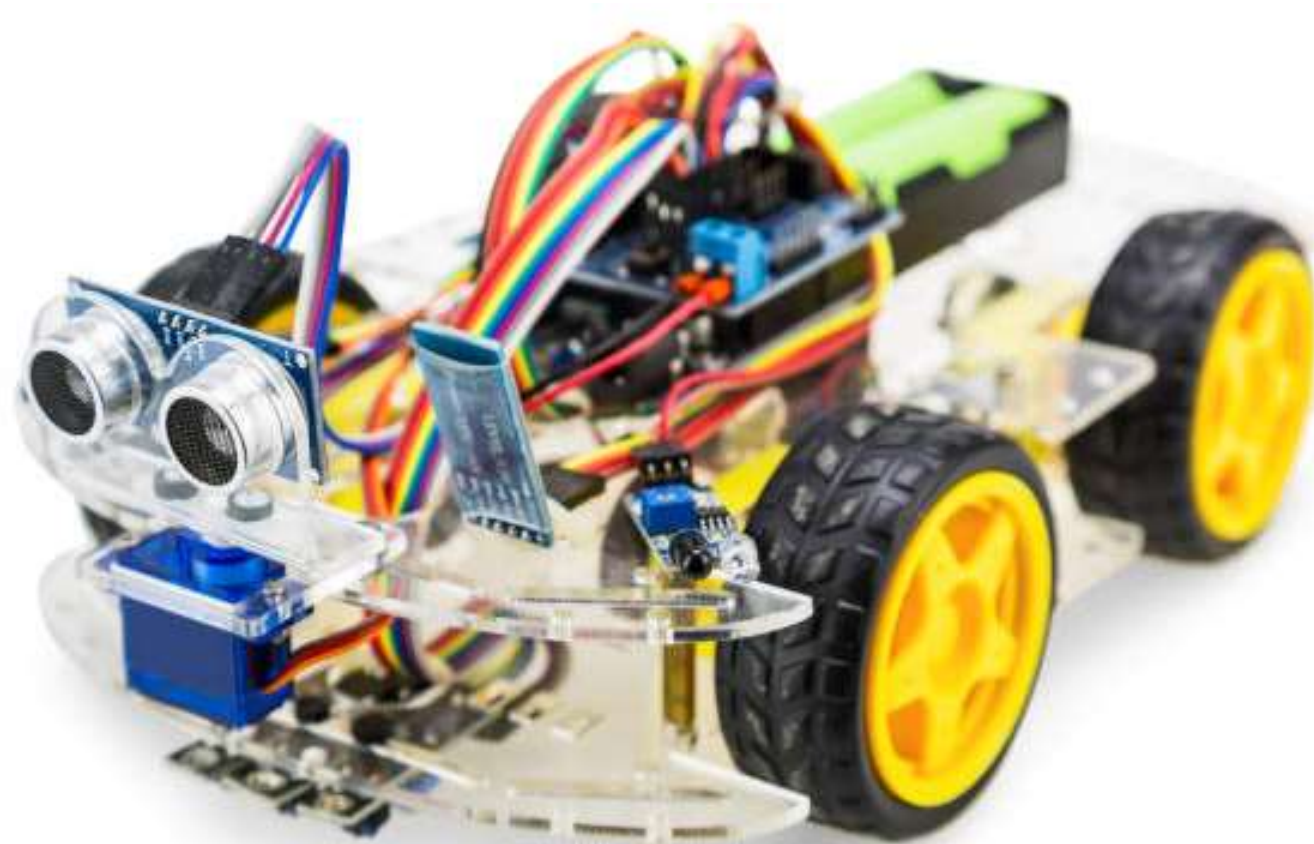
DC Fan



Servo Motor

Actuators: Servo Motor

- A servo motor is an electrical device which can **push or rotate an object** with great precision.

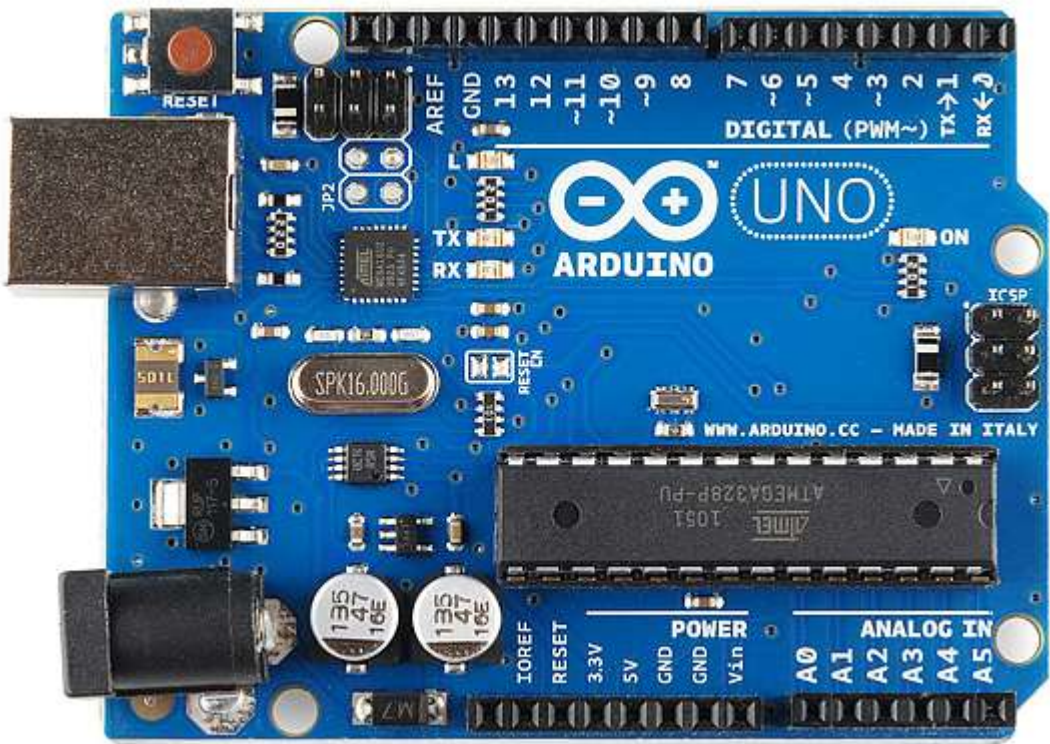


Actuators: Servo Motor

- The **HBE-ROBONOVA AI 3** is an intelligent robot with an MR-C3024 controller board capable of controlling **32 servo motors** simultaneously.



Processing



Arduino

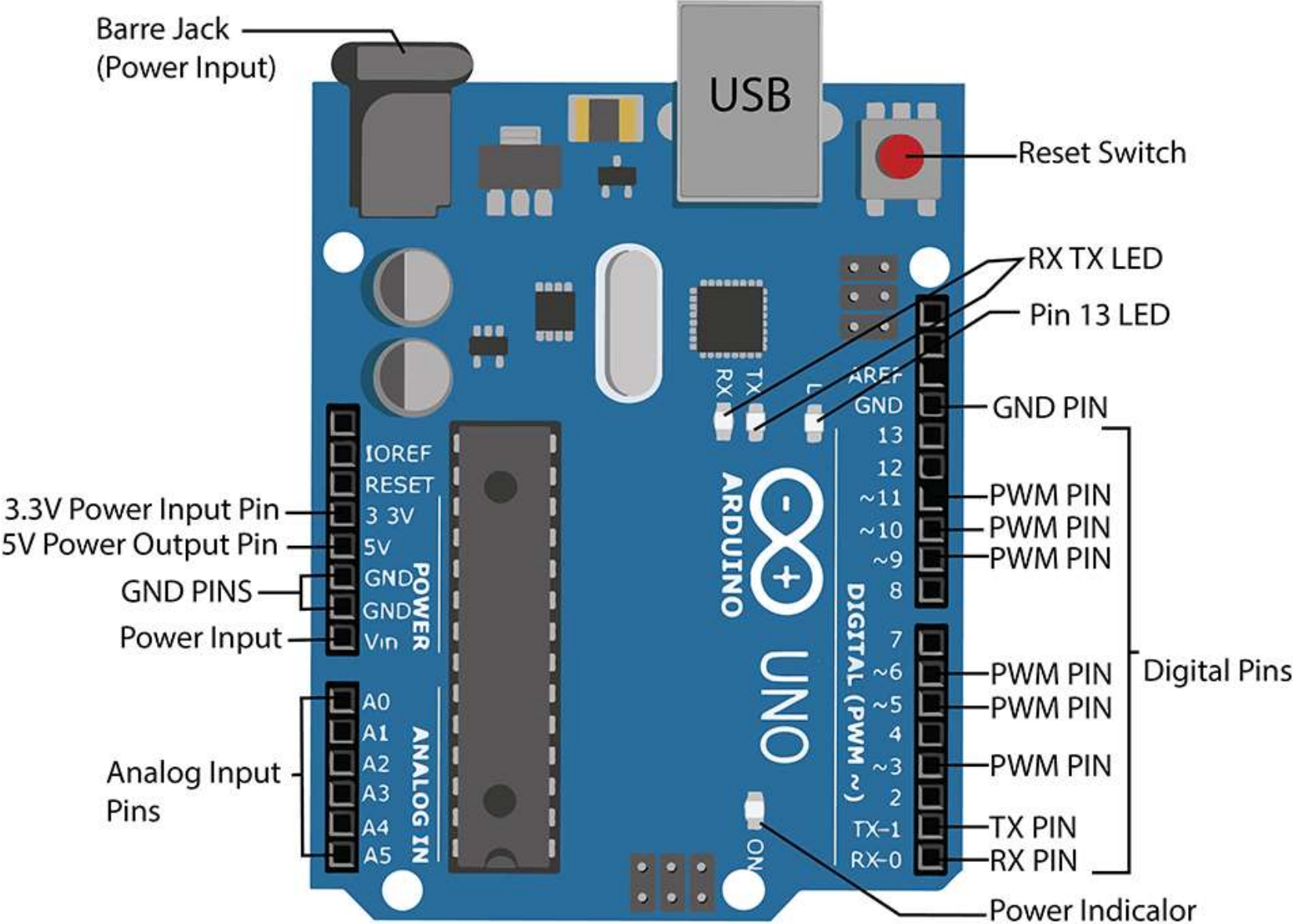


Raspberry Pi

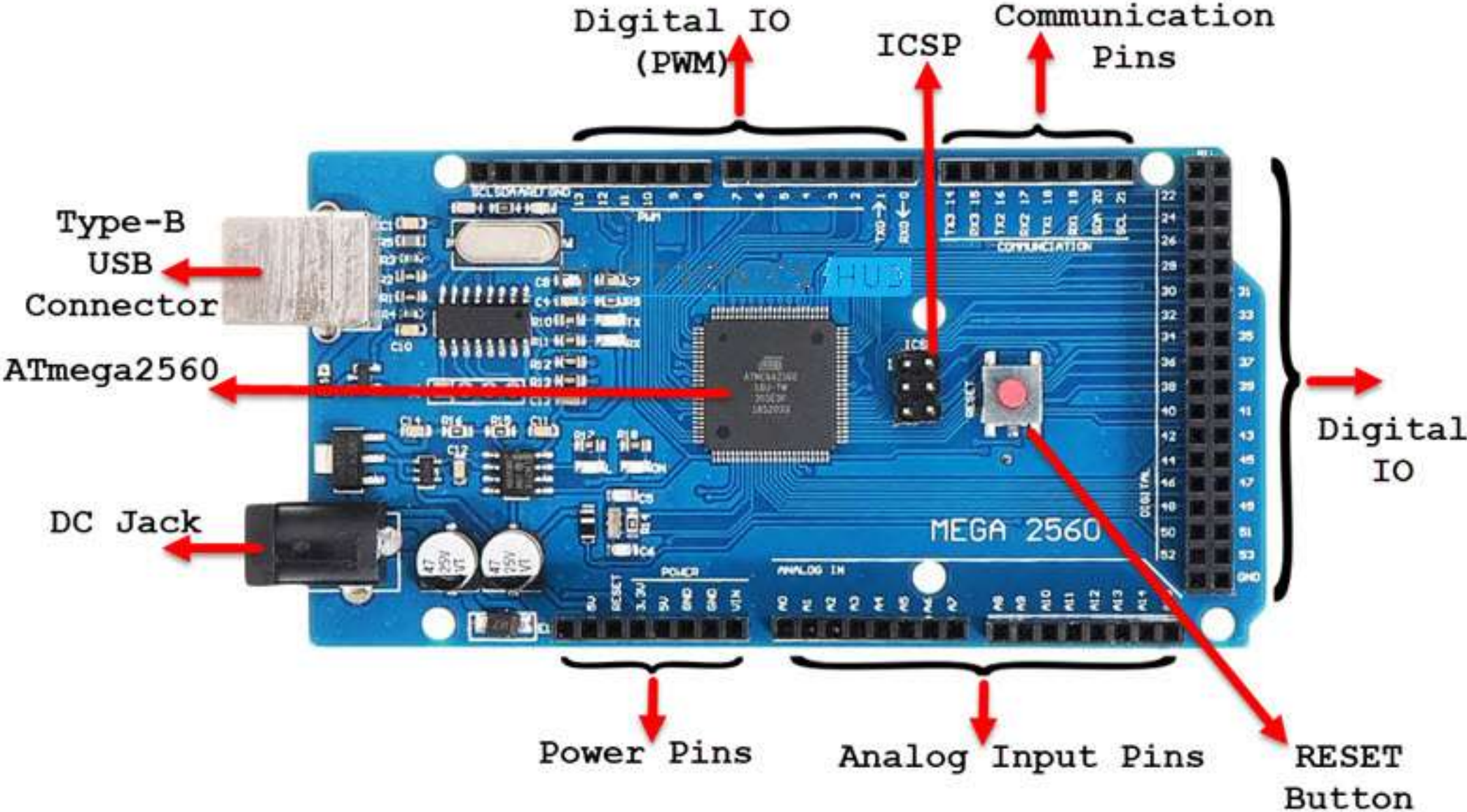
- Arduino is **open-source hardware** that can be used to develop **embedded systems** with **open-source software**.
- Arduino has gained **massive popularity** among **students** for making a working model.
- The reasons behind the popularity of Arduino are its **low cost**, **availability of software**, and **easy- to-interface** possibility.
- The Arduino environment has been designed to be **easy to use for beginners** who have **no software or electronics experience**.

- Arduino is used in many educational programs around the world, particularly by designers who want to easily create prototypes but do not need a deep understanding of the technical details.
- Because it is designed to be used by nontechnical people, the software includes plenty of example code to demonstrate how to use the Arduino board.
- People already working with microcontrollers are also attracted to Arduino because of its facility for quick implementation of ideas.

