Arduino 101: Making a LED Blink With a Button

Arman Mirkazemi on Aug 30th 2013 with 5 Comments

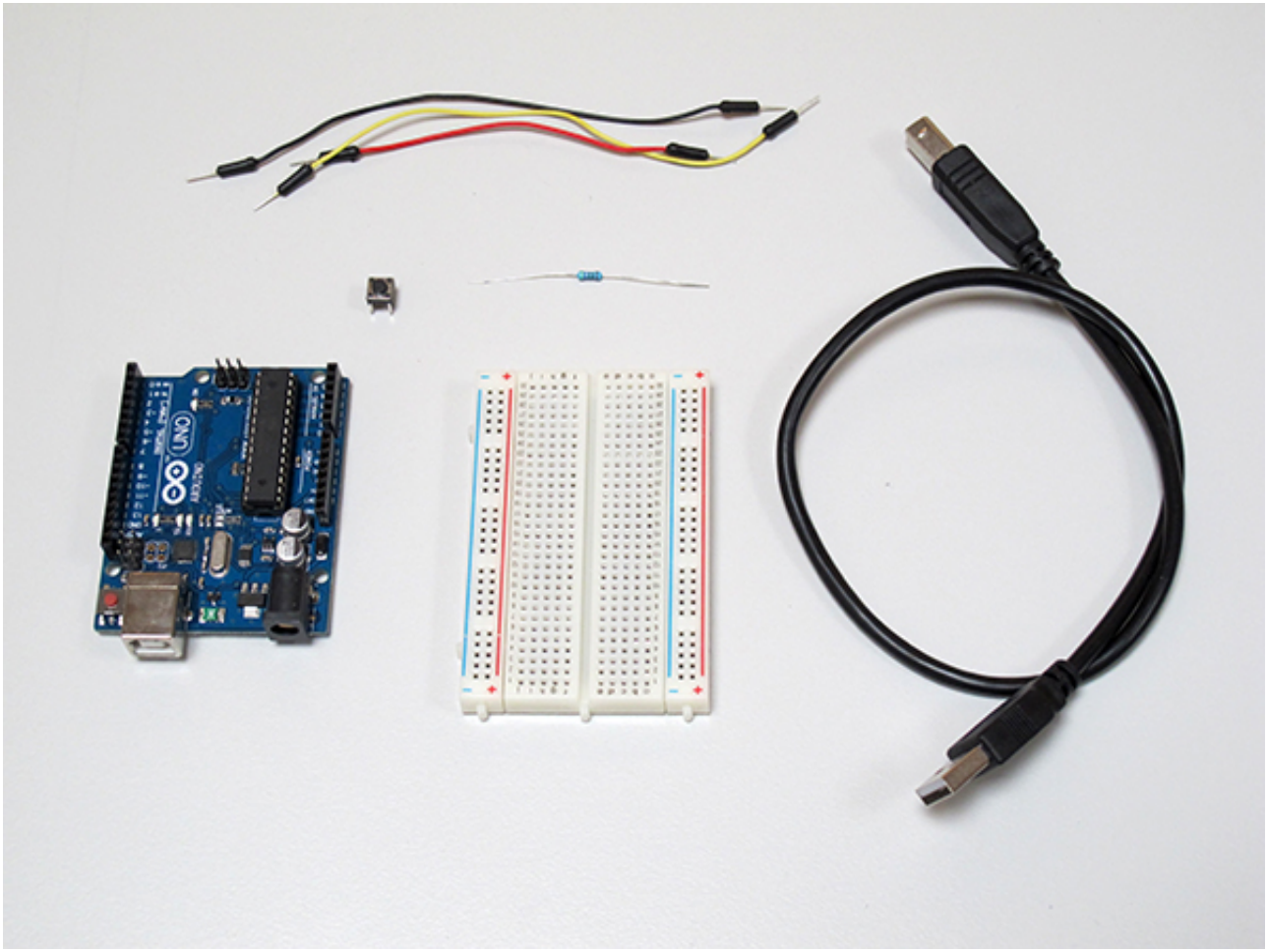
Tutorial Details

-
- **Topics:** Arduino, Electronics
- **Difficulty:** Beginner
- **Estimated Completion Time:** 30 minutes

In my last tutorial, I explained why it is easy to work on electronic projects with Arduino. In this tutorial I will be using an Arduino board to make a simple circuit, one that can turn on a LED light. Using some more basic code I will make the same LED light blink. Finally, I will add a push-button and use it to speed up the blinking.

Some of the diagrams in this article were developed using the Fritzing program.

Supplies



Arduino components

In this tutorial I will be using the following components:

- Arduino Uno R3 x 1
- Breadboard x 1
- Standard Type B USB cable x 1
- Push Button x 1
- 10K resistor x 1
- Short jumper wires x 3

You can buy these items either through a local electronics store, such as Fry's (if you live in the US), or via the Internet from the Arduino Website, Amazon, Little Bird Electronics or even eBay.

