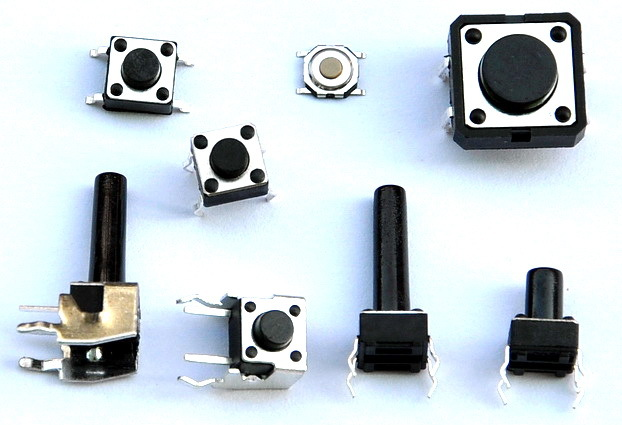
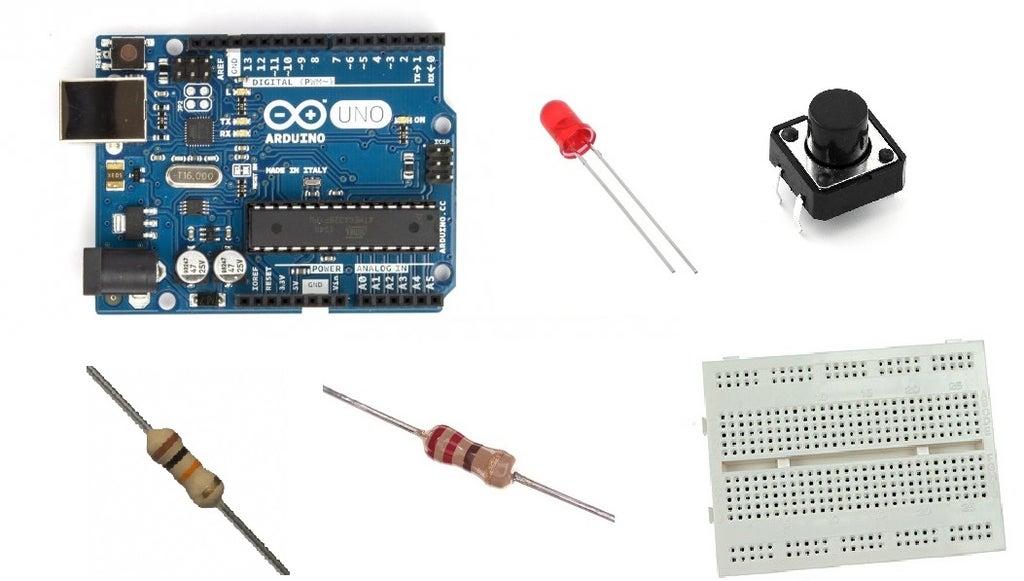
**Understanding Pushbuttons and Pull Up and Pull Down Resistors:**

# **Push Button in Arduino:**

Push buttons or switches connect two points in a circuit when you press them. This example turns on one led when the button pressed once, and off when pressed twice.



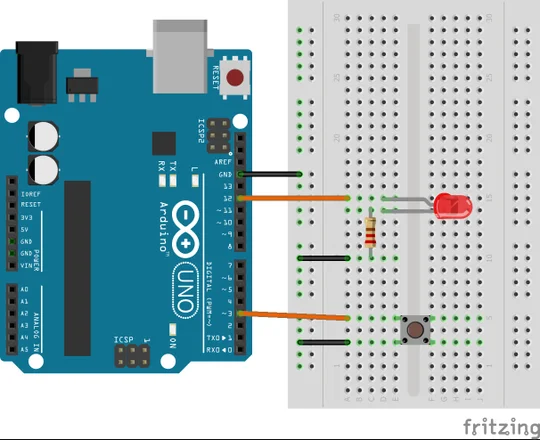
## **What You Will Need to make a circuit:**

****

For this tutorial you will need:

* Arduino uno
* Breadboard
* LED
* 220 Ohm & 10 KOhm resistor
* Push button

## **The Circuit:**

****

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH.

## **The Code:**

const int buttonPin = 4;

const int ledPin = 3;

//Variables

int buttonState = 0;

int flag=0;

void setup() {

//Input or output?

pinMode(ledPin, OUTPUT);

pinMode(buttonPin, INPUT\_PULLUP);

}

void loop(){

//Read button state (pressed or not pressed?)

buttonState = digitalRead(buttonPin);

//If button pressed…

if (buttonState == LOW)

{

//...ones, turn led on!

if ( flag == 0)

{

digitalWrite(ledPin, HIGH);

flag=1; //change flag variable

}

//...twice, turn led off!

else if ( flag == 1){

digitalWrite(ledPin, LOW);