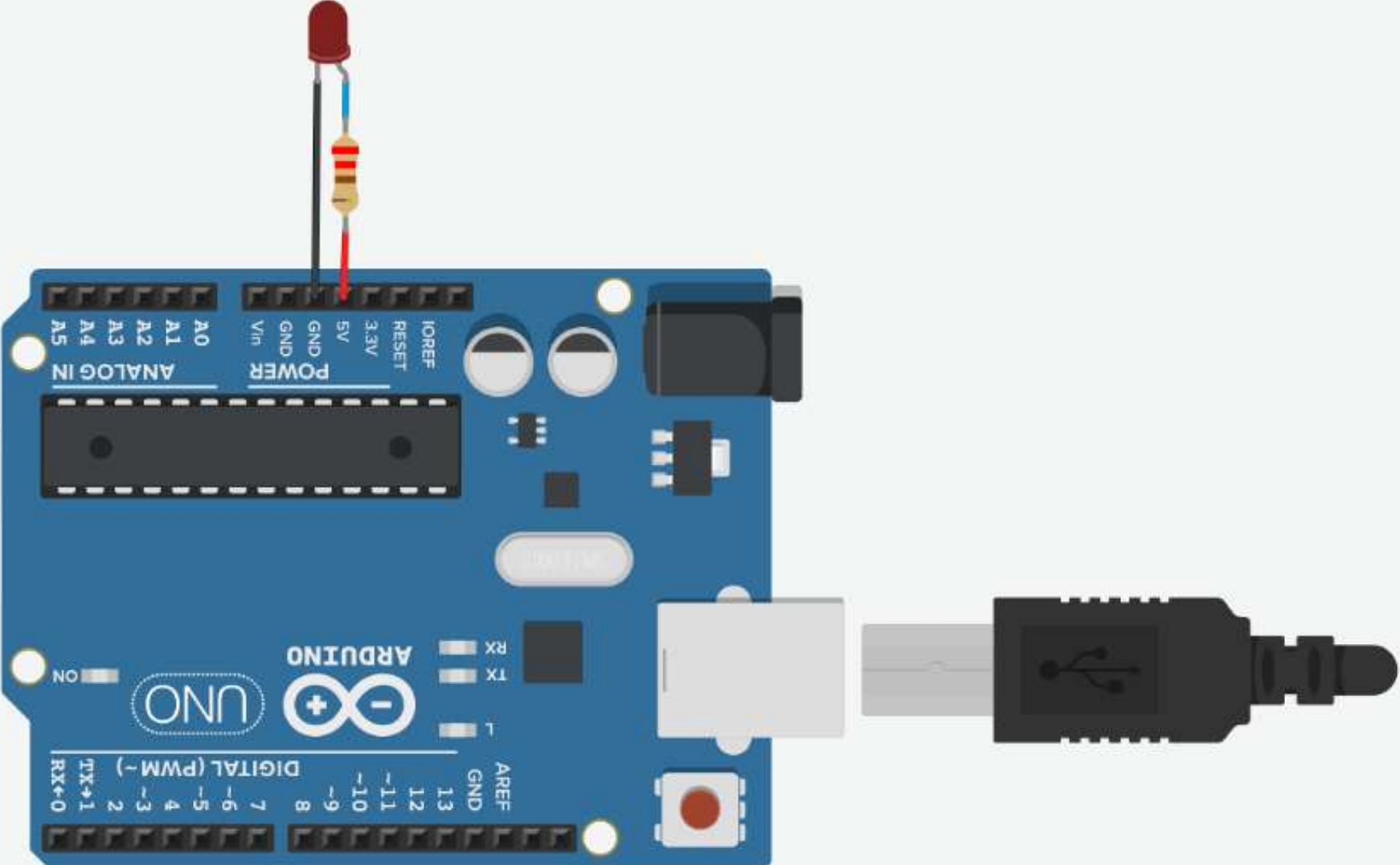
Arduino Uno Board



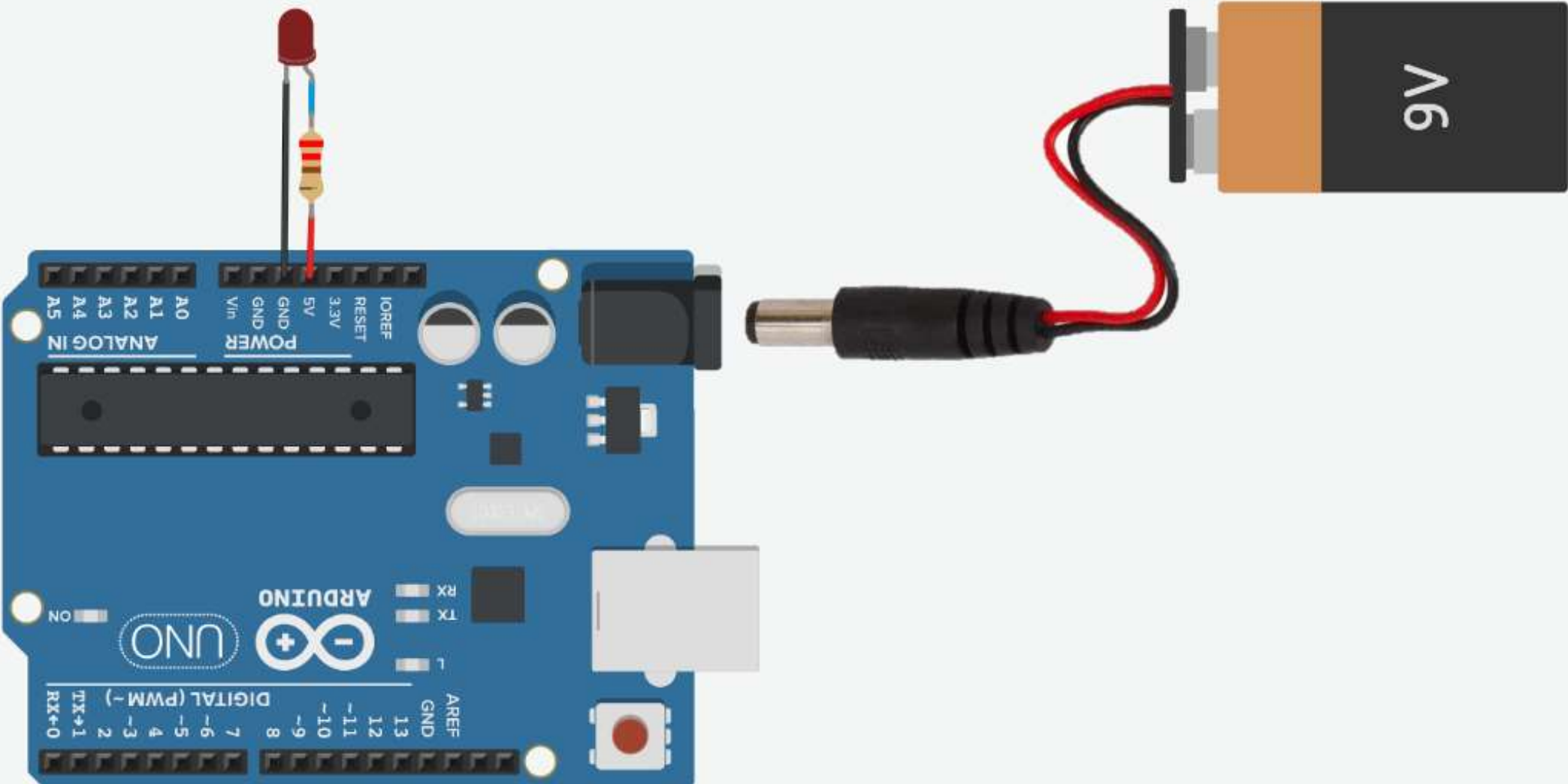
Arduino Mega Board




Connecting Arduino to Power



Connecting Arduino to Power



- The **Arduino IDE** enables you to **write and edit code** and convert this code into instructions that **Arduino hardware understands**.

A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.8.5". The main editor area shows the following code:

```
This example code is in the public domain.  
  
http://www.arduino.cc/en/Tutorial/Blink  
*/  
  
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

The status bar at the bottom shows "32" on the left and "Arduino/Genuino Uno on COM1" on the right.

Downloading Arduino IDE

- Go to <https://www.arduino.cc/en/software> website.



Downloads



Arduino IDE 2.0.3

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows .NET Installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

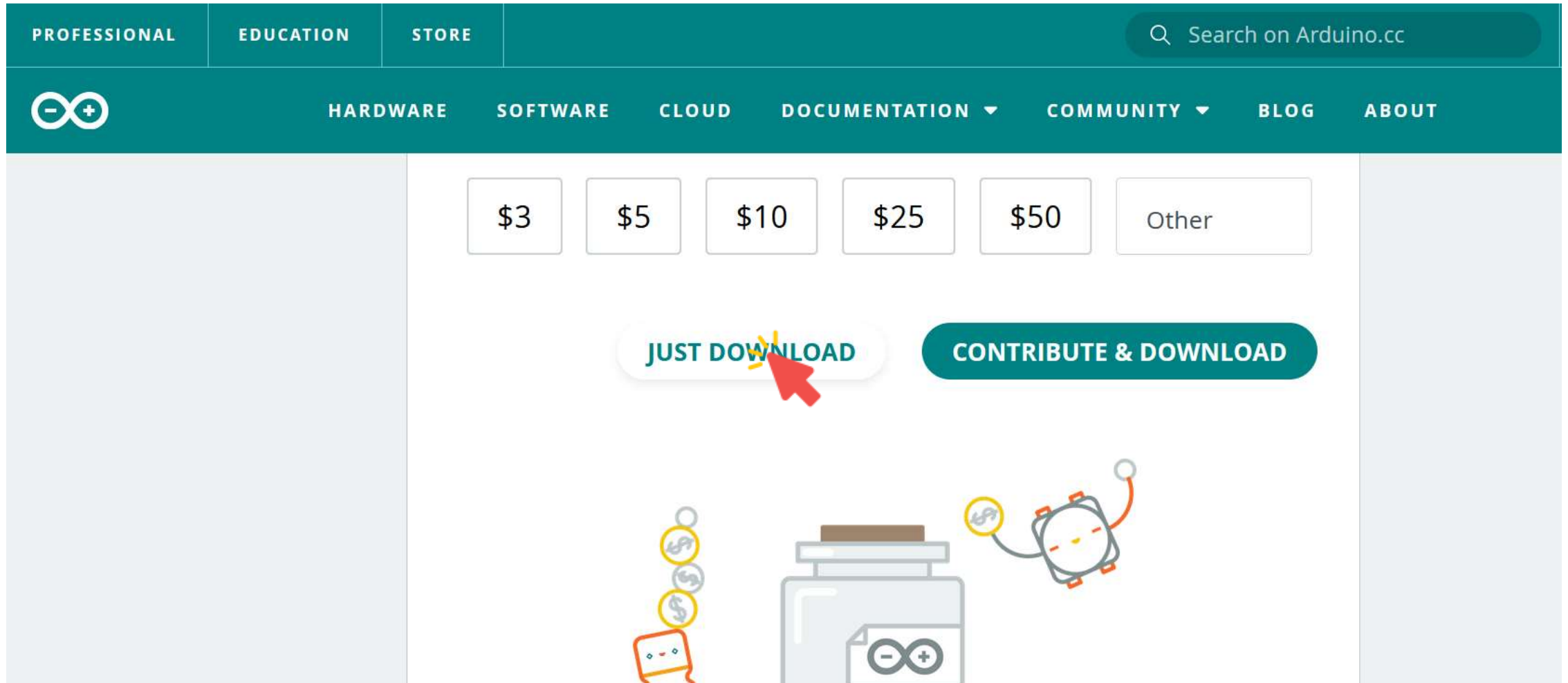
macOS Intel, 10.14: "Mojave" or newer, 64 bits

macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

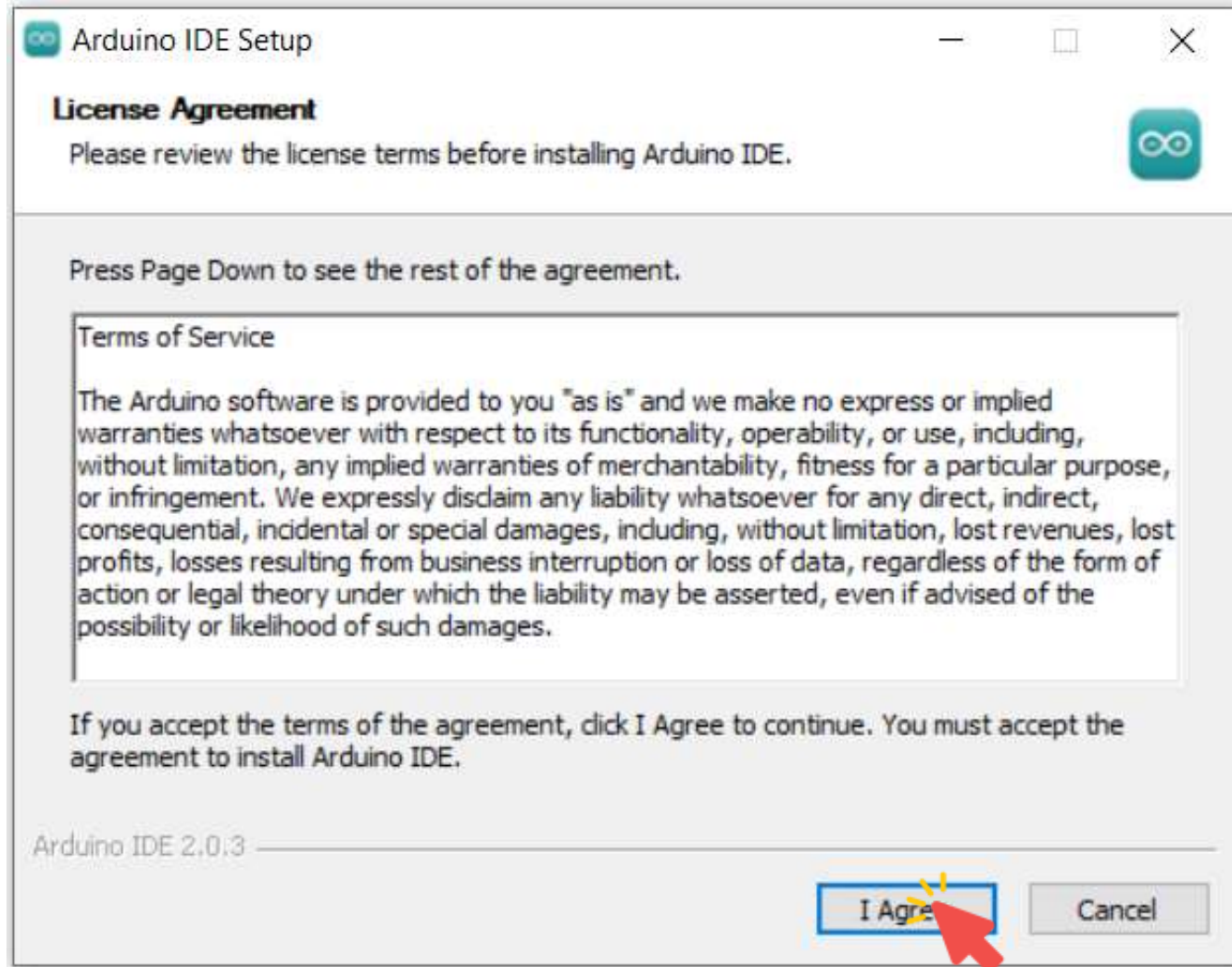
[Release Notes](#)

Downloading Arduino IDE

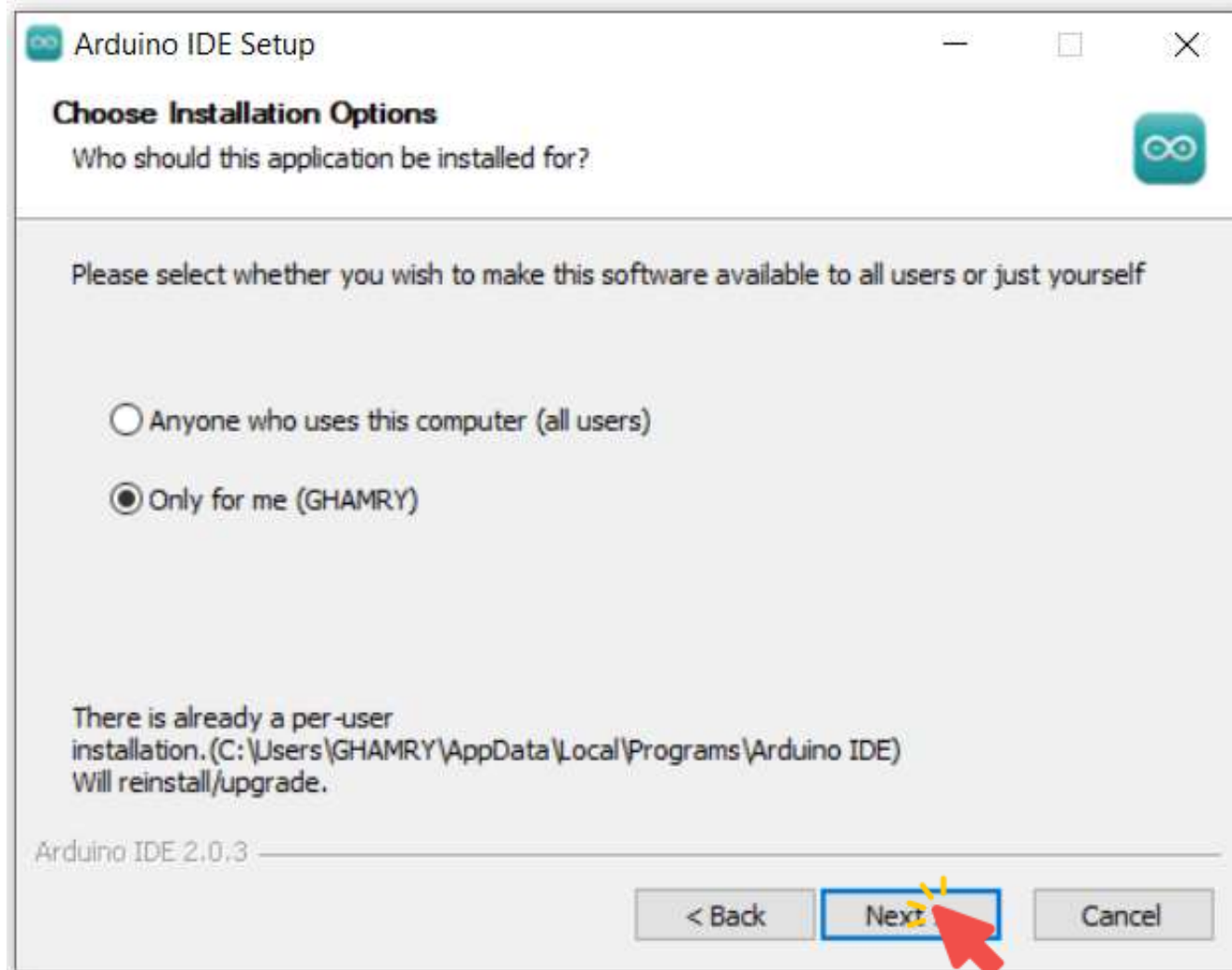
- Click the “Just Download” option.



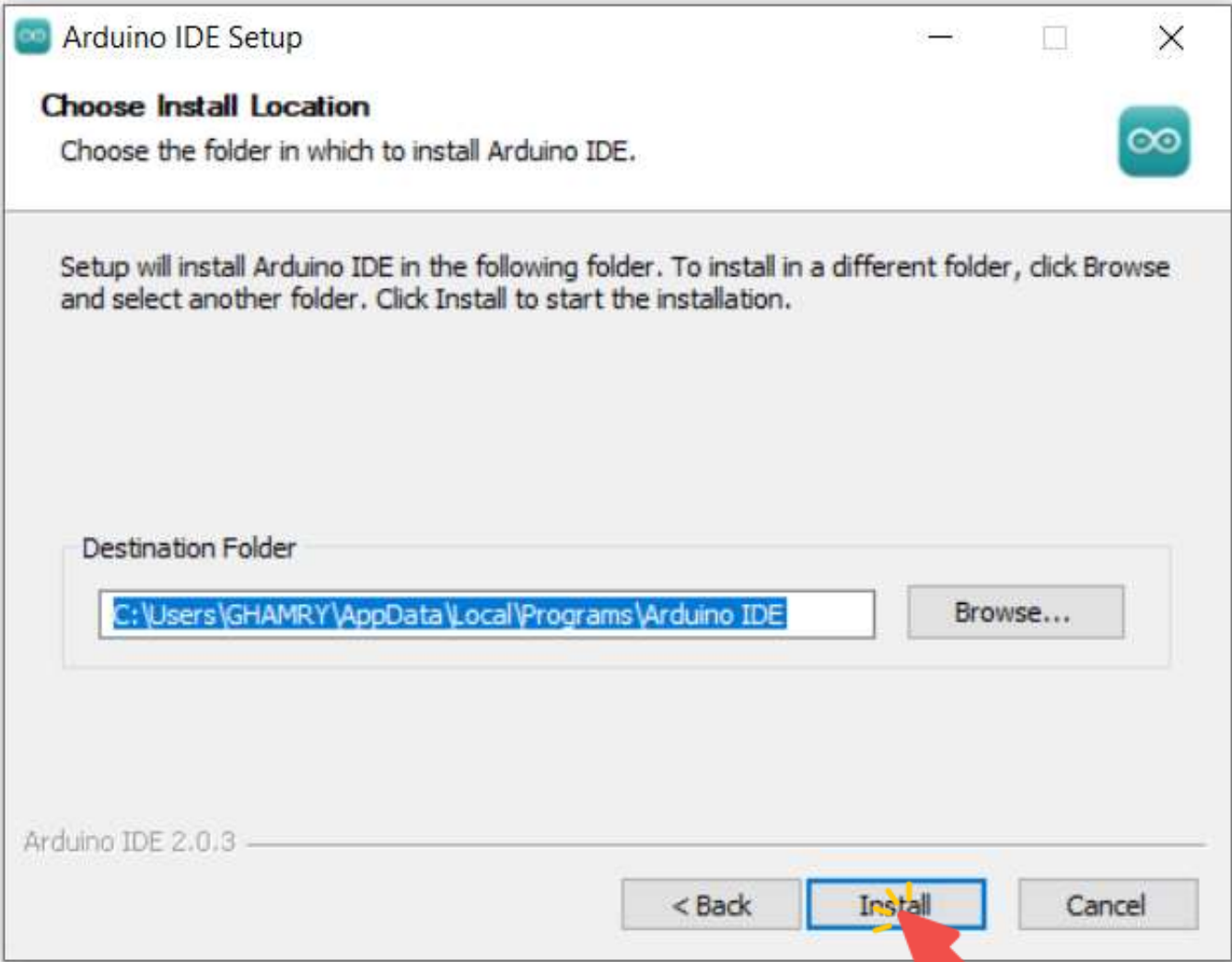
Installing Arduino IDE



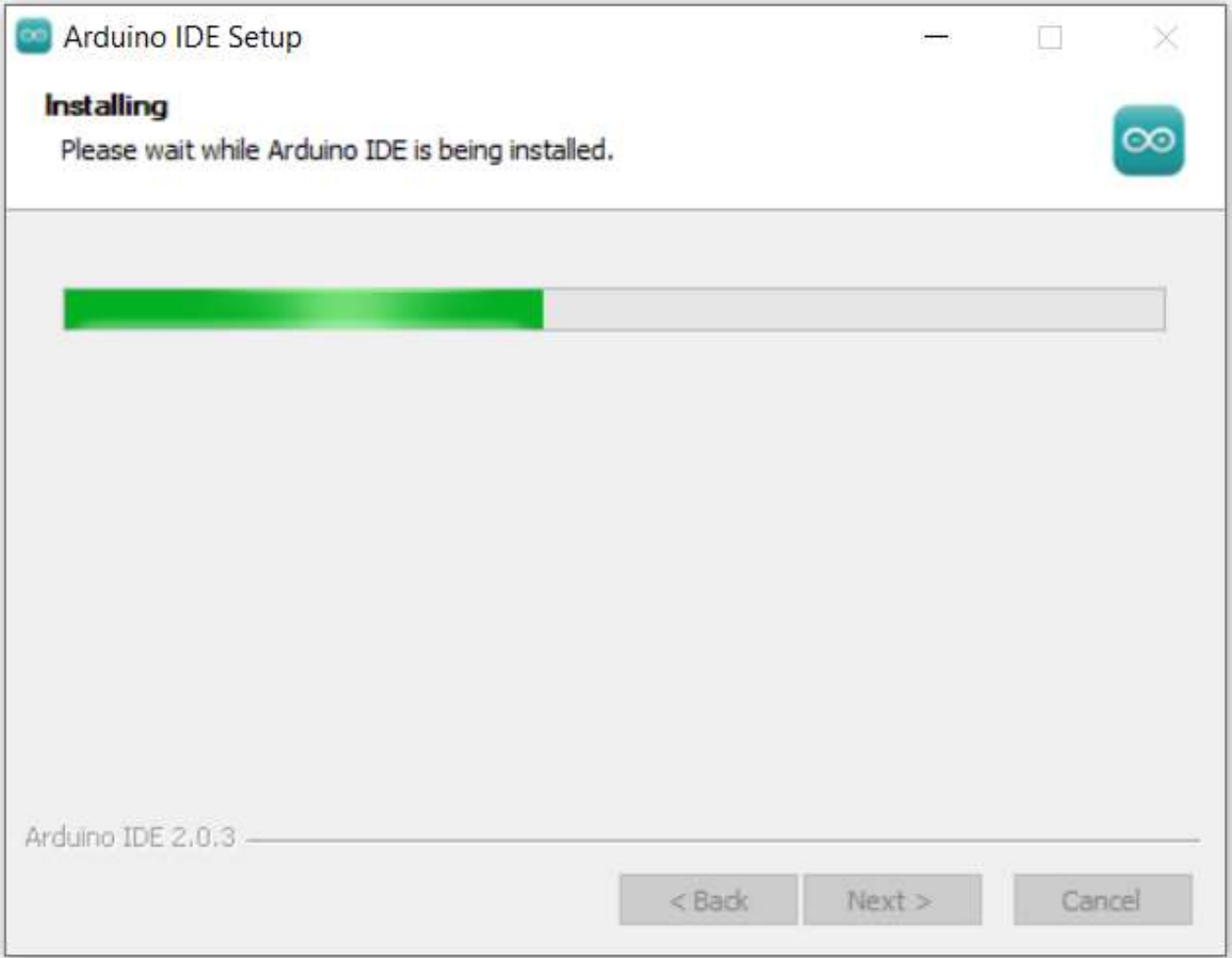
Installing Arduino IDE



Installing Arduino IDE



Installing Arduino IDE



Arduino Sketches

- A **sketch** is the name that Arduino uses for a **program**.

