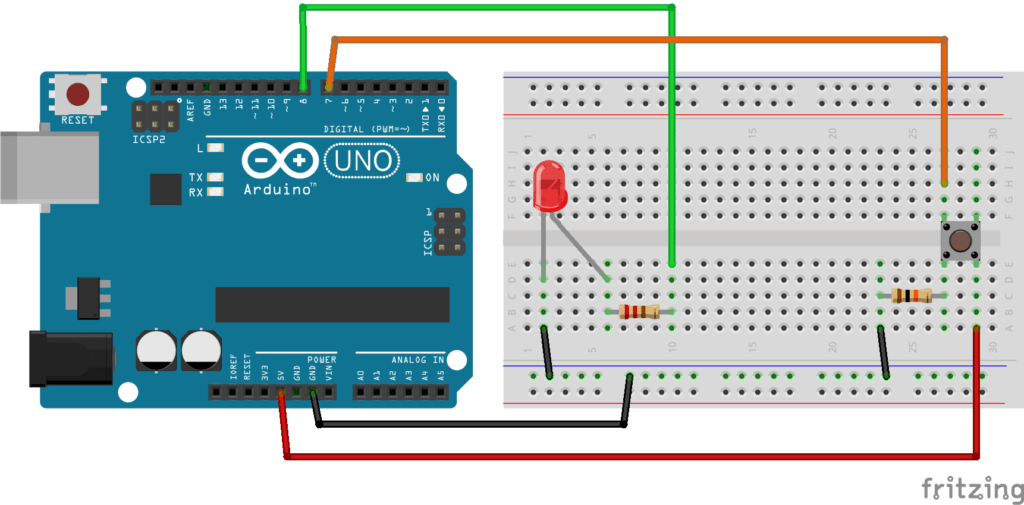
**Lab Exercise 2- Arduino circuit with and Code for Push Button and LED**

To build the circuit you will need those components:

* Arduino board (any board, if you don’t have Uno you can easily adapt by finding corresponding pins).
* Breadboard.
* LED – any color.
* Push button.
* 220 Ohm resistor for the LED. If you don’t have this specific value, any resistor from 330 to 1k Ohm will do.
* 10k Ohm resistor for the push button.
* A bunch of male to male wires (including if possible black, red, and other colors).

Here’s the circuit you have to make.



Step by step instructions to build the circuit:

* First, make sure to power off your Arduino – remove any USB cable.
* Plug a black wire between the blue line of the breadboard and a ground (GND) pin on the Arduino board.
* Plug the LED. You can notice that the LED has a leg shorter than the other. Plug this shorter leg to the ground (blue line here) of the circuit.
* Connect the longer leg of the LED to a digital pin (here pin no 8, you can change it). Add a 220 Ohm resistor in between to limit the current going through the LED.
* Add the push button to the breadboard, like in the picture.
* Connect one leg of the button to the ground, and put a 10k Ohm resistor in between. This resistor will act as a “pull down” resistor, which means that the default button’s state will be LOW.
* Add a red wire between another leg of the button and VCC (5V).
* Finally, connect a leg of the button (same side as the pull down resistor) to a digital pin (here 7).

All right your circuit is now finished. You can start writing code.

Turn on the LED when button is pressed, turn it off otherwise

What we want to achieve is simple: when the button is not pressed, the LED is off. And when we press the button the LED should be on.

**The code**

**#define LED\_PIN 8**

**#define BUTTON\_PIN 7**

**void** setup() {

pinMode(LED\_PIN, OUTPUT);

pinMode(BUTTON\_PIN, INPUT);

}

**void** loop() {

**if** (digitalRead(BUTTON\_PIN) == HIGH) {

digitalWrite(LED\_PIN, HIGH);

}

**else** {

digitalWrite(LED\_PIN, LOW);

}

}

Let’s break this code down line by line.

**Setup**

**#define LED\_PIN 8**

**#define BUTTON\_PIN 7**

First, as a best practice, we use some defines to keep the pin number for the LED and push button. That way, if you have used different pins than I, you just need to modify those 2 lines. Also, in the future if you want to change the LED from pin 8 to pin 11 for example, you can modify this line without touching anything else in the code.

**void** setup() {

pinMode(LED\_PIN, OUTPUT);

pinMode(BUTTON\_PIN, INPUT);

}

The setup function is executed once at the beginning of the program. This is the perfect time to initialize our pins with the pinMode() function:

* OUTPUT for the LED, as we’re going to write data to it.
* INPUT for the push button, as we’re going to read data from it.

Now, the digital pins are correctly set up.

**Loop – Turn on the LED when button is pressed**

**void** loop() {

**if** (digitalRead(BUTTON\_PIN) == HIGH) {

digitalWrite(LED\_PIN, HIGH);

}

**else** {

digitalWrite(LED\_PIN, LOW);

}

}

In the loop function, we start by reading the button’s state with the digitalRead() function. As we have a pull down resistor on the button, we know that the non-pressed state will give us the value LOW.

So, once we get the button’s state, we check if it’s HIGH or LOW:

* HIGH (pressed): we power on the LED with digitalWrite() and the HIGH state.
* LOW (not pressed): we power off the LED with digitalWrite() and the LOW state.

**Simulation:**