Alternatively, you may choose any of these other Arduino boards to follow this tutorial:

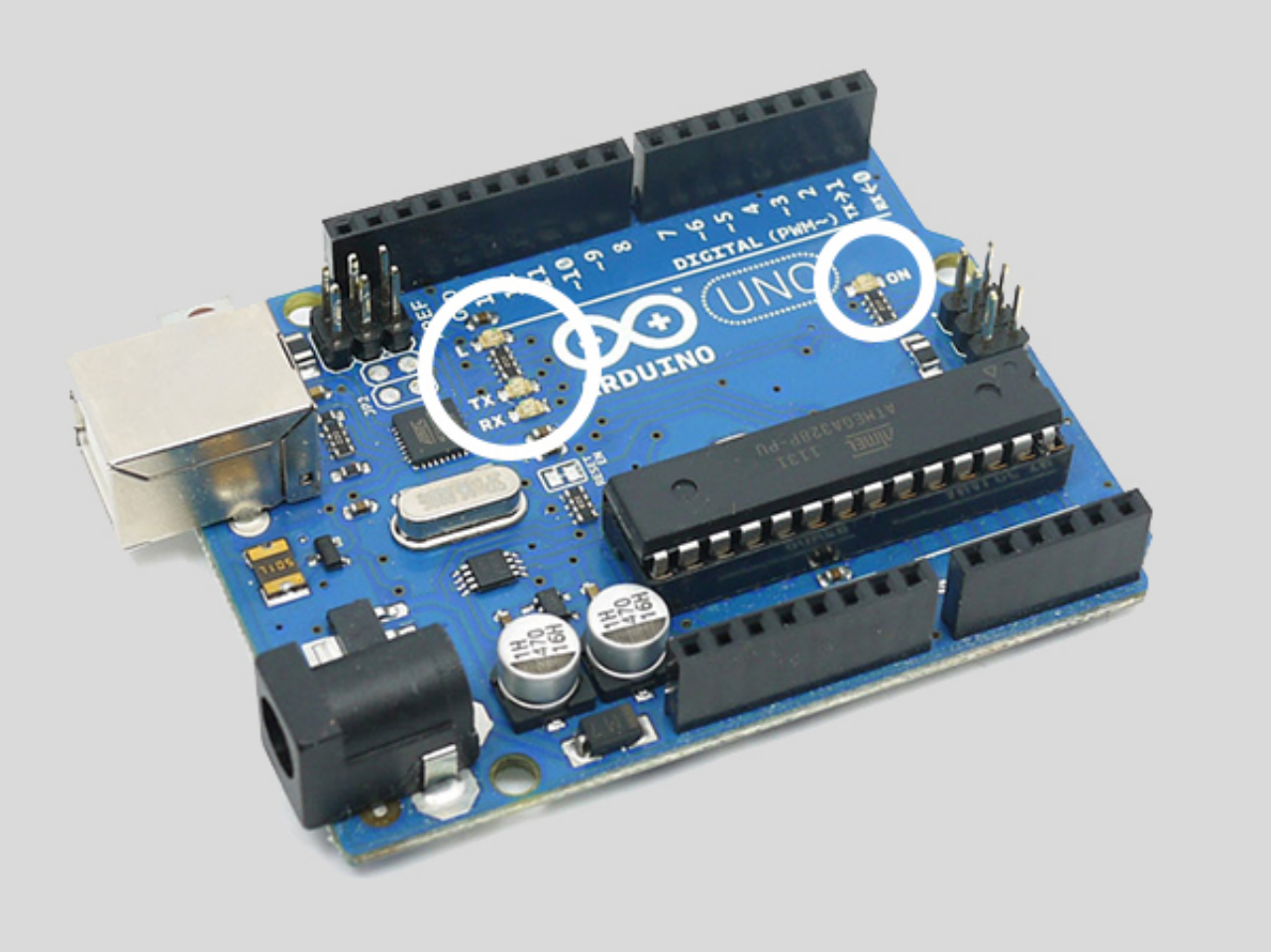
- Arduino Leonardo
- Arduino Mega Due
- Arduino Mega 2560
- Arduino Micro

Other Arduino boards or third party compatible boards may work as well. They, however, may require a

different USB connector. Ensure that your Arduino board can connect to your computer using a suitable USB cable.

Furthermore, ensure that you have downloaded and installed the latest version of the Arduino IDE on your computer. Installing the Arduino IDE may require a few more steps on Windows computers because you will need to install some drivers. For further instructions please follow the installation guides on the Arduino website.

You may be wondering how we are going to turn a light on when we have not included one in the list of components? That's easy! Arduino boards are usually equipped with an onboard LED, which we can use in our projects. To keep our circuit as simple as possible we are going to use this LED.