```
void setup() {  
    // put your setup code here, to run once:  
  
}
```

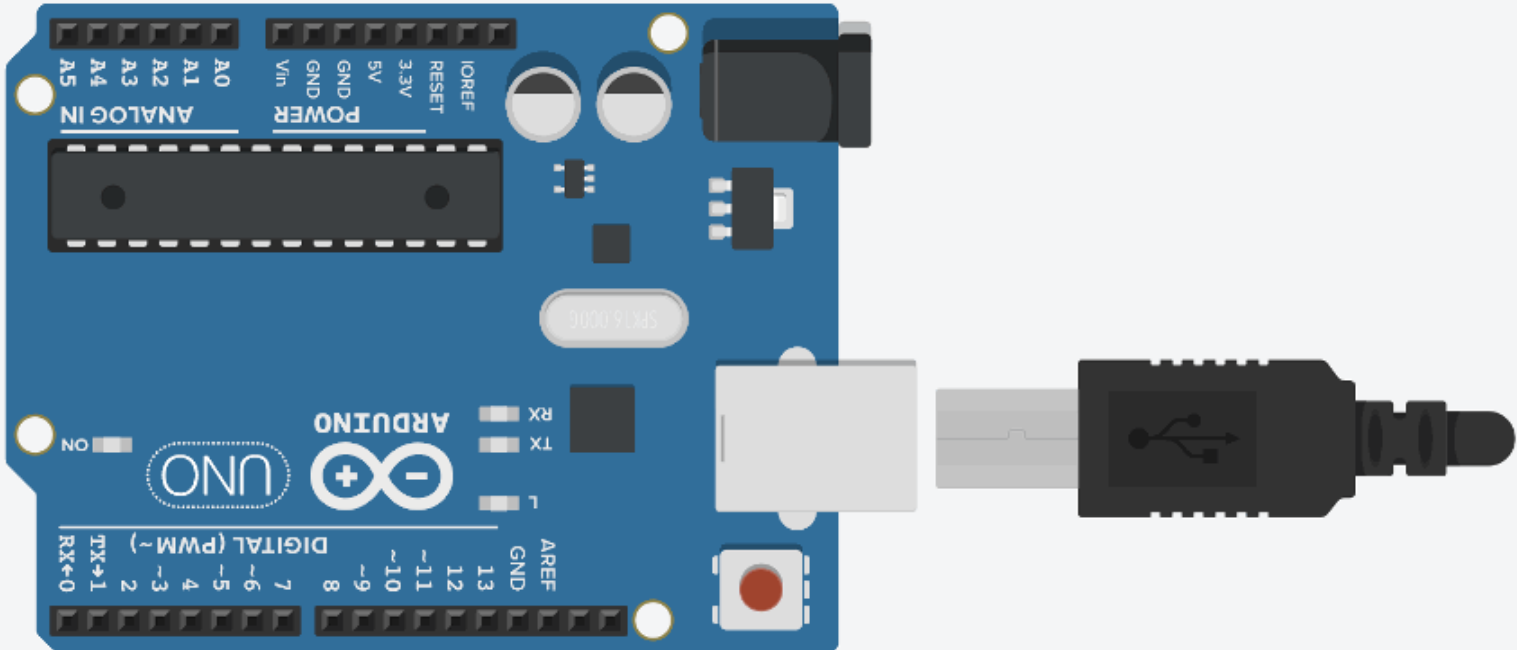
```
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```

Arduino Sketches

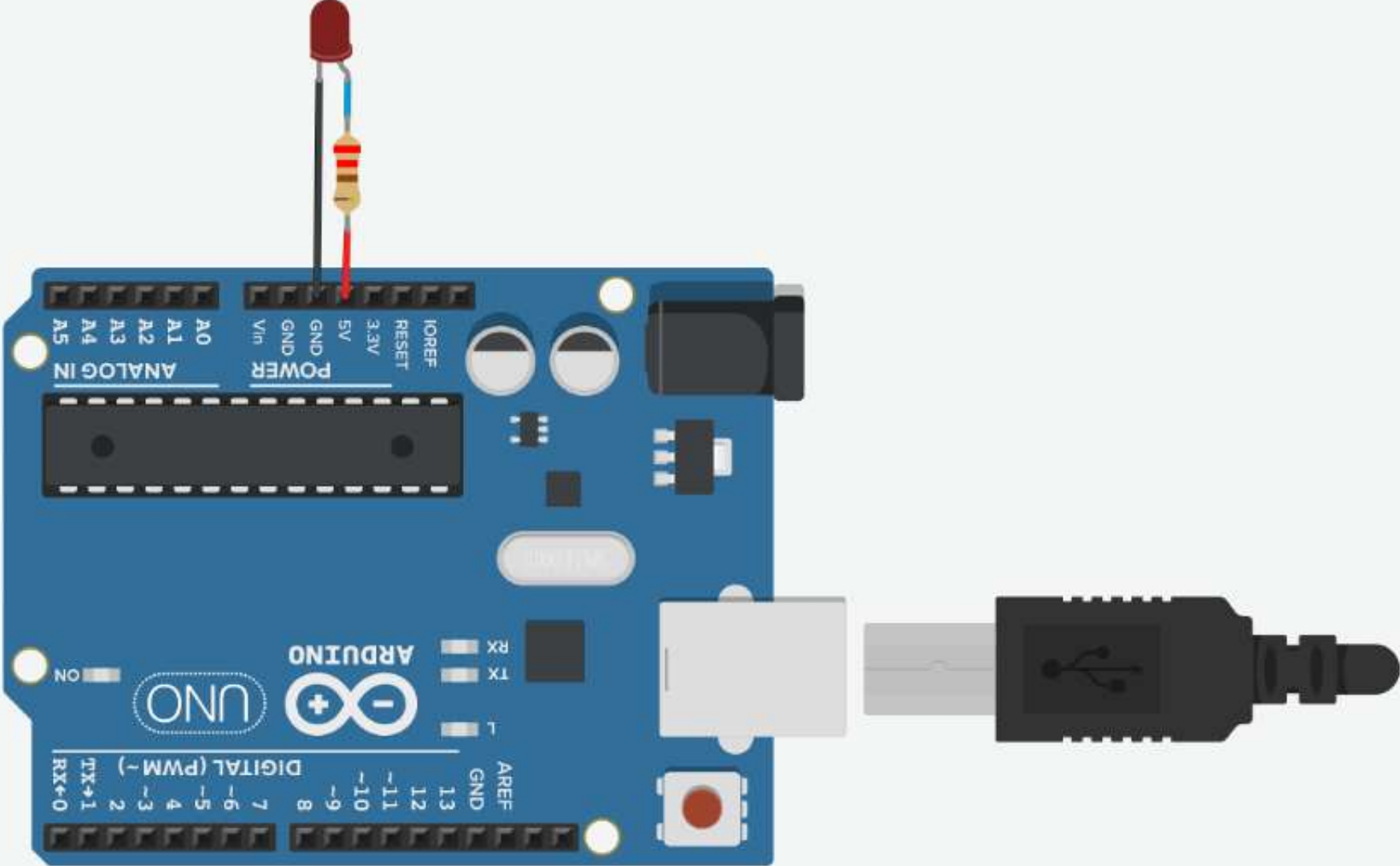
- There are two **special functions** that are a **part of every Arduino sketch**: `setup()` and `loop()`.
- The `setup()` is **called once**, when the sketch starts.
- It's a good place to do **setup tasks** like setting **pin modes**.
- The `loop()` function is **called over and over** and is heart of most sketches.
- You need to **include both functions in your sketch**, even if you don't need them for anything.

Turning on an LED

LED Cathode Towards GND
The LED cathode (short leg or look for flat side of LED casing) connects to GND