flag=0; //change flag variable again

}

}

delay(200); //Small delay

}

**They are two types of resistors:**

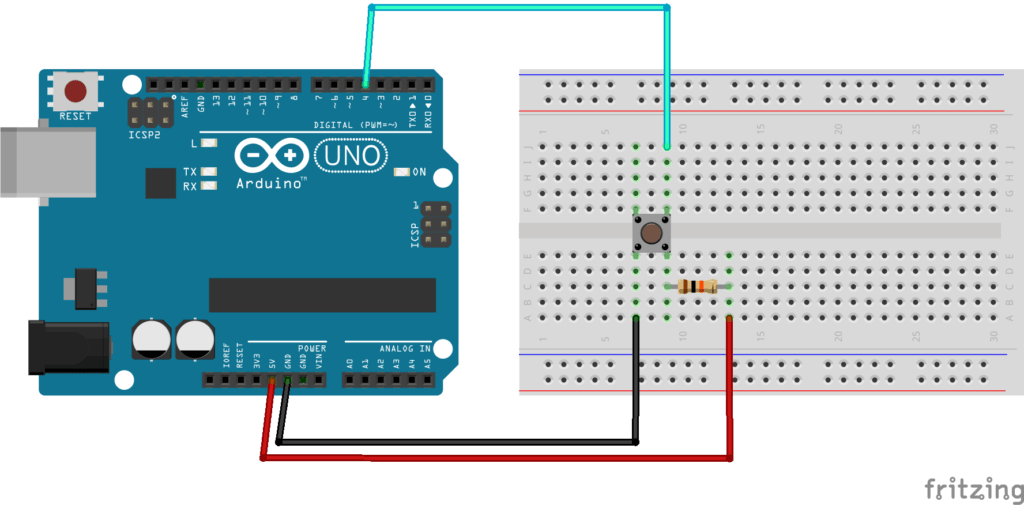
* Pullup
* Pulldown

With this little test I hope you'll understand why the pull-up (and pull-down) resistors are needed in digital circuits like in Arduino.

* With a pull-up resistor and with the button unpressed you make a logic state ON and with the button pressed you make a logic OFF.
* With a pull - down resistor and a pressed button you make an ON logic state and OFF logic state when its unpressed.

### **Arduino push button with external pull up resistor**

For this circuit you will need a 10k Ohm resistor, and every other component we previously used.



* Before you modify the circuit make sure to power off the Arduino, to avoid any risk for your hardware.
* For the ground wire (black) and data wire (blue), that’s the same thing as before. Make sure they are on different sides of the button – for example GND on left side, and digital pin on right side.
* You are going to add the 10kOhm resistor to the same side as the digital pin connection – in this example on the right side. The other leg of the resistor goes to a different line on the breadboard, and from this leg, you add a wire (red) to the 5V pin of the Arduino. This is the pull up resistor, which will make sure the default voltage you read is 5V, hence HIGH in the code.

Now, here’s the code.

**#define BUTTON\_PIN 4**

**void** setup()

{

Serial.begin(9600);

pinMode(BUTTON\_PIN, INPUT);

}

**void** loop()

{

byte buttonState = digitalRead(BUTTON\_PIN);

**if** (buttonState == LOW) {

Serial.println("Button is pressed");

}

**else** {

Serial.println("Button is not pressed");

}

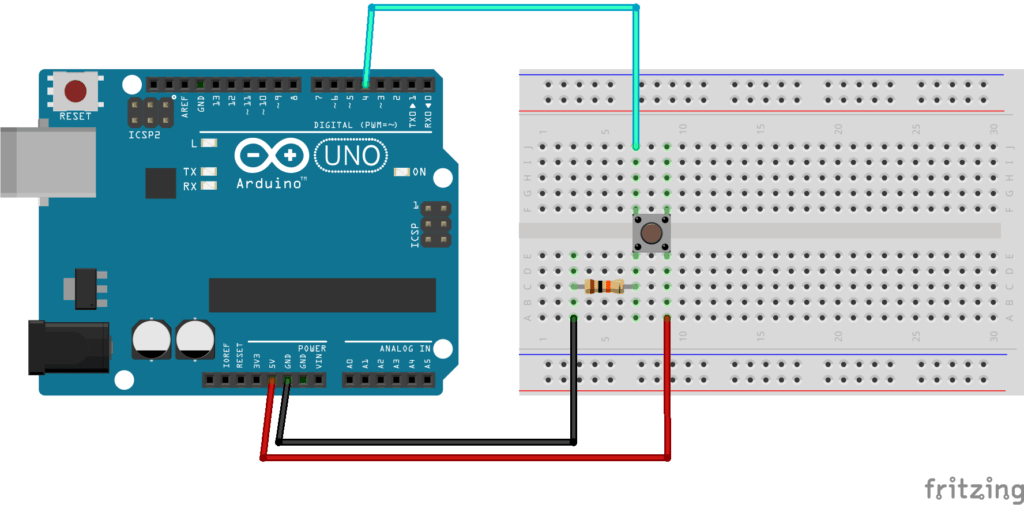
delay(100);

}

As you can see, the code is the same, we just modified the mode in the pinMode() function. Instead of using INPUT\_PULLUP, which will set the pin as INPUT + activate the internal pull up resistor, we just use INPUT.

### **Arduino push button with external pull down resistor**

For this circuit we will also use a 10k Ohm resistor.



The principle is the same, but note that things are a bit different here:

* Once again, make sure to power off the Arduino before doing anything.
* You are going to add the 10k Ohm resistor on the side connected to the ground. So, starting from one leg of the button, you have the resistor, and then a wire (black) connected to GND.
* The wire connected to the digital pin should now be on the same side as the resistor – in this example, both GND and digital pin connections are on the left side (and not on opposite sides).
* You add a wire (red) between one leg of the button – from the opposite side of GND, so here on the right side – and 5V.

And here is the code.

**#define BUTTON\_PIN 4**

**void** setup()

{

Serial.begin(9600);

pinMode(BUTTON\_PIN, INPUT);

}

**void** loop()

{

byte buttonState = digitalRead(BUTTON\_PIN);

**if** (buttonState == HIGH) {

Serial.println("Button is pressed");

}

**else** {

Serial.println("Button is not pressed");

}

delay(100);

}

As you can see, we also use the INPUT mode in the pinMode() function.

But now, when we read the state from the push button, LOW means that the button is not pressed. So we have to change the way we analyze the data in our code. That’s what you can see on line 13: if the state is HIGH, now it means the button has been pressed.