Our code is going to work with the light marked with letter L next to it.

---

1. Set Up a New Project

To begin, connect your Arduino board to your computer using the appropriate USB cable. You will notice that one or multiple lights may momentarily or permanently turn on. Arduino Uno has four onboard LED lights. If your board has more than one LED, blinking lights indicate that it is connected to a power source and is booting up. It takes around five seconds for Arduino to become ready for you to interact with it.

Launch the Arduino IDE on your computer. You will be presented with an empty working space where you will write the necessary code in order to program your Arduino board.

Next, you must tell the Arduino IDE which board your are going to connect to. From the menu, select **Tools > Board**, and then select your Arduino board from the list. If you have chosen to use an official Arduino board, then its name should be listed for you to select. Third party boards are usually equivalent to another official Arduino board. If you know which board that is, then go ahead and select that one from the list. Otherwise, refer to its manual to work out which model from the list should work with your specific board.

Finally, you must select the right port for communication with your Arduino board. Once again from the menu, go to **Tools > Serial Port**, and select the right Serial port. On Macs, the correct serial port is often listed as `/dev/tty.usbmodem1421` or similar. On Windows, the connection should be listed as a `COM` port.

---

2. Turning the Light On

Arduino is equipped with many different input and output connectors, which we will refer to as IO Pins. Right now, we want to use a digital IO pin to instruct the LED light to come on. Because we are also using the onboard LED light, the appropriate IO pin has already been decided for us by the Arduino makers. It is pin 13, which, by design, has been attached to the onboard LED light.

Copy the following code and paste it into your Arduino IDE:

```
1 int led_pin = 13; void setup() { pinMode(led_pin, OUTPUT); }
2 void loop() { digitalWrite(led_pin, HIGH); }
```

In the code, I have made use of two Arduino functions: `pinMode(pin_number, mode)` and `digitalRead(pin_number, value)`.

I will call the `pinMode()` function inside `setup()` to instruct Arduino to treat its pin-13 as an output. Then by calling the `digitalWrite()` function inside `loop()` I can activate `HIGH` signal on pin-13 which turns on the LED light.

Click on the **Upload** button to push the above code into your Arduino board. Provided the correct Arduino board and Serial port is selected, you should see a progress bar followed by a **Done Uploading** message.

Whilst the code is being uploaded you may see different lights flashing, which indicates successful communication between your PC and the Arduino board. At this point, the onboard LED light should be permanently turned on.

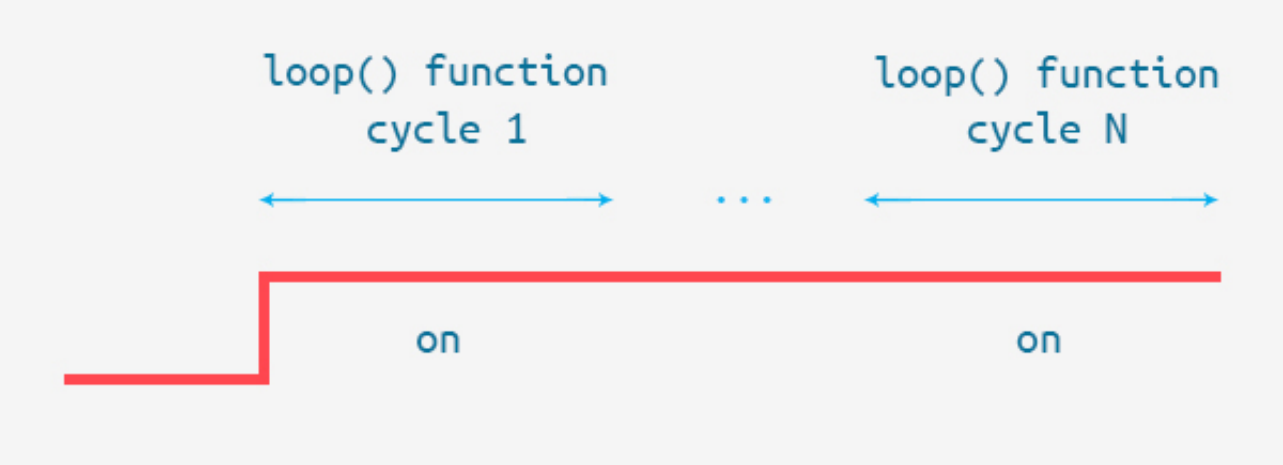
---

3. Make It Blink

You have completed the Arduino equivalent of “Hello World”. Now you are going to make that light blink by introducing the `delay()` function into the above code. The `delay()` function accepts an integer value, equal to a length of time in milliseconds. 1000 milliseconds is equal to one second.

```
int delay_value = 1000; int led_pin = 13;
void setup() {
  pinMode(led_pin, OUTPUT);
}
void loop() {
  digitalWrite(led_pin, HIGH);
  delay(delay_value); digitalWrite(led_pin, LOW);
  delay(delay_value);
}
```