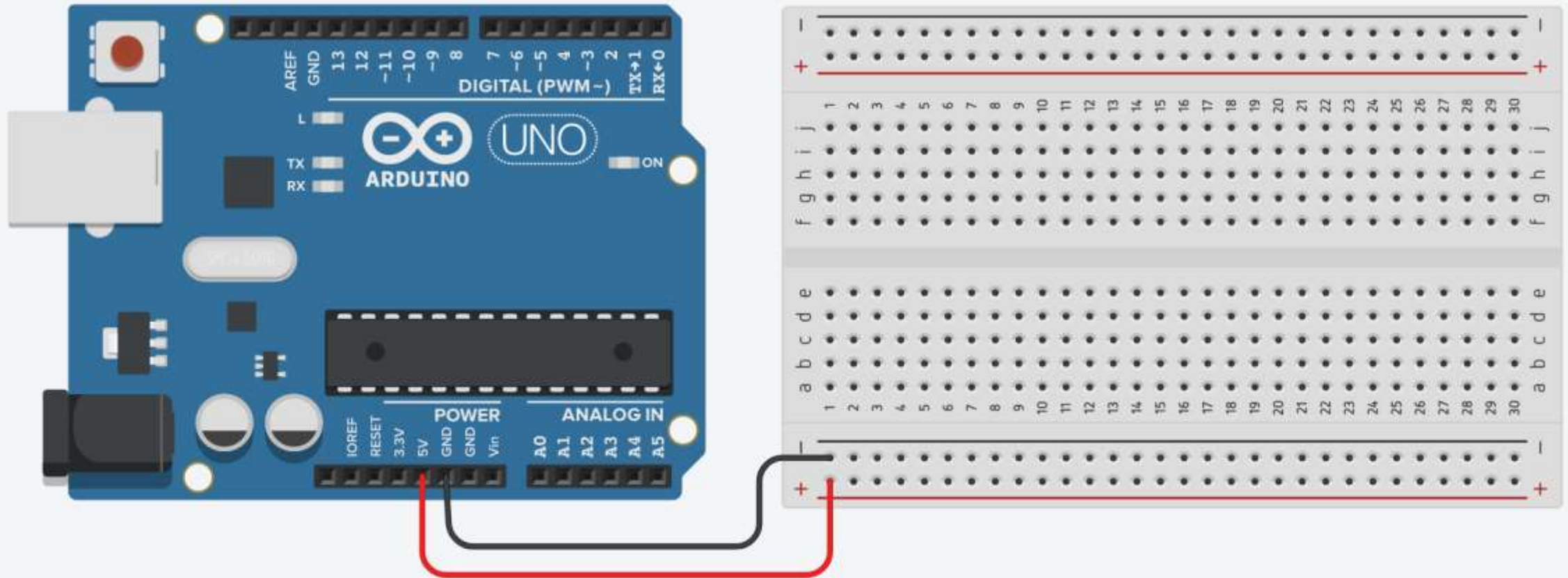


Turning on an LED



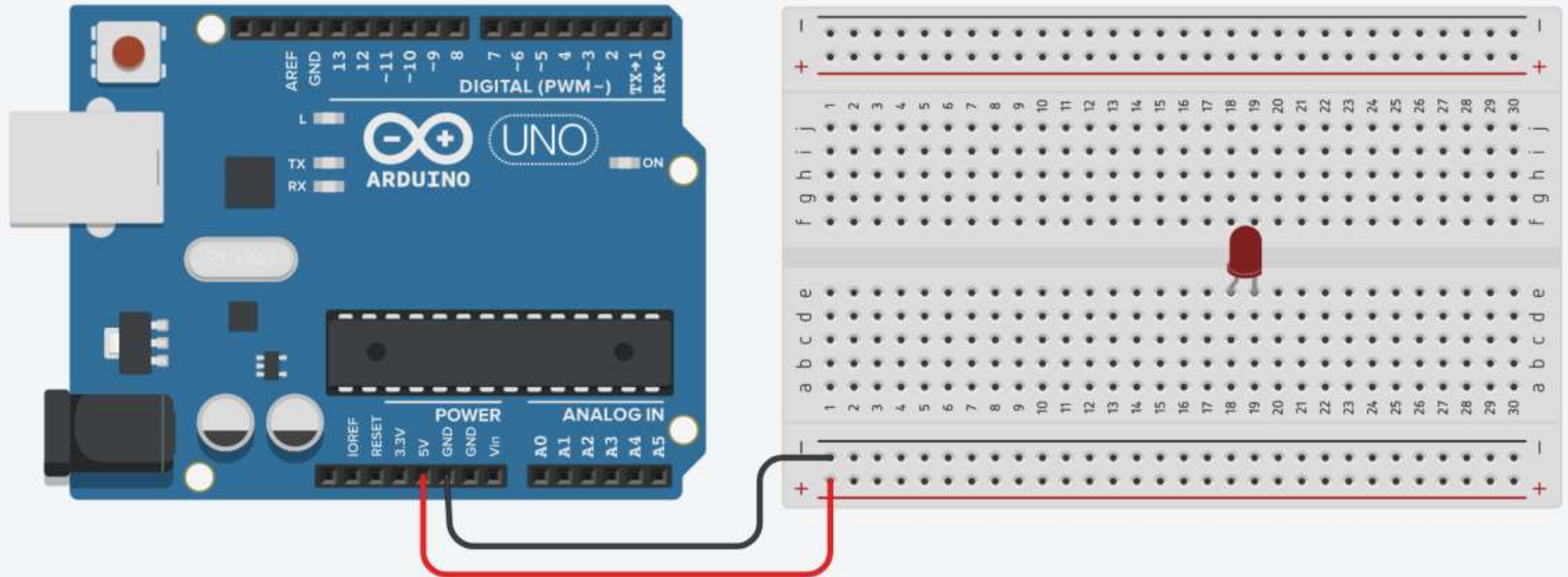
Turning on an LED: Steps

1. Connect breadboard **power (+)** and **ground (-)** rails to Arduino **5V** and **ground (GND)**, respectively.



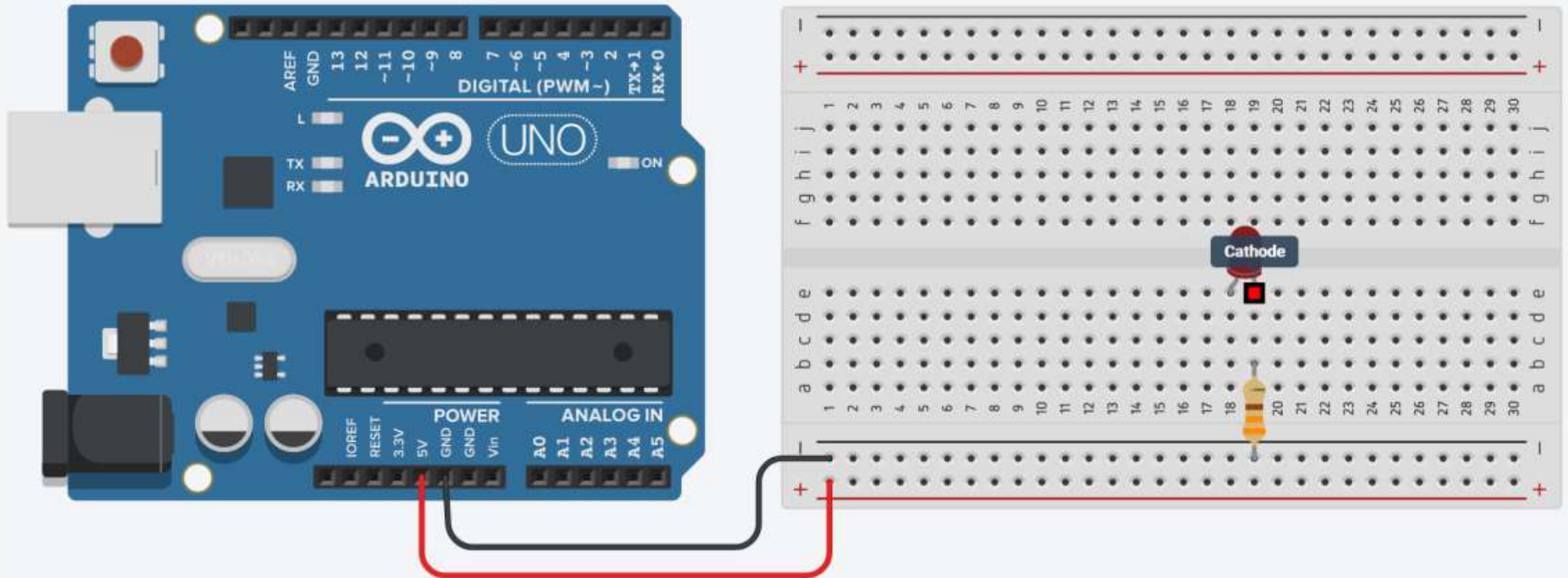
Turning on an LED: Steps

2. Plug the **LED** into two different breadboard rows.



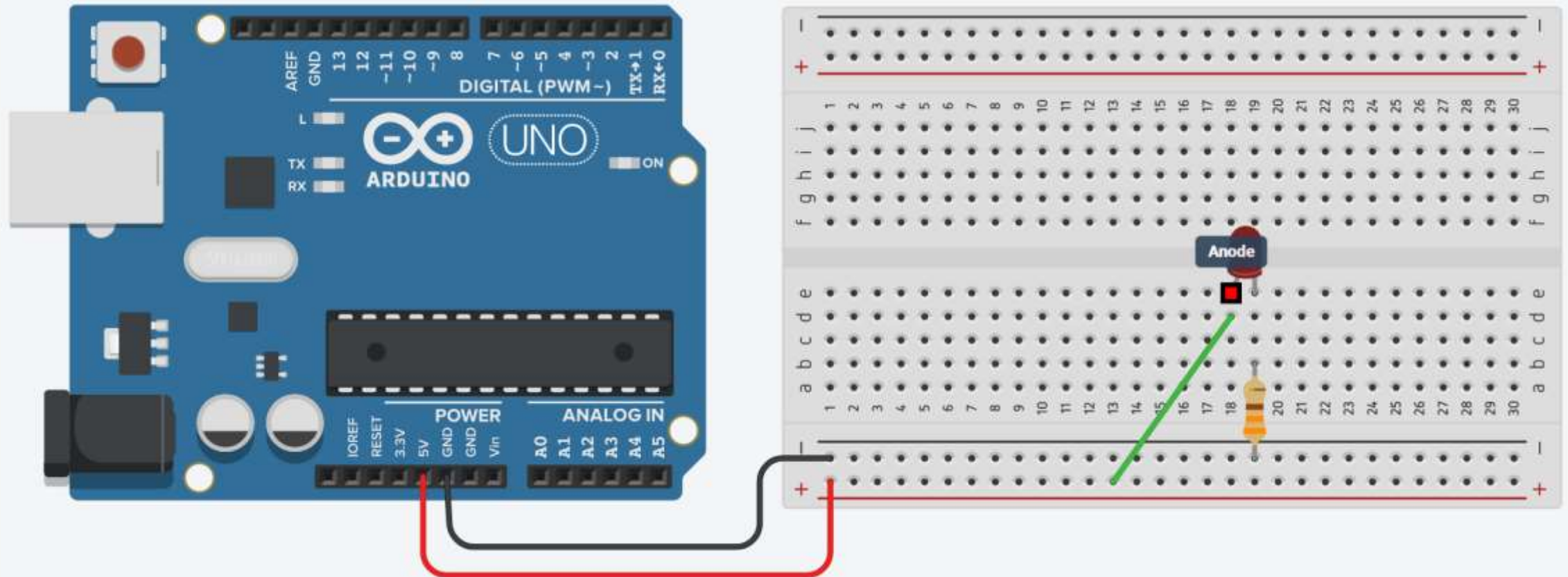
Turning on an LED: Steps

- The **cathode (shorter leg)** connects to one leg of a **resistor of 330Ω** , and the **other resistor leg to the ground**.



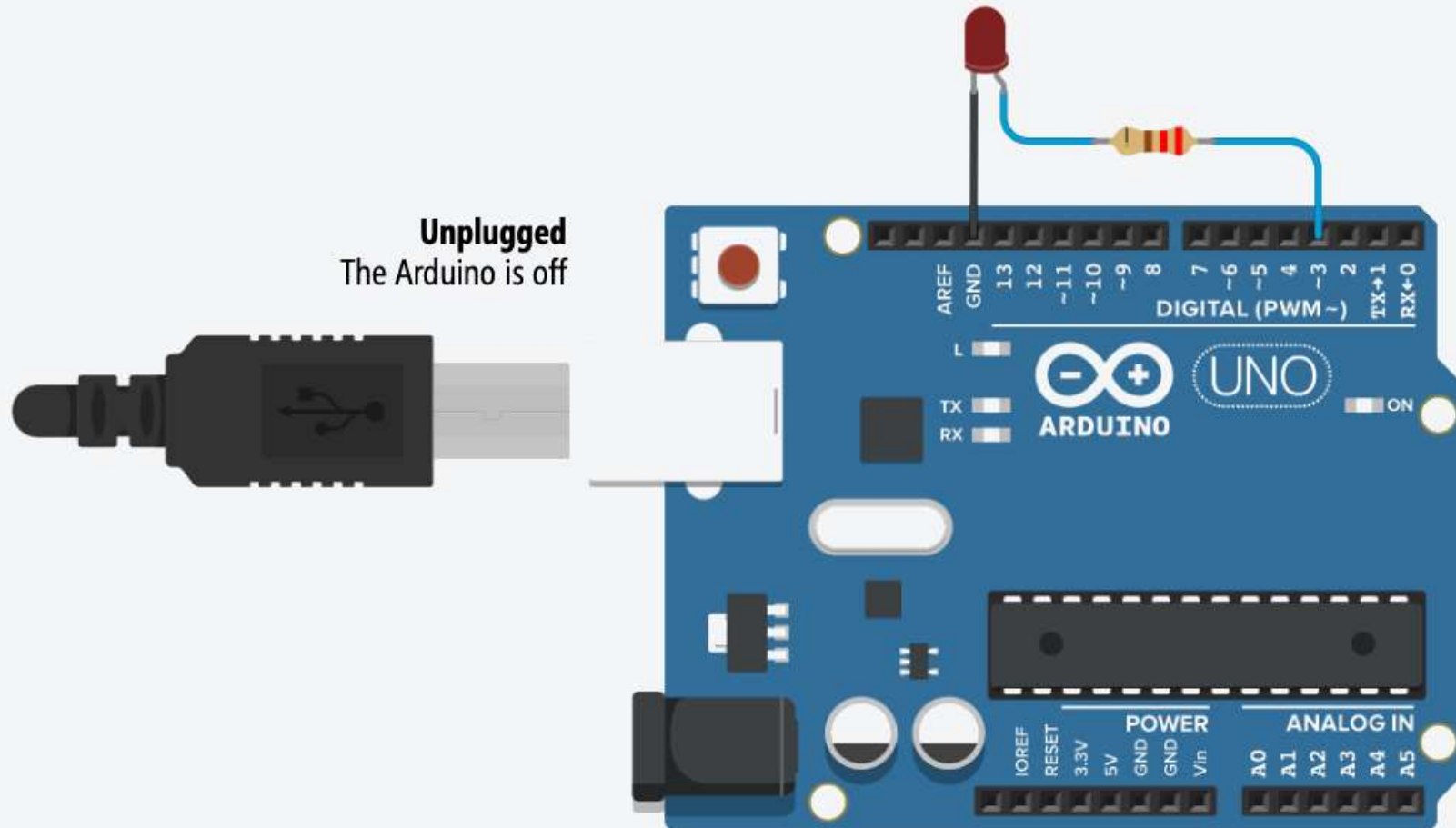
Turning on an LED: Steps

4. Wire up the LED anode (longer leg) to the **power**.



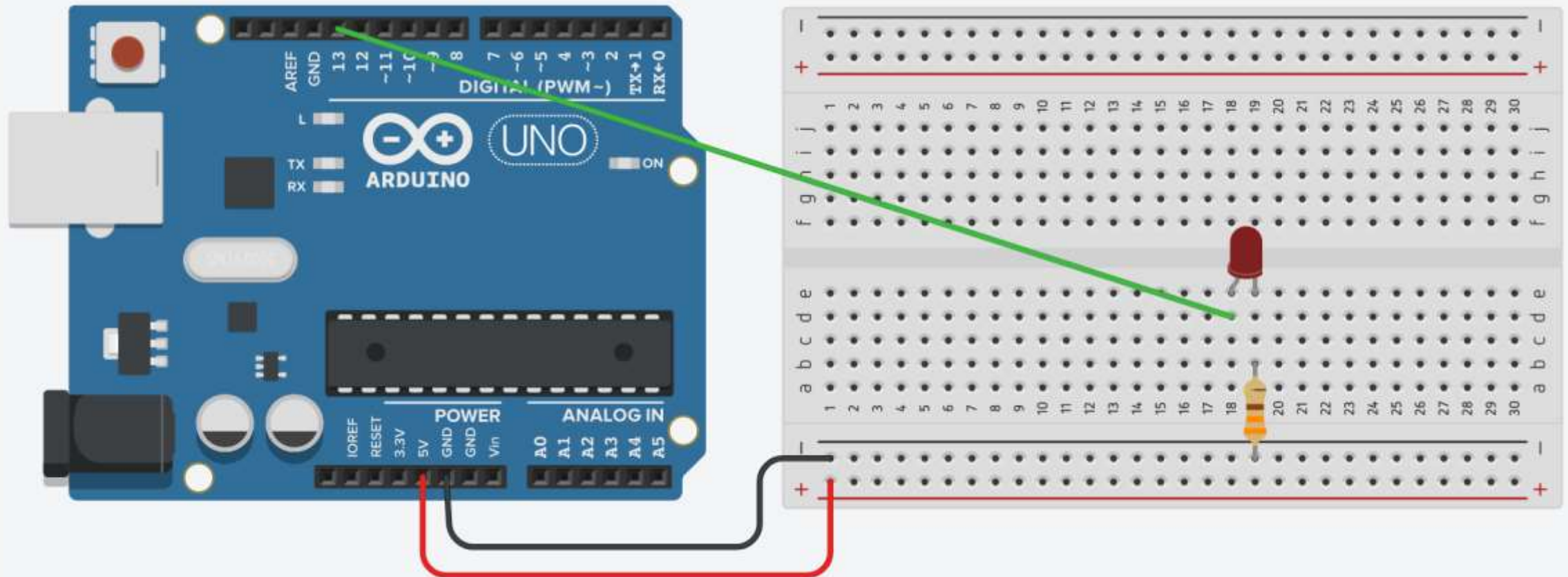
Your First Arduino Project: Blinking an LED

- Turn an LED on and off every second.



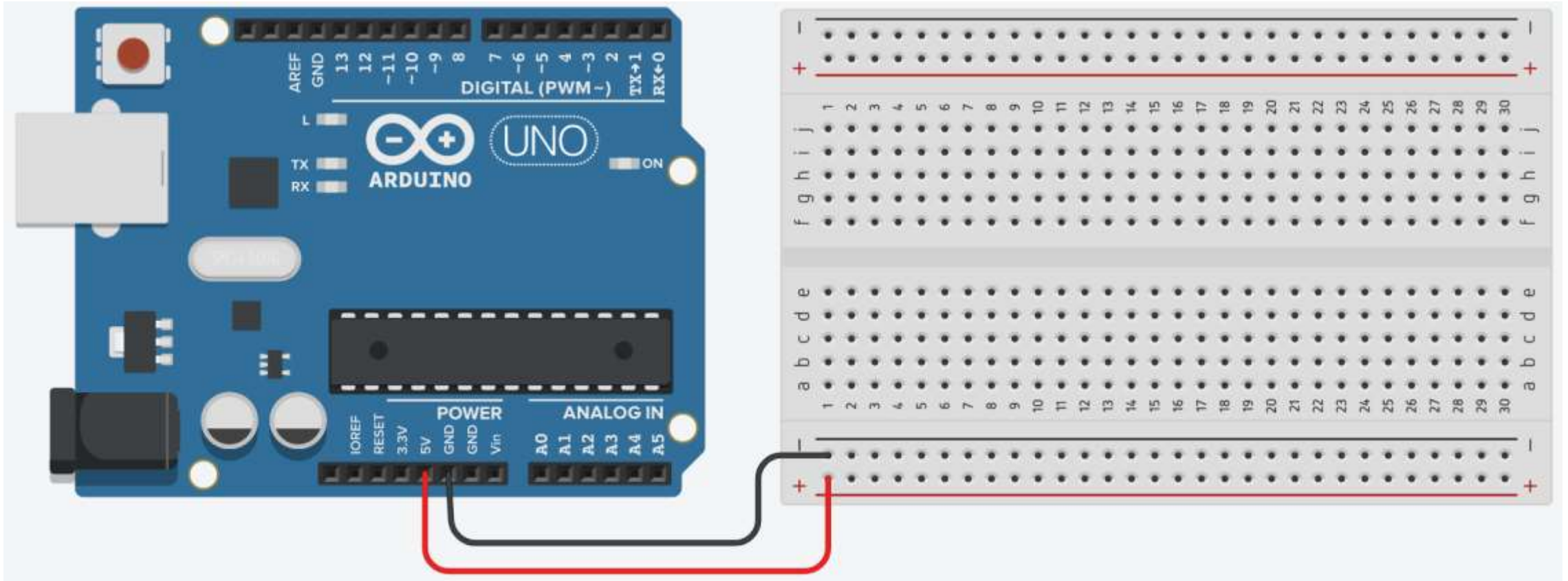
Your First Arduino Project: Circuit

- Turn an LED on and off every second.



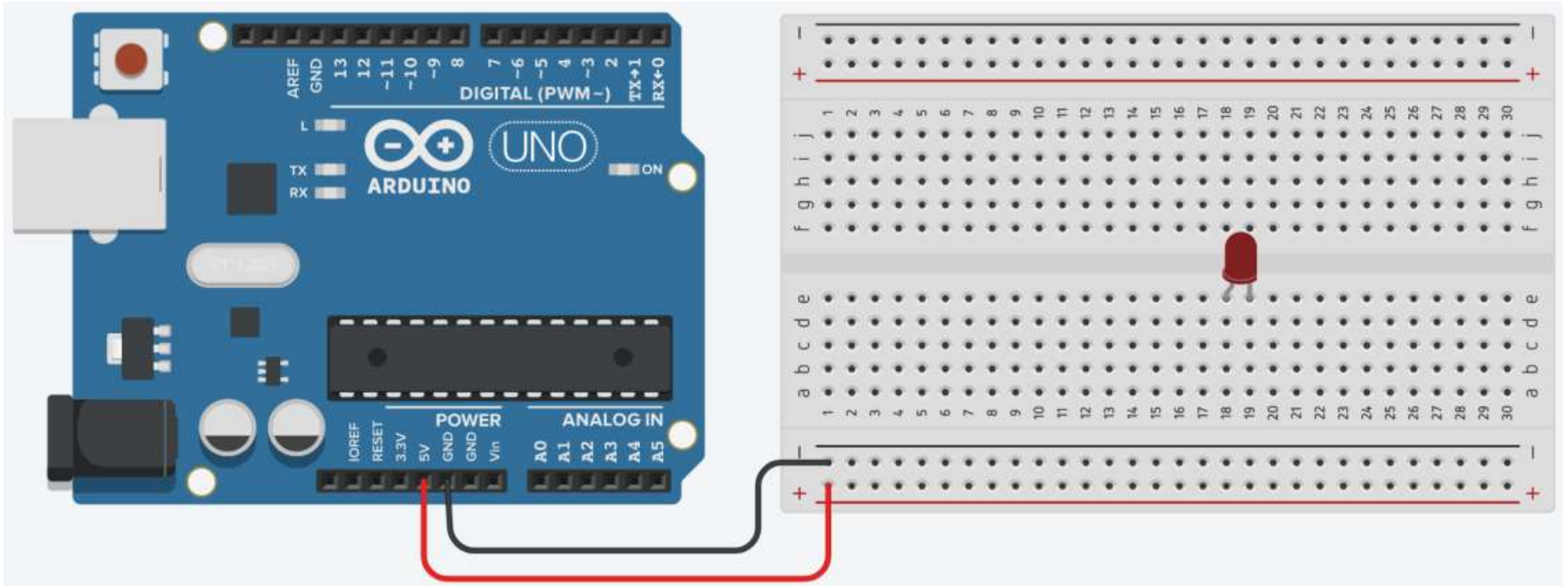
Your First Arduino Project: Steps

1. Connect breadboard **power (+)** and **ground (-)** rails to Arduino **5V** and **ground (GND)**, respectively.



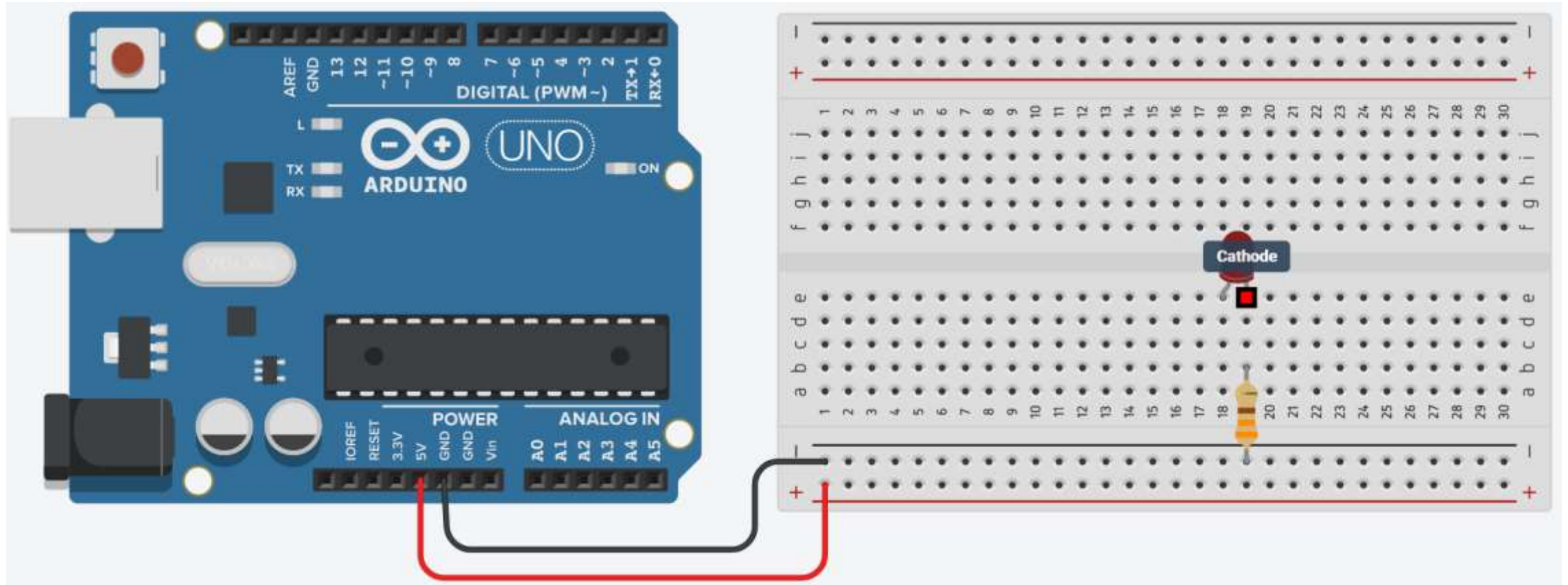
Your First Arduino Project: Steps

2. Plug the **LED** into two different breadboard rows.



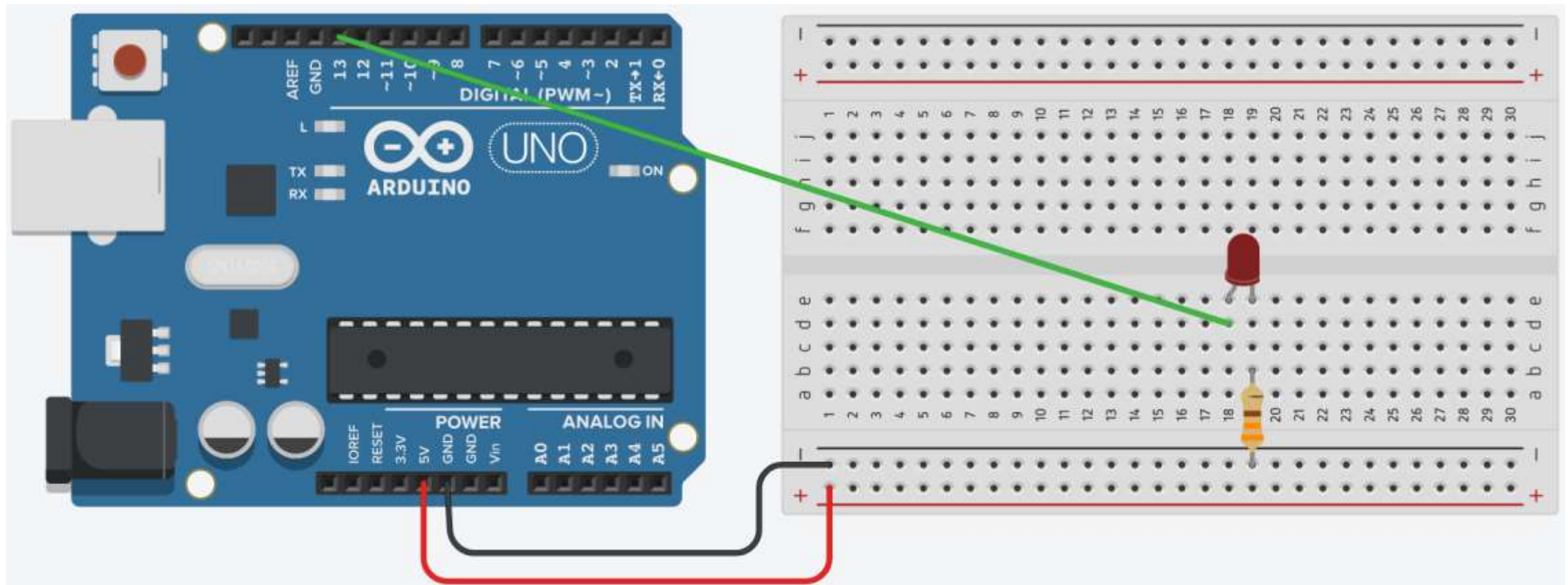
Your First Arduino Project: Steps

- The **cathode (shorter leg)** connects to one leg of a **resistor of 330Ω** , and the **other resistor leg to the ground**.

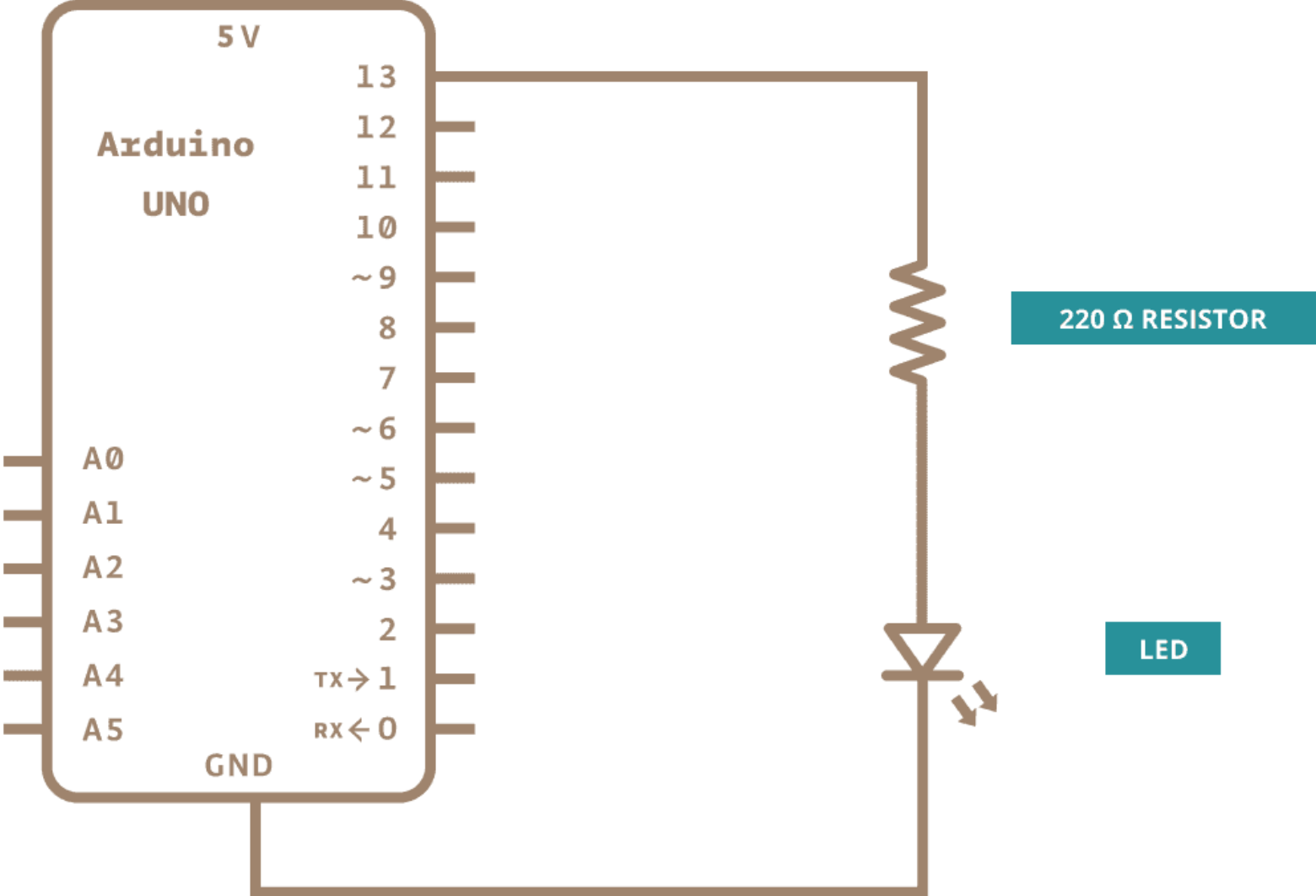


Your First Arduino Project: Steps

4. Wire up the LED anode (longer leg) to Arduino **pin 13**.

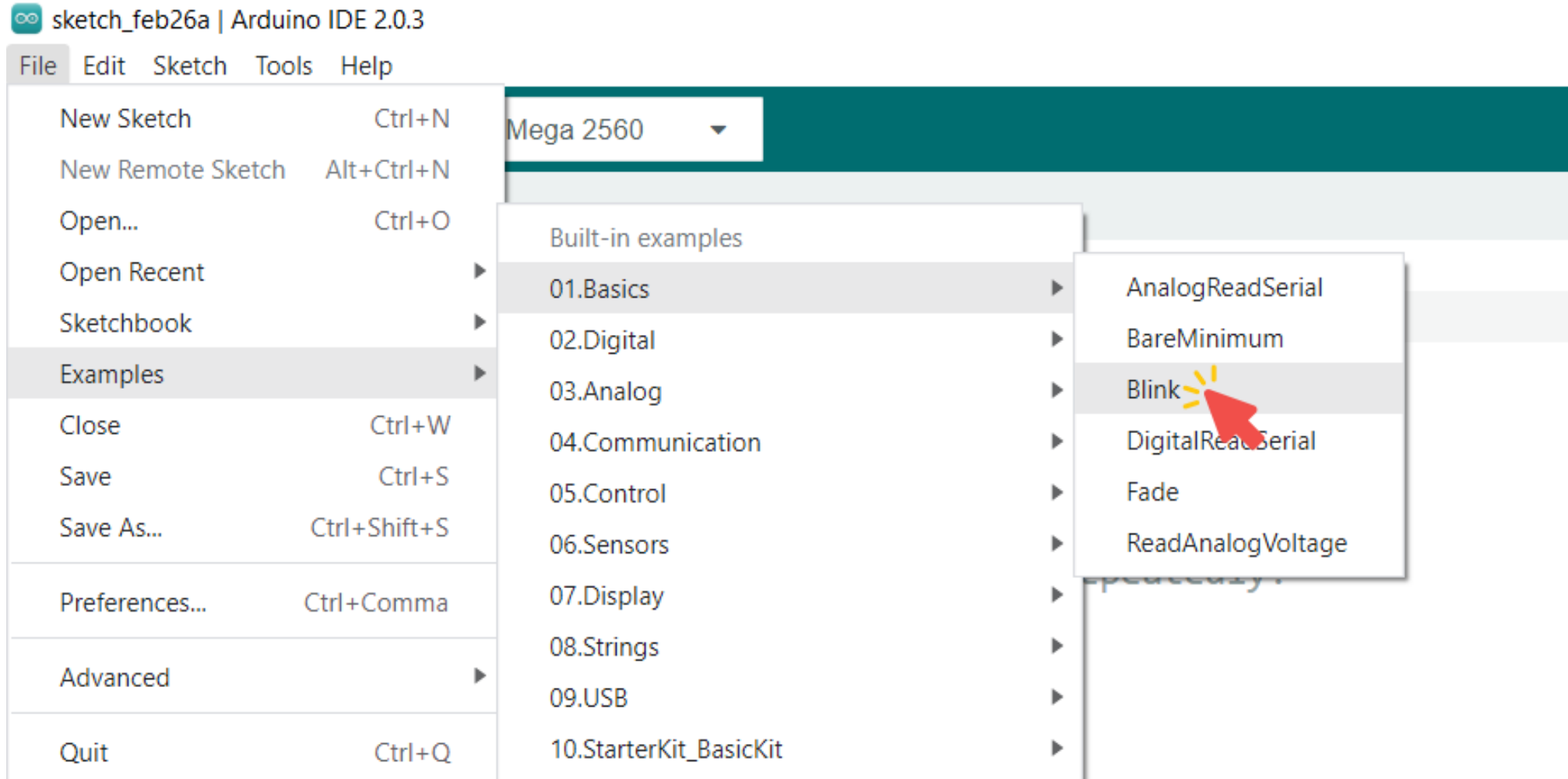


Your First Arduino Project: Schematic



Your First Arduino Project: Blink

You may also load it from **File** → **Examples** → **01.Basics** → **Blink**



Your First Arduino Project: Code

```
// Turns an LED on for one second, then off for one second, repeatedly.

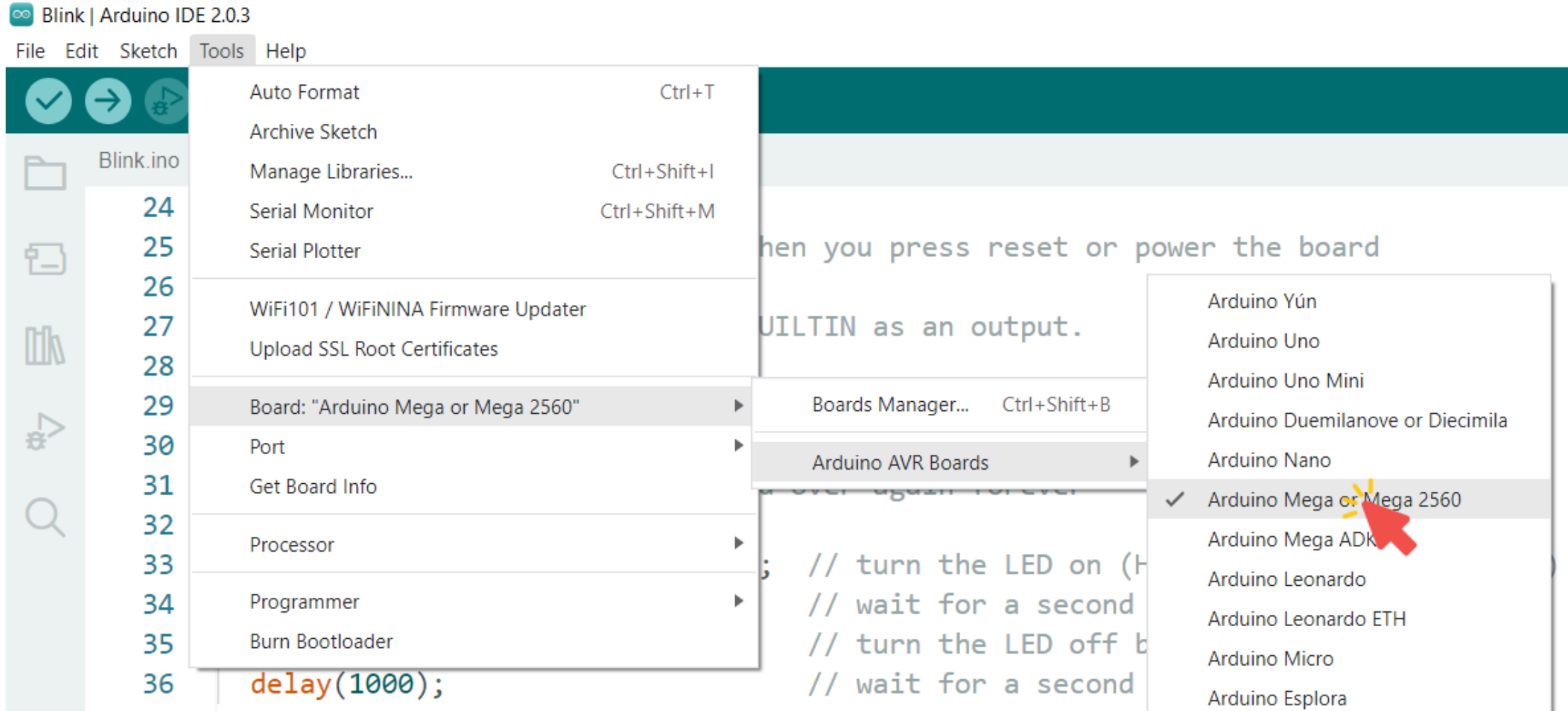
// The setup function runs once when you press reset or power the board
void setup() {
  // Initialize digital pin LED_BUILTIN (13) as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);    // Turn the LED on
  delay(1000);                        // Wait for a second

  digitalWrite(LED_BUILTIN, LOW);    // Turn the LED off
  delay(1000);                        // Wait for a second
}
```

Your First Arduino Project: Arduino AVR Boards

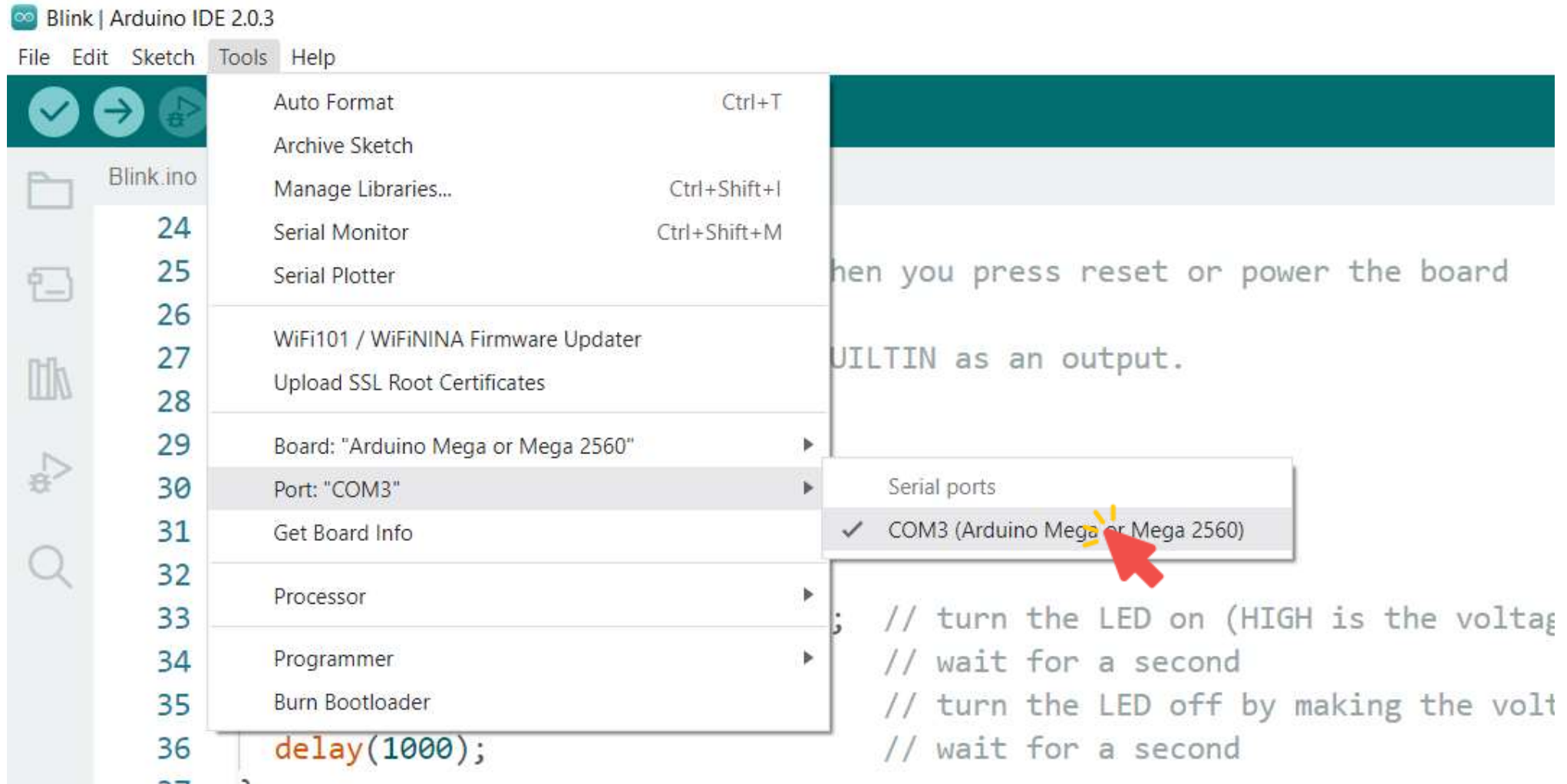
Go to **Tools** → **Board**, and select **your board**.