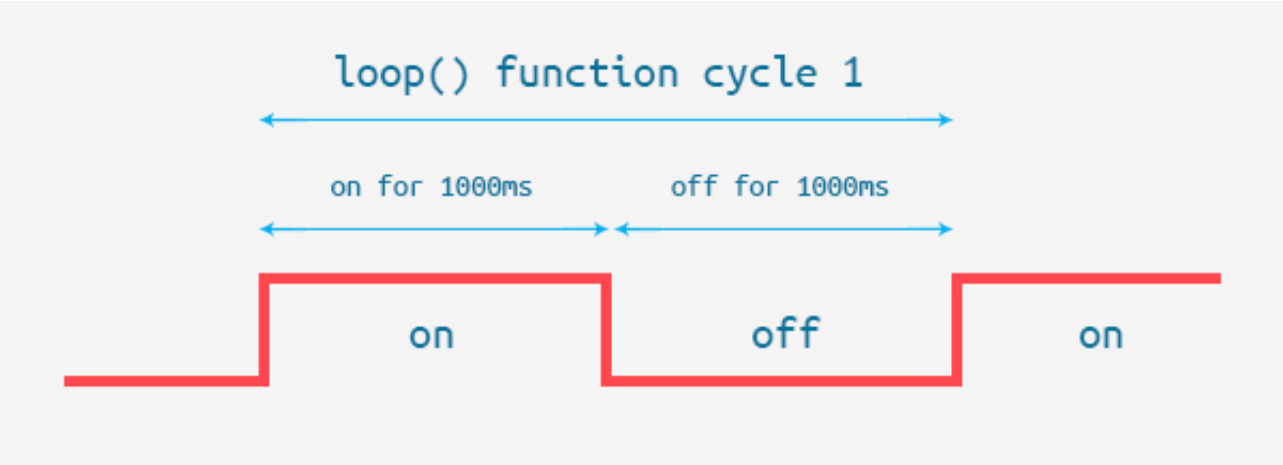
The `delay()` function keeps the LED light on and also turns it off for one second during every iteration of the `loop()` function. As the `loop()` function is perpetually repeated, this code will turn the LED light on and off repeatedly.



Digital Wave diagram

Using a digital wave diagram, I can loosely explain how the original behavior has changed. Before introducing the delay function in step three, I was producing a digital wave that looked like the diagram, above. Every cycle of every loop() function iteration was spent on keeping the LED light turned on.

Using the delay() function, our code divides each cycle into two parts, making every iteration last for two seconds. During the first second the LED is turned on, and, during the next second it is turned off.



This figure shows how the the delay() function is used to maintain the current state.

---

4. Change The Delay Time Using a Button

Up to this point, the behavior of this Arduino project has been nicely driven by the code we have written. However, once it is uploaded and running, we don't have any way of interacting with this electronic circuit. This is very static and I am going to change it by adding a pushbutton which will let me change the blinking speed. In this step, I will need to use a *breadboard*. This is a good time to take a look at what a breadboard is and how it can be used.