The screenshot shows the Arduino IDE 2.0.3 interface. The 'Tools' menu is open, and the 'Board' option is selected, which has opened a submenu. In the submenu, 'Arduino AVR Boards' is selected, which has opened a third-level menu. In this third-level menu, 'Arduino Mega or Mega 2560' is selected and highlighted with a red arrow. The main menu items are: Auto Format (Ctrl+T), Archive Sketch, Manage Libraries... (Ctrl+Shift+I), Serial Monitor (Ctrl+Shift+M), Serial Plotter, WiFi101 / WiFININA Firmware Updater, Upload SSL Root Certificates, Board: "Arduino Mega or Mega 2560", Port, Get Board Info, Processor, Programmer, and Burn Bootloader. The Boards Manager... option (Ctrl+Shift+B) is also visible in the submenu. The background shows a code editor with a sketch named 'Blink.ino' and some code lines, including a `delay(1000);` statement.

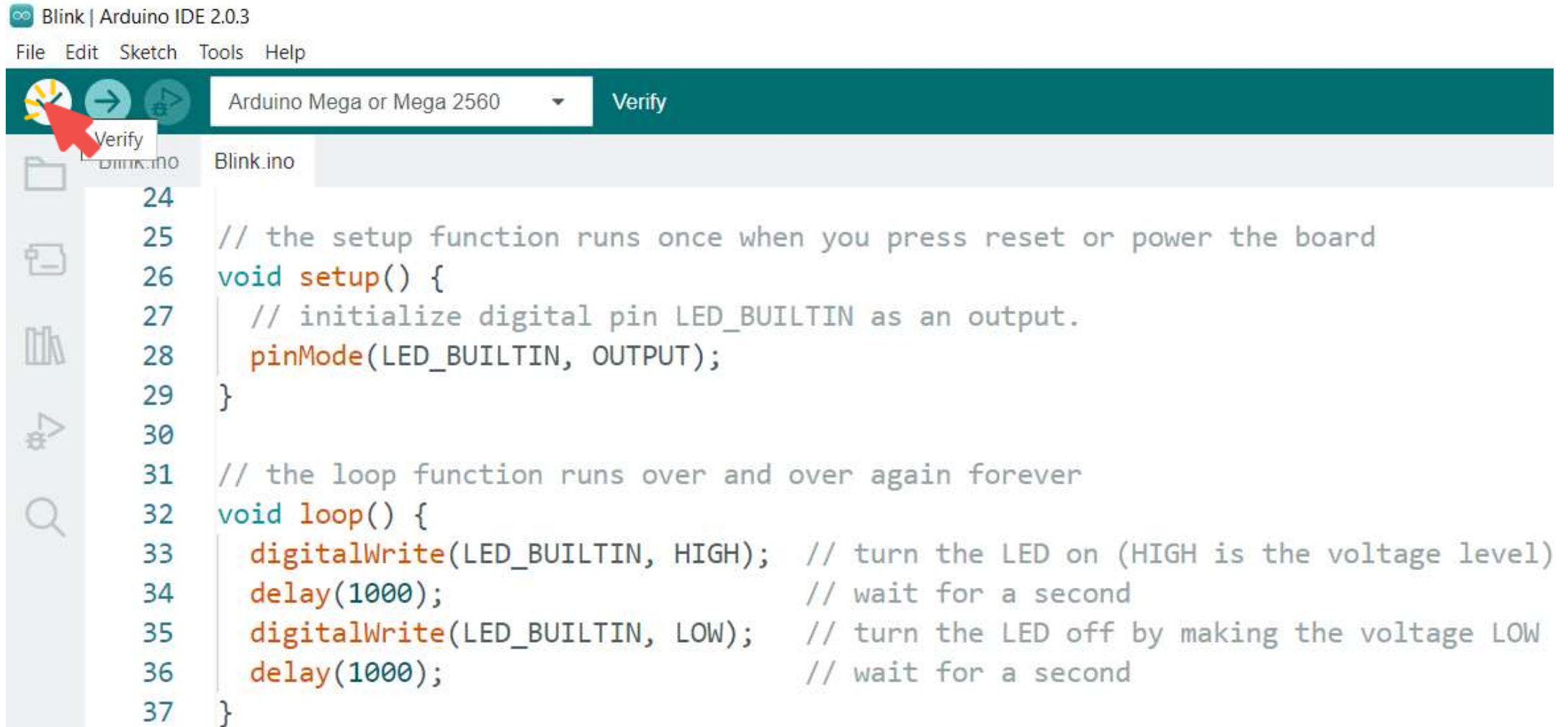
Your First Arduino Project: Port

Go to **Tools** → **Port**, and **select the port** of the Arduino board.



Your First Arduino Project: Verify a Sketch

Click the **Verify** button to **try compiling the sketch and check for errors.**

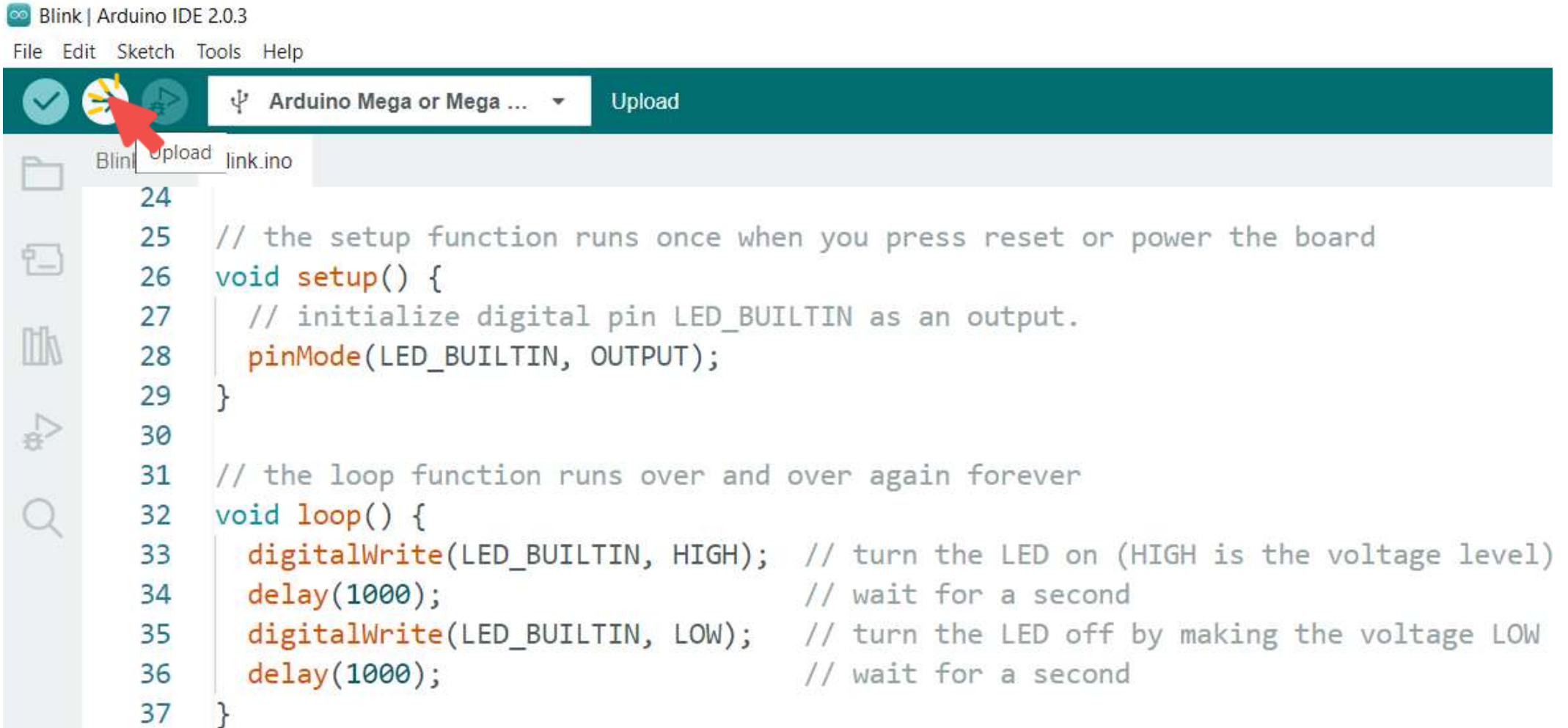


The screenshot shows the Arduino IDE 2.0.3 interface. The title bar reads "Blink | Arduino IDE 2.0.3". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for "Verify" (a yellow lightning bolt), "Upload" (a right-pointing arrow), and "Download" (a left-pointing arrow). A dropdown menu is open, showing "Arduino Mega or Mega 2560" and a "Verify" button. A red arrow points to the "Verify" button in the toolbar. The main editor window displays the following code:

```
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34     delay(1000); // wait for a second
35     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36     delay(1000); // wait for a second
37 }
```

Your First Arduino Project: Upload a Sketch

Click the **Upload** button to **program the board** with the sketch.



The screenshot shows the Arduino IDE 2.0.3 interface. The title bar reads "Blink | Arduino IDE 2.0.3". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains several icons: a checkmark, a USB symbol, a play button, and a dropdown menu currently set to "Arduino Mega or Mega ...". The "Upload" button, represented by a USB symbol, is highlighted with a red arrow. Below the toolbar, the file explorer shows "Blink" and "link.ino". The main editor area displays the following C++ code:

```
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34     delay(1000); // wait for a second
35     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36     delay(1000); // wait for a second
37 }
```

Your First Arduino Project: Discussion

- The first thing you do is to initialize `LED_BUILTIN` pin as an **output pin** with the line:

```
pinMode(LED_BUILTIN, OUTPUT);
```

- In the main loop, you turn the LED on with the line:

```
digitalWrite(LED_BUILTIN, HIGH);
```

- Then you turn it off with the line:

```
digitalWrite(LED_BUILTIN, LOW);
```

Your First Arduino Project: Discussion

- The `delay()` causes the Arduino to wait for the specified number of milliseconds before continuing on to the next line.
- There are 1000 milliseconds in a second, so the following line creates a delay of one second.
`delay(1000);`
- Constants are used to make the programs easier to read.
- The constant `LED_BUILTIN` is the number of the pin to which the on-board LED is connected.
- Most boards have this LED connected to digital pin 13.

Your First Arduino Project: Alternative Code

```
// Turns an LED on for one second, then off for one second, repeatedly.

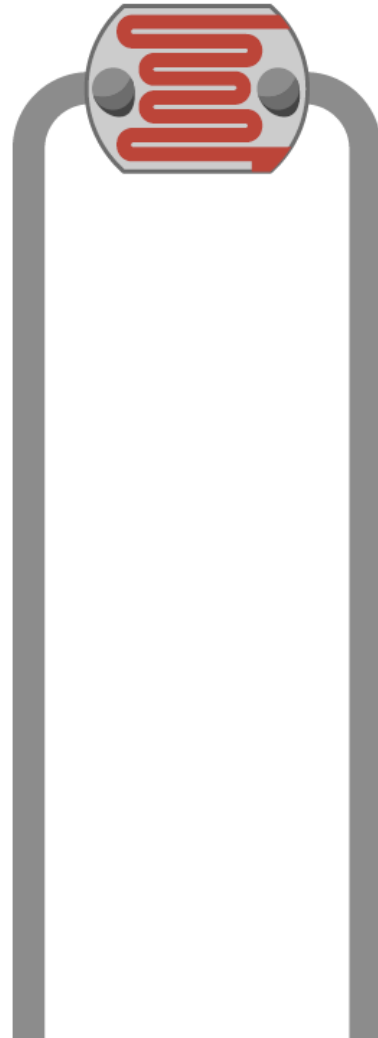
// The setup function runs once when you press reset or power the board
void setup() {
  // Initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);      // Turn the LED on
  delay(1000);                 // Wait for a second

  digitalWrite(13, LOW);      // Turn the LED off
  delay(1000);                 // Wait for a second
}
```

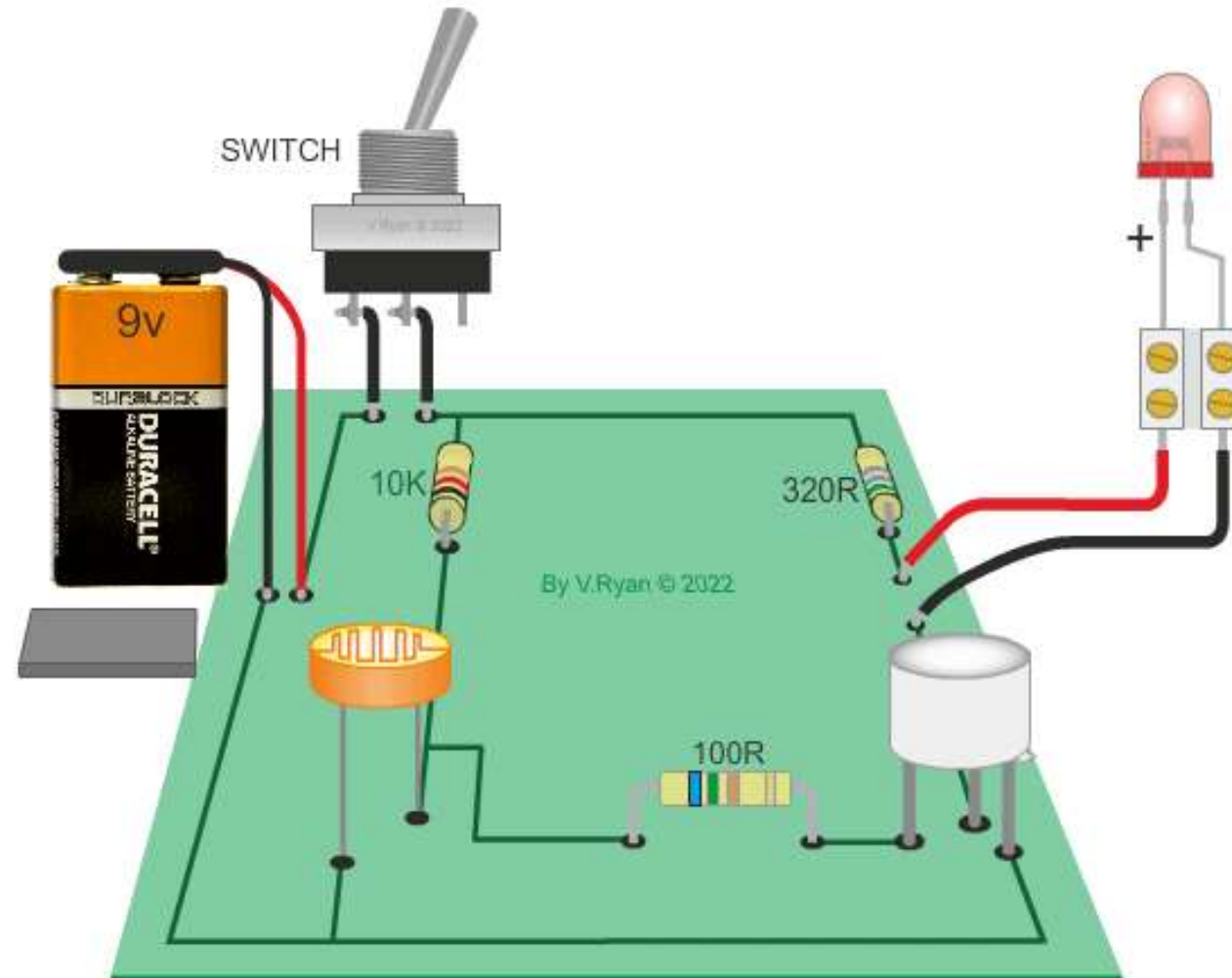
Photoresistor (Light Sensor)

- The photoresistor is a **lightsensitive**, variable resistor.

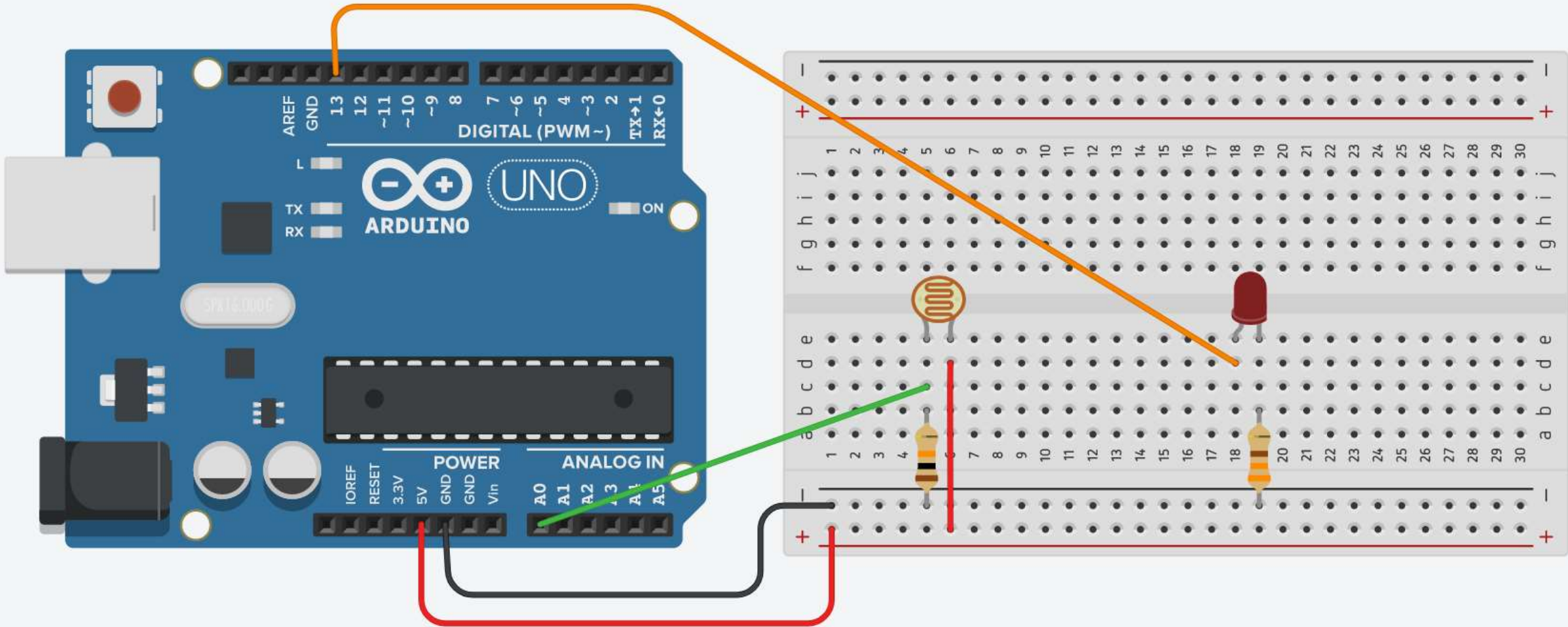


Photoresistor (Light Sensor)

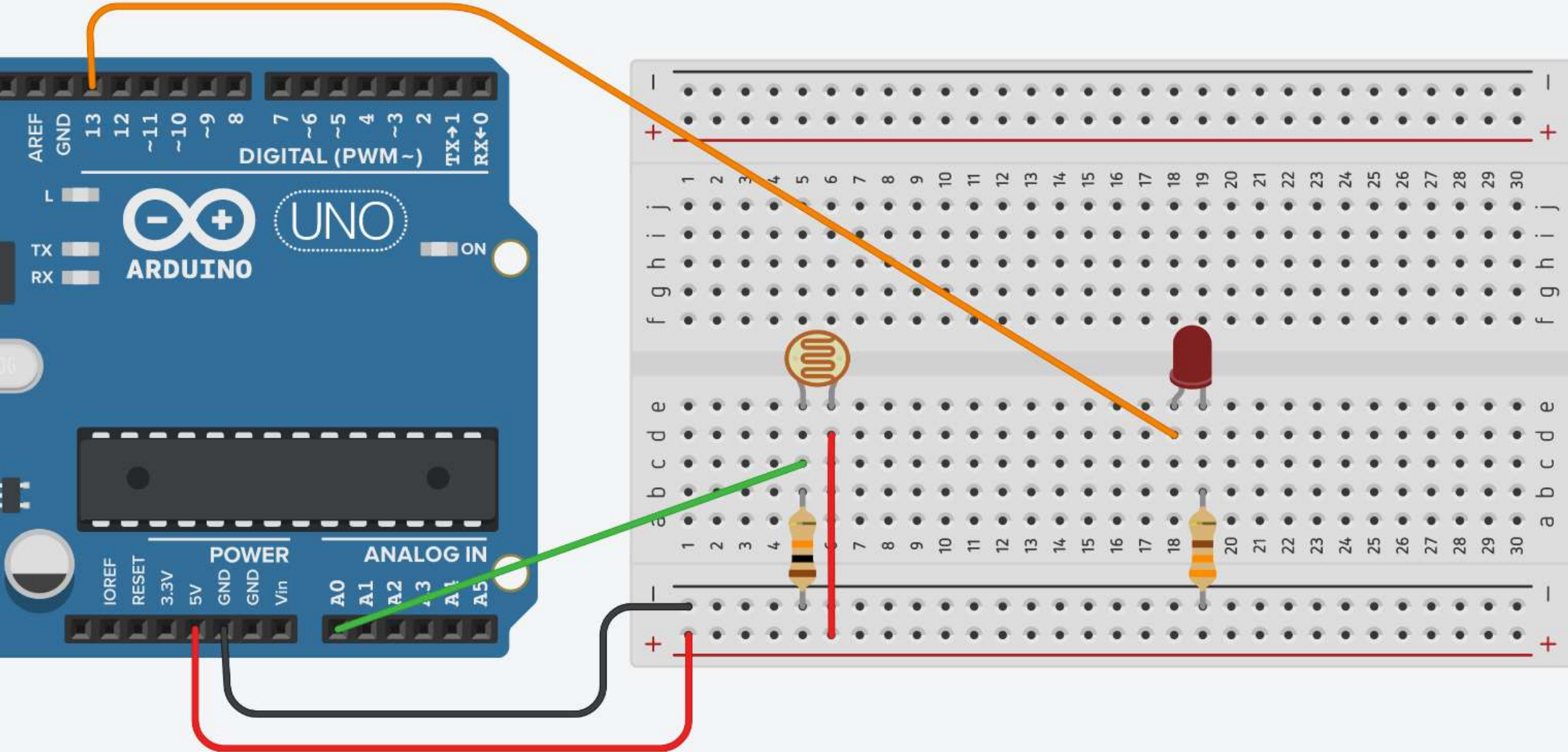
- Photoresistors are perfect for making **light controlled switches**.



Photoresistor: Circuit



Photoresistor: Circuit

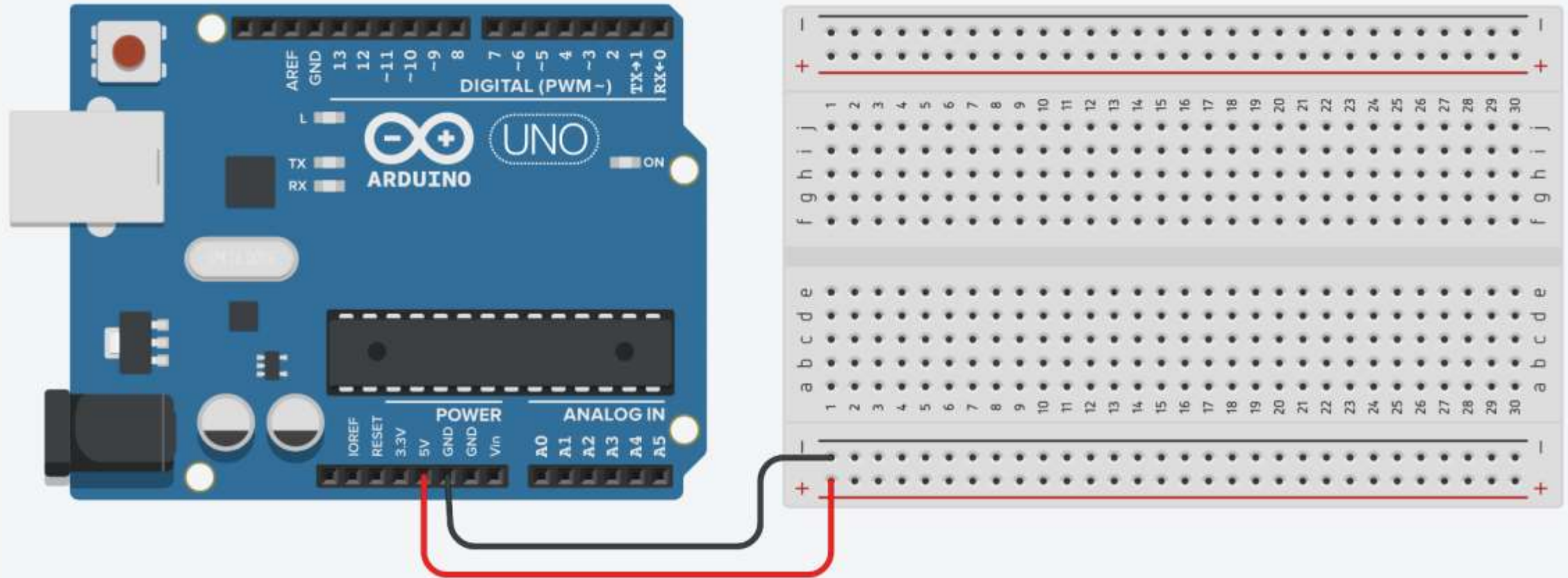


Photoresistor: Components

- You need
 - Arduino
 - LED
 - Photoresistor
 - 330 Ω Resistor
 - 10K Ω Resistor
 - Jumpers
 - Breadboard