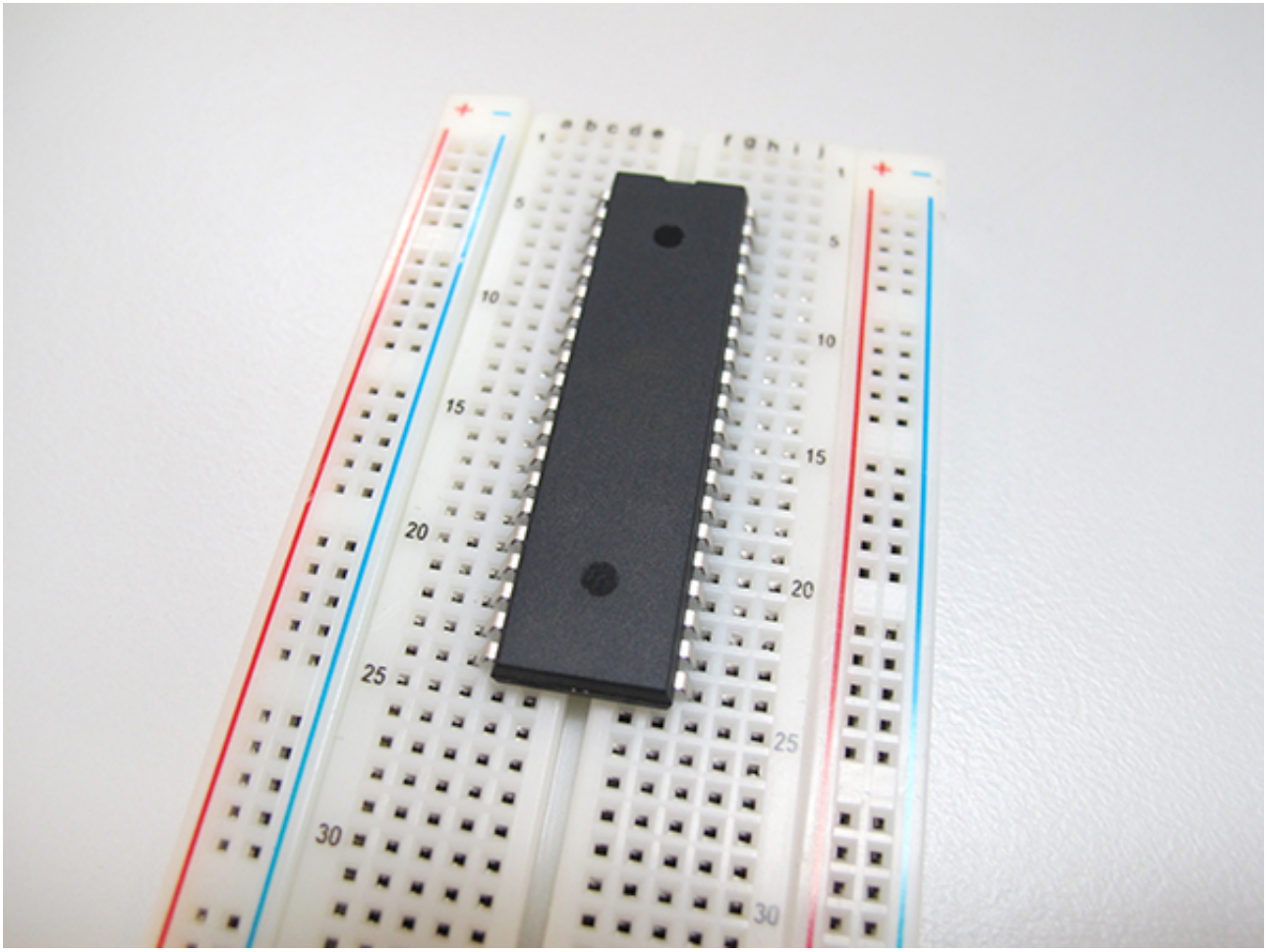
Breadboard

A breadboard is a solderless prototyping board used for making temporary electronic circuits, mainly for experimenting with different circuit board designs.

Modern breadboards are made up of a solid piece of perforated plastic, with many copper clips under its surface for making electrical connections. The numerous holes on the surface of these boards allow for inserting different electronic components without the need to solder any in place.

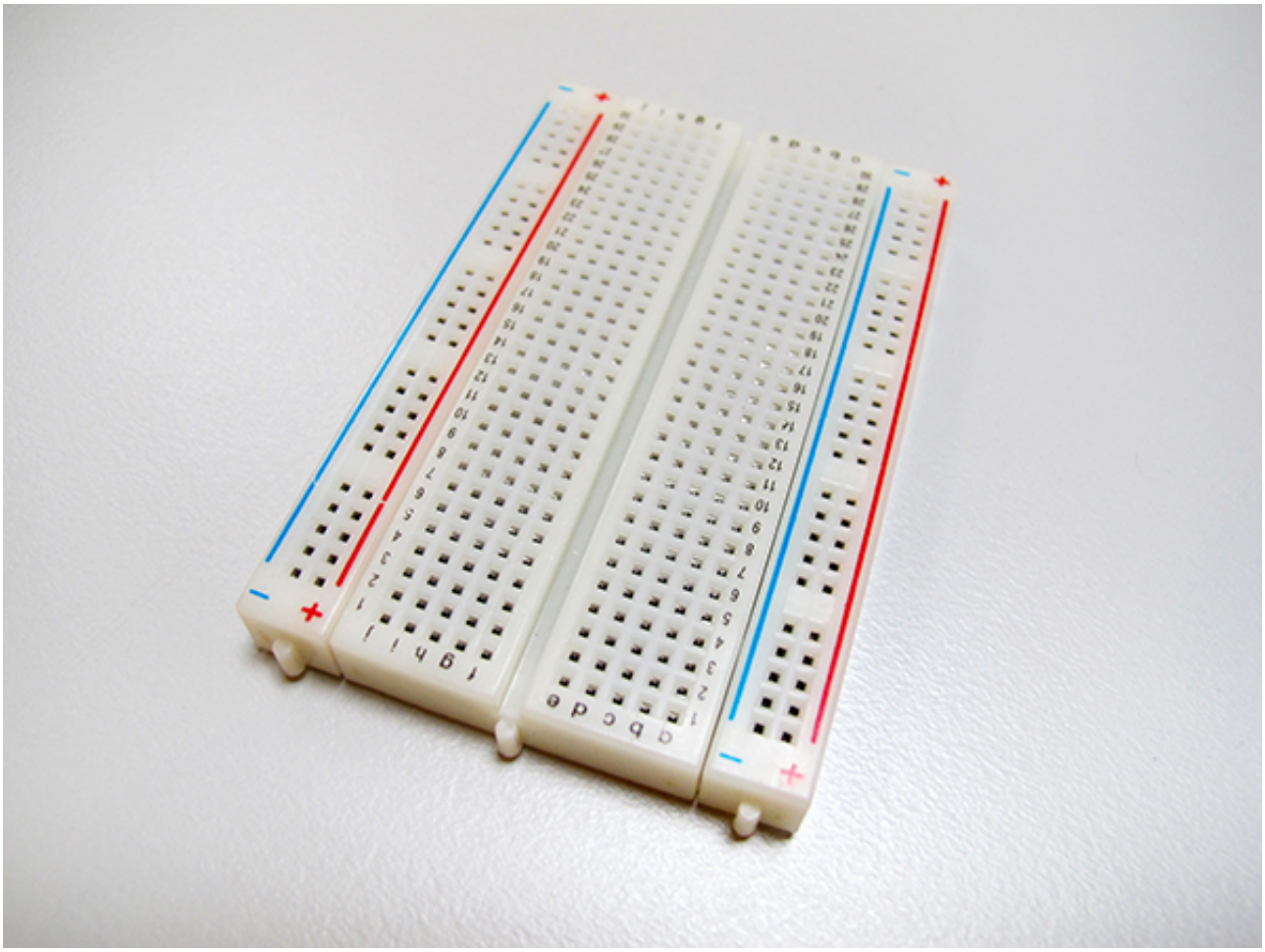
Common breadboards have two columns, each with their own rows of five holes. Any hole is connected to all other holes in the same row but not to any of the holes in the adjacent column. In other words, if two wires are not placed in the same row of the same column, then the two are not connected to each other. This setup allows us to share a single connection from a component through four other connection points in the same row.