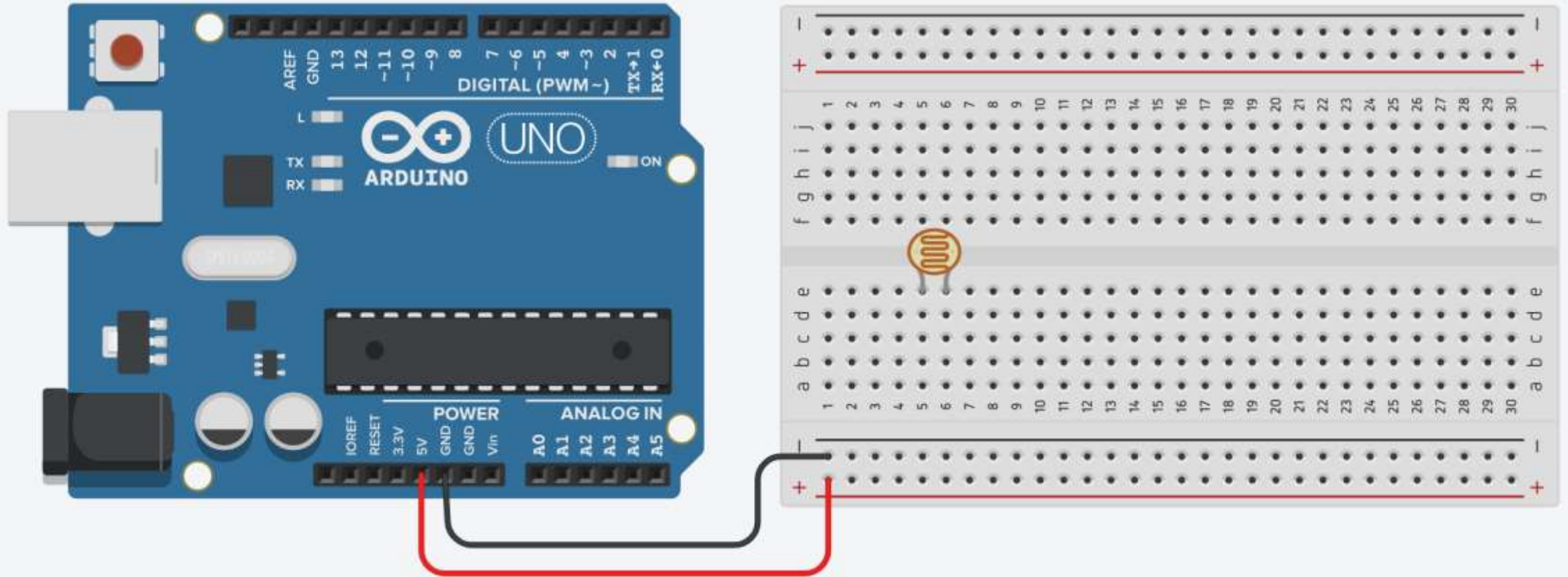
Photoresistor: Steps

1. Connect breadboard **power (+)** and **ground (-)** rails to Arduino **5V** and **ground (GND)**, respectively.



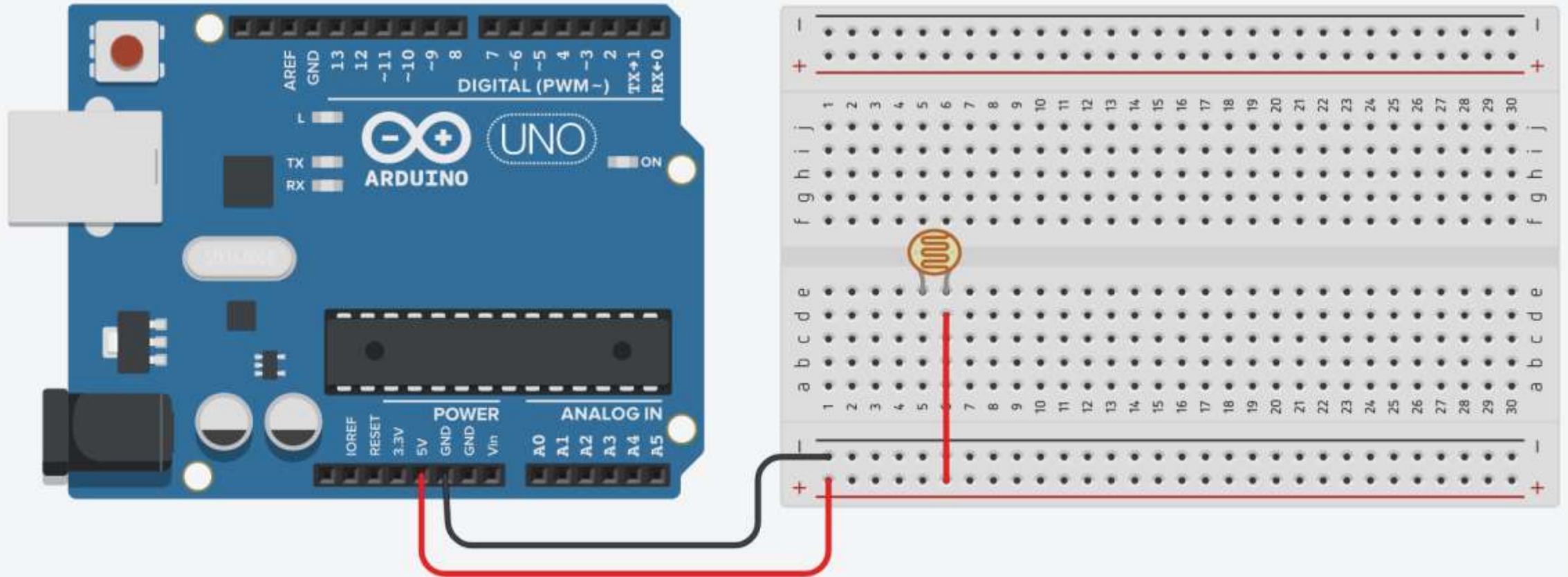
Photoresistor: Steps

2. Drag a photoresistor to your breadboard, so its legs plug into two different rows.



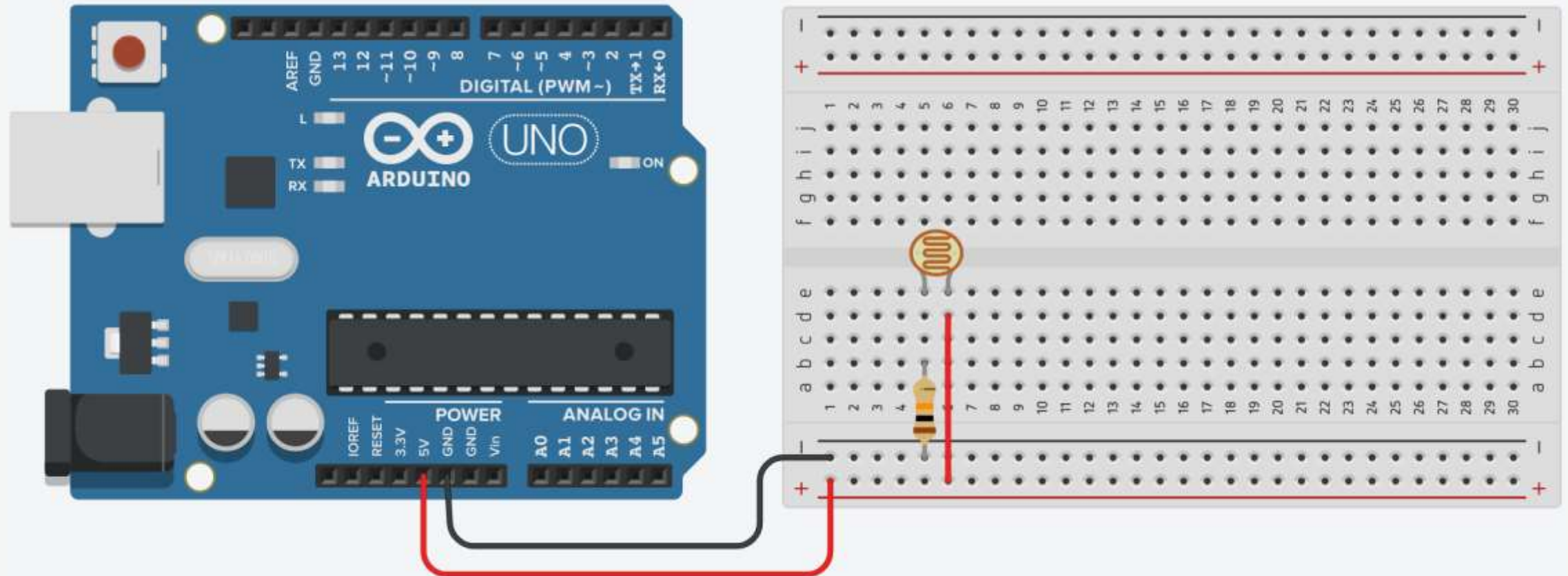
Photoresistor: Steps

3. Create a wire connecting one photoresistor leg to **power**.



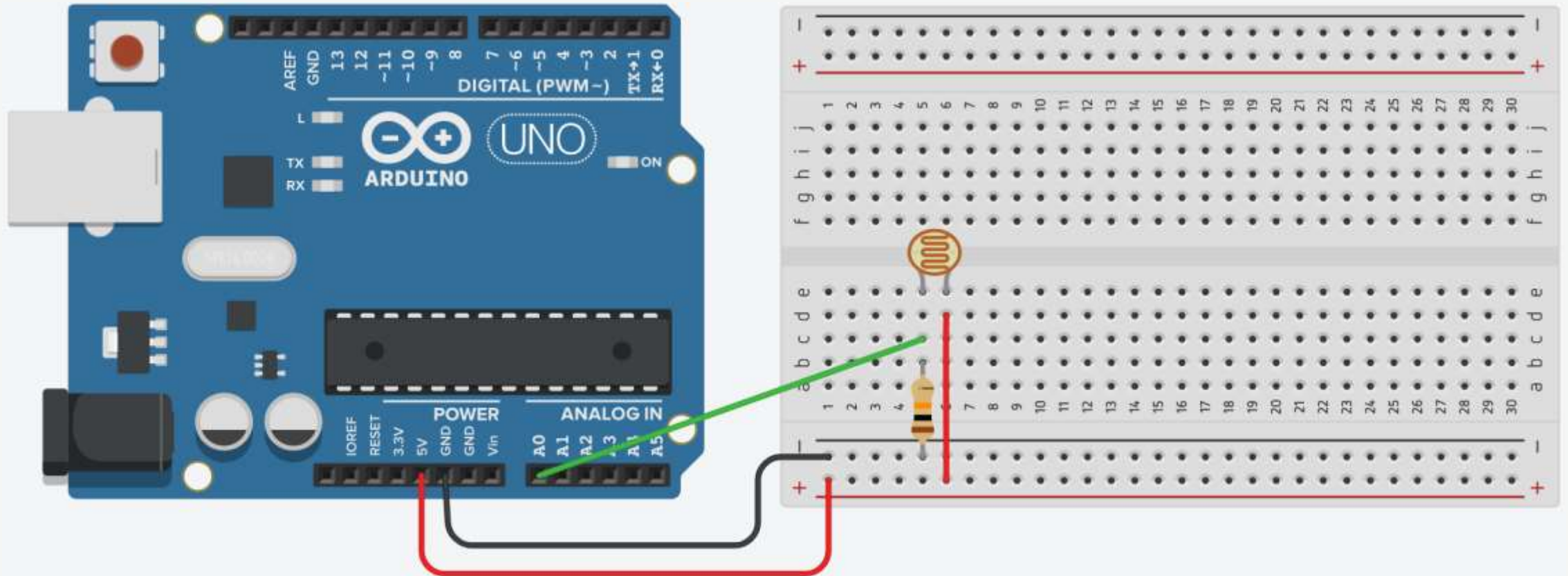
Photoresistor: Steps

4. Drag a $10\text{K}\Omega$ resistor to connect the other photoresistor leg to the ground.



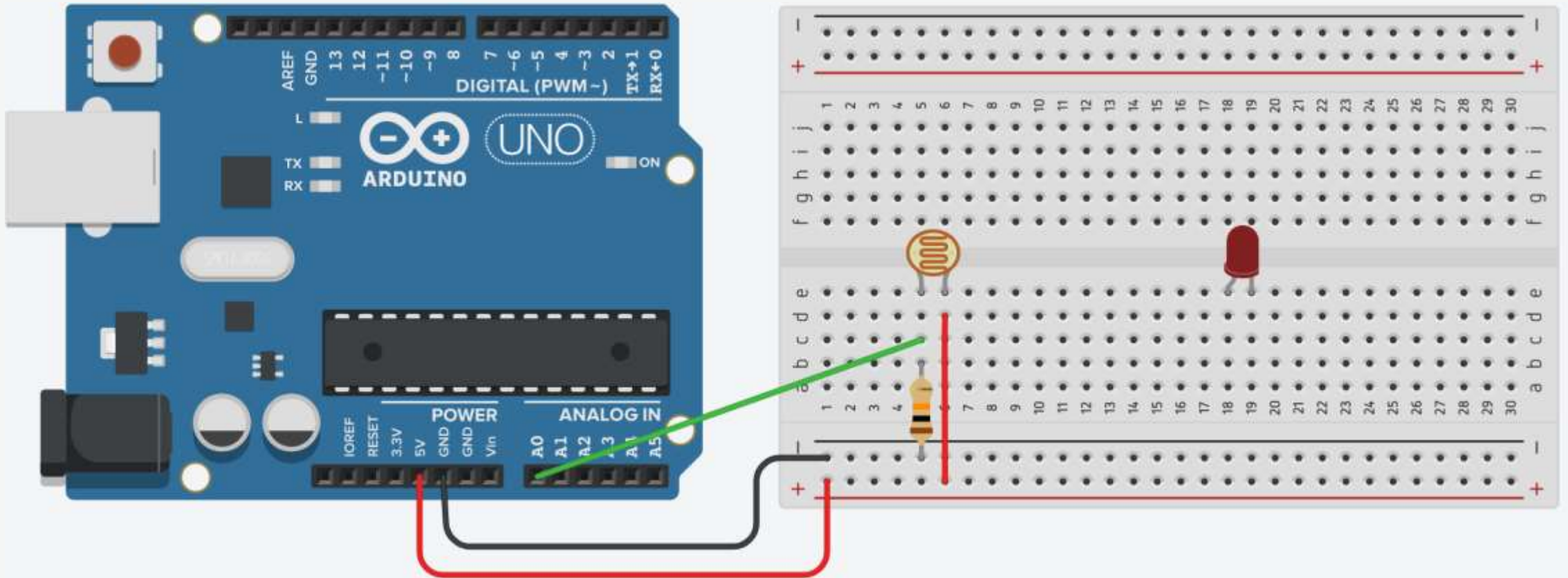
Photoresistor: Steps

5. Connect the photoresistor leg that is connected with the ground to the Arduino **A0 pin**.



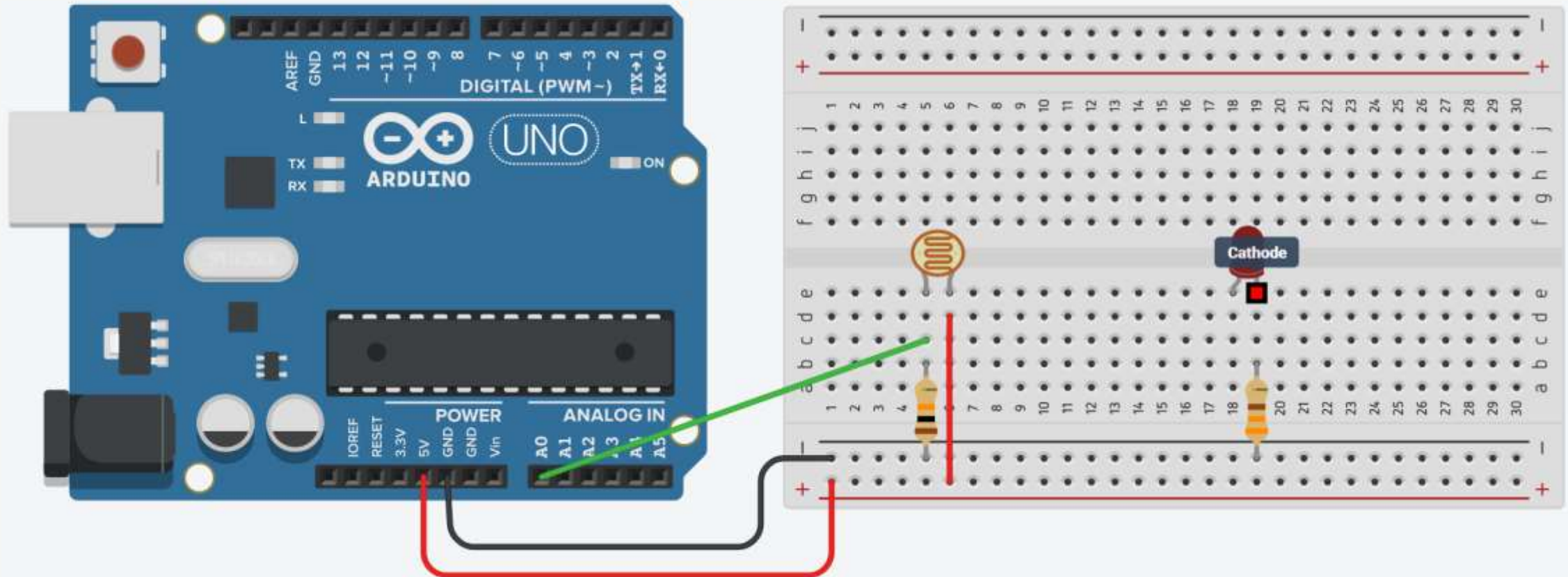
Photoresistor: Steps

6. Plug the **LED** into two different breadboard rows.



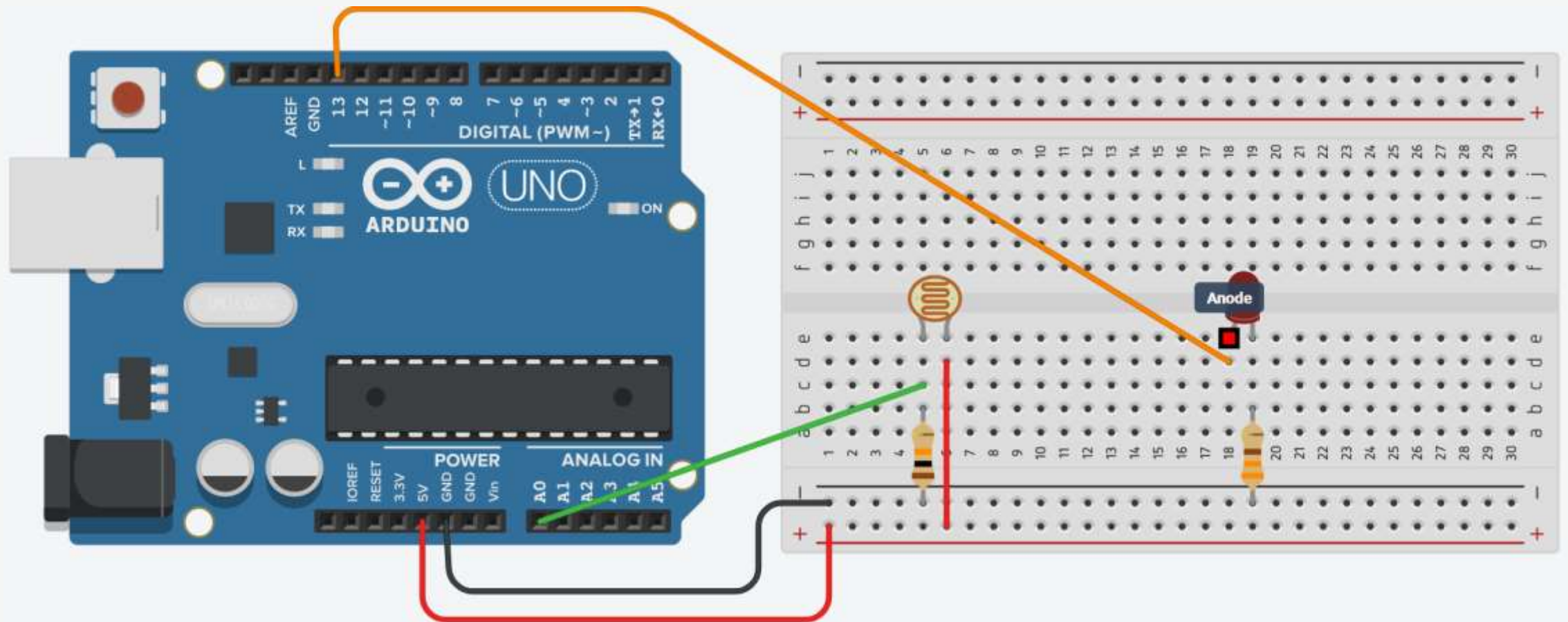
Photoresistor: Steps

- The **cathode (shorter leg)** connects to one leg of a **resistor of 330Ω** , and the **other resistor leg to the ground**.



Photoresistor: Steps

8. Wire up the LED anode (longer leg) to Arduino **pin 13**.



Photoresistor: Code

```
int photoresistor = 0;           // A variable holds the value of photoresistor
int threshold = 750;           //

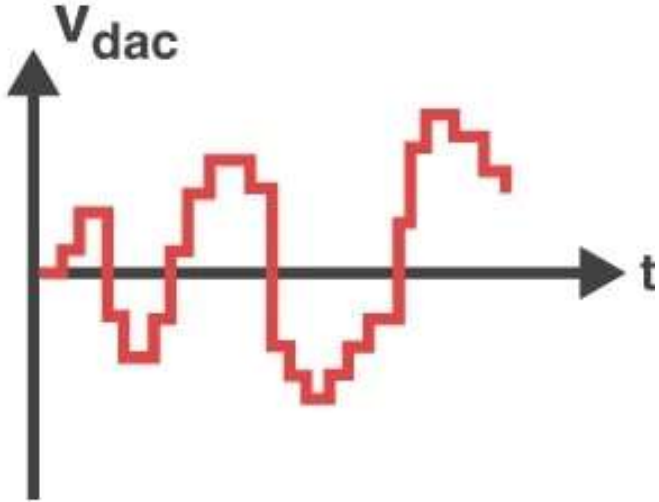
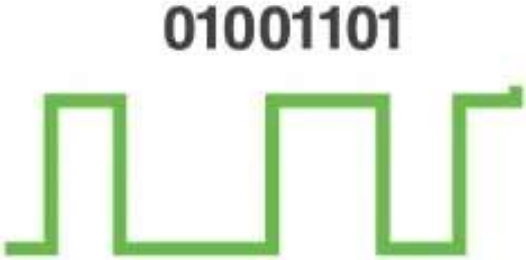
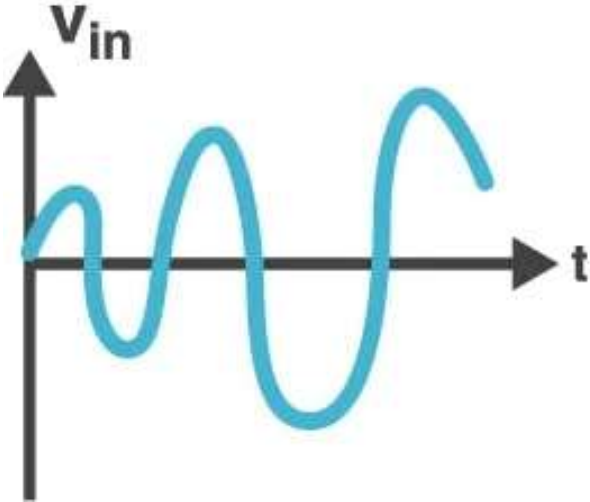
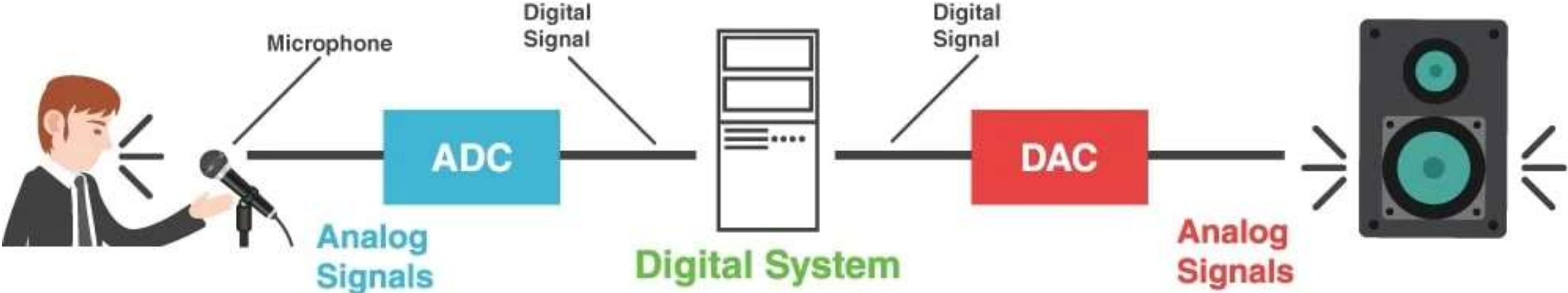
void setup()
{
  Serial.begin(9600);           // Start a serial connection with the computer
  pinMode(13, OUTPUT);         // Set pin 13 as an output pin
}

void loop()
{
  photoresistor = analogRead(A0); // Read the brightness of the LED
  Serial.println(photoresistor);  // Print the value of photoresistor

  // If the photoresistor value < threshold turn the light on, otherwise turn it off
  if (photoresistor < threshold)
    digitalWrite(13, HIGH);      // Turn on the LED
  else
    digitalWrite(13, LOW);      // Turn off the LED

  delay(100);                   // Short delay
}
```

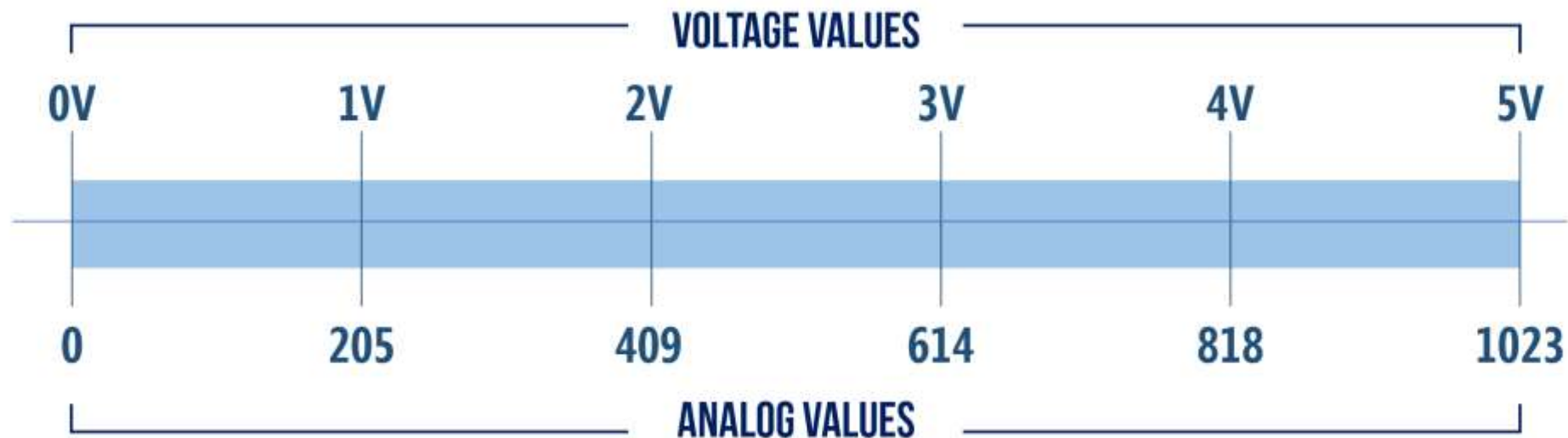

ADC vs. DAC



Read Analog Voltage

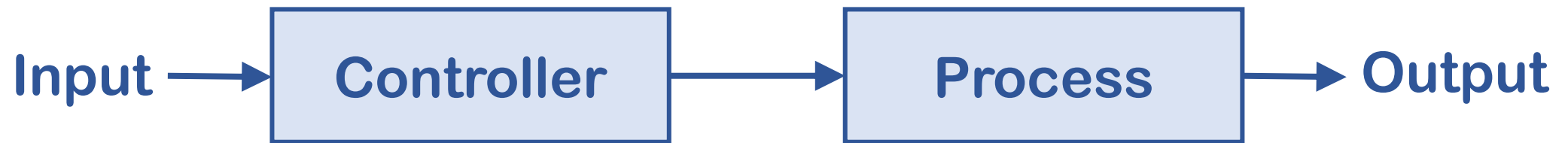
- The `analogRead()` returns a **number between 0 and 1023** that is **proportional to the amount of voltage** being applied to the pin.
- To scale the numbers between 0 and 5, divide 5 by 1023 and multiply that by `sensorValue` :

`voltage = sensorValue * (5.0 / 1023.0);`



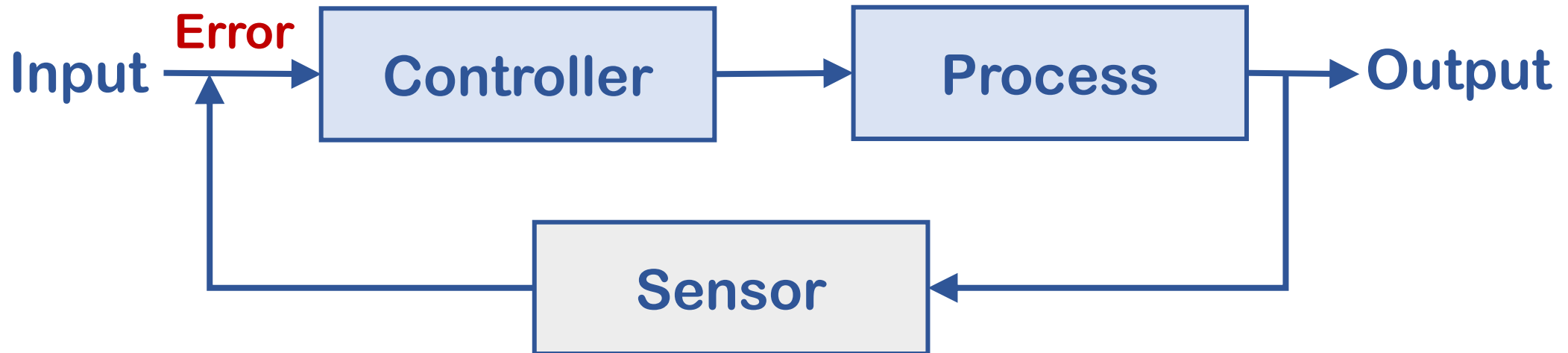
Closed-Loop vs. Open-Loop Control Systems

- An open-loop control system **does not monitor the output** to determine what adjustments to make to the input.
- For example, when using a clothes dryer, you might set the **timer** on the dryer to run the drying cycle for **one hour**.
- At the **end of the hour**, the **dryer will stop**.
- The **level of dryness** of the clothes will **vary depending upon their level of wetness** at the beginning of the cycle.



Closed-Loop vs. Open-Loop Control Systems

- In a **closed-loop** control system, the **output is measured** to determine whether it is the **desired output** and **adjust the input as appropriate**.
- For example, if the **clothes dryer** is equipped with **moisture sensors**, the input may be a **level of dryness** that adjusts the cycle by **extending the drying time** until the **sensors indicate the clothes are dried**.



Appendix 1: SparkFun Inventor's Kit (SIK)

- Go to <https://www.sparkfun.com/sikcode> and download the examples.



Appendix 2: Arduino Reference

- Go to <https://www.arduino.cc/reference/en/> to learn Arduino basics.