The reason for having two or more columns on the same breadboard is to allow multiple Integrated Circuits (ICs) to be connected. ICs are usually connected into both columns of a breadboard because they have more than two pins on both sides as illustrated in the following diagram.



40-pin IC attached to a breadboard.

On either sides of most boards, there are long strips that are used for sharing power. These strips are often referred to as the bus strips, or power rails, as they run along the entire length of the board. The power rails on some boards will let you connect both positive and ground connections. Unlike normal rows, all of the holes along the length of the board are connected to each other.

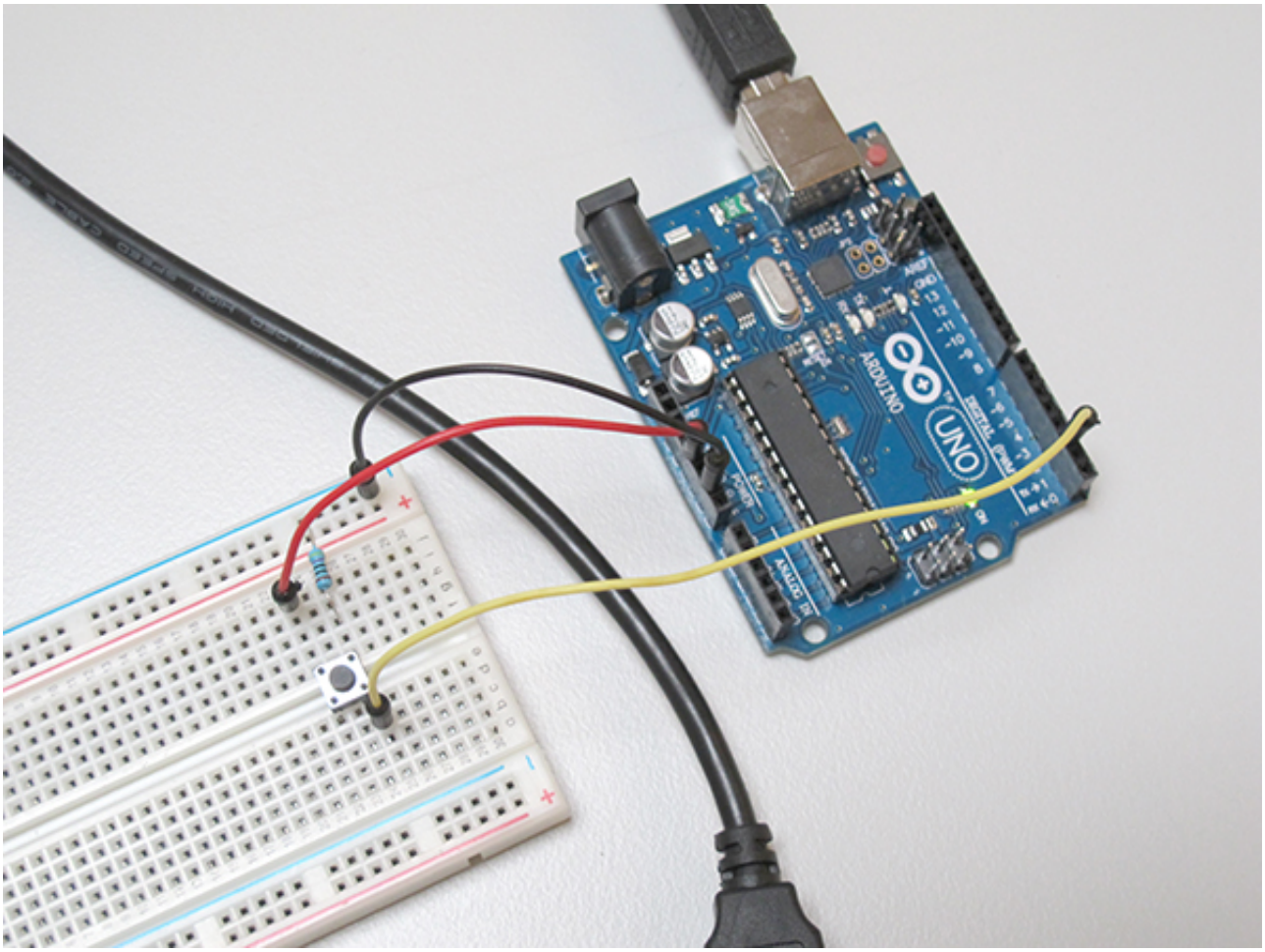


Power rails are usually marked with red and blue lines running along the entire length of the board.

When buying a breadboard, it's a good idea to choose the type with grooves on either sides of the board. These grooves can be used to attach multiple breadboards together to make a larger working space. A good quality breadboard is marked with numbers and letters making it easy to identify each row and column.

Step 1

Begin by placing the push button and connecting it with the power jumpers from the Arduino board. Arduino Uno can output two levels of power, 3-Volts and 5-Volts. For this circuit, we will need to use the 5V rail. The reason why you'd use one over the other depends on the components you're going to connect. Some components may need lower power voltage to operate, hence the 3V output.

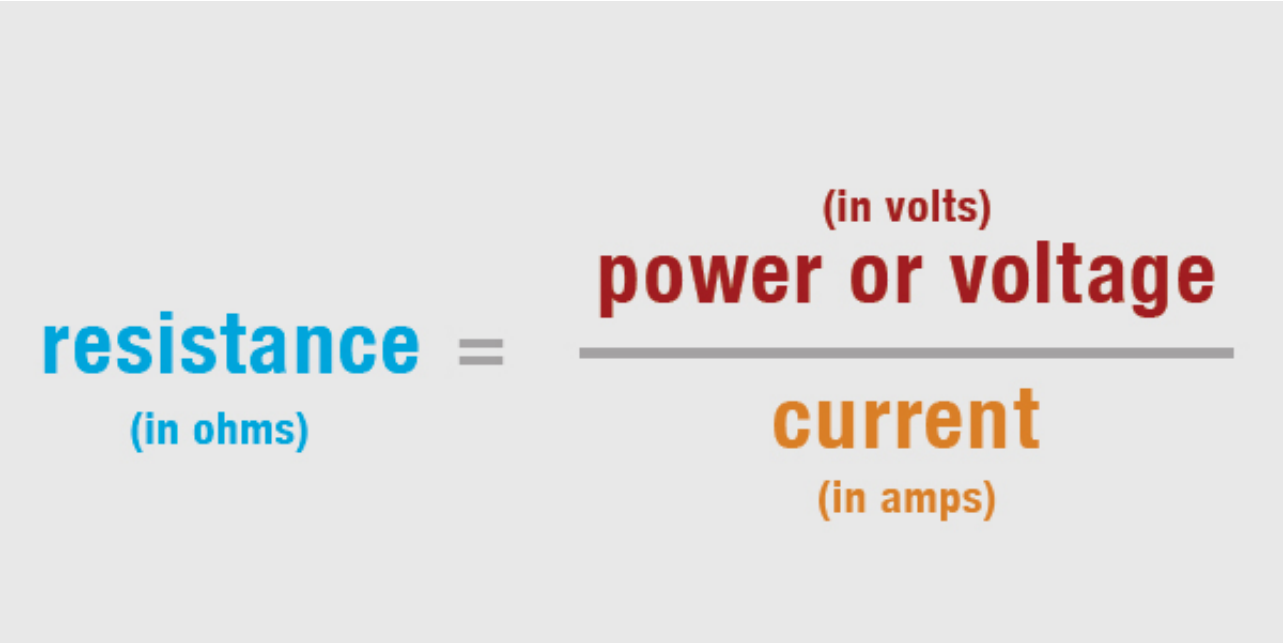


Circuit with Arduino and breadboard

In the diagram, above, we have a complete circuit. We have connected the top pins of the push button to both the 5V-pin, on the Arduino, and to the 10K resistor. This then connects to the ground (GND) pin on our Arduino. Our third wire (in yellow) connects to the digital pin–2 and will carry the ON signal to the Arduino board.

Resistor

The purpose of a resistor is to slow down the electrical current, as current passes through it, thereby limiting the amount of current flowing through the circuit. This is achieved by making resistors from materials with a low conductive property. Resistance is measured in *ohms* andcan be determined from the follow equation:



resistance (in ohms) = power or voltage (in volts) / current (in amps)

Suppose I want to connect a LED light to a 9V power source, but the LED can tolerate only 30 milliamps of current. Based on the above equation we will need to use a 300 Ohms resistor in order to limit the current flowing through the LED light.

9 volts / 0.03 milli-amps = 300 Ohms

There are three main categories of resistors:

1. Fixed resistors, such as the one we are using here,
2. Variable resistors, commonly known as *Potentiometers*, and
3. Variable resistors that are dependent on physical qualities, such as temperature (*thermistors*) or light

(*photovoltaic cells*)

Although resistors serve to limit current flow, there are different types of resistors in each of those three categories for different applications.

Most fixed resistors are marked with colored bands to help us work out their resistance. From the left, the first two bands give the first and second digits of the resistance value. The third band gives the multiplication factor. Finally, the fourth band gives the tolerance of the resistor.

Resistor color codes

From the colors on the above resistor, we can work out the following:

- Brown (first digit) = 1
- Black (second digit) = 0
- Orange (multiplier) = 10^3
- Gold (tolerance) = +/- 5%
- 10 * 10^3 = 10,000 ohms or 10 killo ohms or 10K resistance

Here is the complete table of colors. For more information, please refer to Wikipedia on resistor color codes.

Color	Digit	Multiplier	Tolerance
Black	0	x10^0	±1%
Brown	1	x10^1	±2%

Red	2	x10^2	-
Orange	3	x10^3	(±5%)
Yellow	4	x10^4	±5%
Green	5	x10^5	±0.25%
Blue	6	x10^6	±10%
Violet	7	x10^7	±1%
Gray	8	x10^8	±0.05% (±10%)
White	9	x10^9	-
Gold	-	x10^–1	±5%
Silver	-	x10^–2	±10%
None	-	-	±20%

The Purpose of the Resistor

I connected the yellow signal wire from digital pin 2 to one leg of the pushbutton. That same leg of the button, on its other side, connects through the 10K resistor to the ground to form a complete circuit. When the button is not pushed, the traveling current gets read by Arduino as a LOW.

Once the button is pushed down, a connection between pin 2 and positive 5V will get established through the push button legs. Since electricity will always travel through the path of least resistance, it will avoid going through the resistor and will flow through to pin 2 which results in a HIGH reading by the Arduino board.

Step 2

Now, let's finish Step 4 and make the LED light blink faster when we press down the pushbutton.

```
int delay_value = 1000;
int led_pin = 13;
int button_pin = 2;
void setup() {
  pinMode(led_pin, OUTPUT); pinMode(button_pin, INPUT);
}
void loop() {
  digitalWrite(led_pin, HIGH);
  delay(delay_value);
  digitalWrite(led_pin, LOW);
  delay(delay_value);
  int button_state = digitalRead(button_pin);
  if (button_state == HIGH) {
    delay_value = 100;
  } else {
    delay_value = 1000;
  }
```

```
18 }
```

This time, the code is instructing Arduino to treat its pin–2 as an input source by calling the `pinMode(button_pin, INPUT)` inside the `setup()` function. This allows us to read the state of the pushbutton later inside the `loop()` function by calling `digitalRead(button_pin)`. Getting the state of the pushbutton lets us determine whether the delay function should be called with a smaller value.

Now go ahead and upload the above code to your Arduino, and then press the pushbutton to see the LED light blink faster.

Troubleshooting

If you've gotten this far and the above code isn't working for you, there could be a few reasons for this:

- This may sound obvious, but ensure your Arduino is connected to a power source and the ON LED is turned on.
- Make sure all the pins and wires, the resistor and the push button are firmly connected to your Arduino board and to the breadboard. If you're unsure about continuity of your connections, use a multimeter to measure the continuity.
- Ensure all connections to the Arduino board are connected to the right digital inputs.
- If your problem persists, then consult the Arduino troubleshooting guide.

Conclusion

The finished project

In this tutorial, you learned some basic techniques in using an Arduino board, a breadboard, resistors and pushbuttons along with the Arduino IDE. You also learned how the `delay()` function can be used to maintain a state for any given length of time.

If you have any questions about this tutorial, please leave them in the comments section, below.

By Arman Mirkazemi

This author has yet to write their bio.

Note: Want to add some source code? Type <pre><code> before it and </code></pre> after it. Find out more