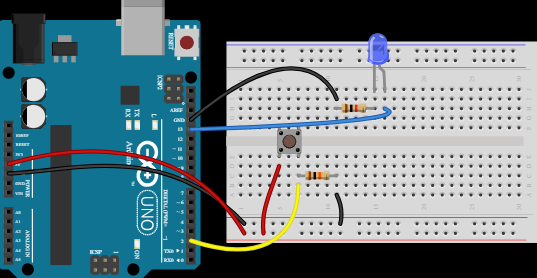
**Digital Sensors – Push Button**

Let’s start with a simple assembly, an Arduino that will ***turn on an LED when you press a button***. To carry out this installation, we will need an Arduino, a prototyping board (or breadboard), a push button, an LED, Prototyping wires and some 1 Kiloohm resistors. Here is the assembly we’re going to build

**Assembly**



To make this, you will need:

* An Arduino board
* A USB cable
* Two 1KΩ resistors
* Jump leads (prototyping wires)
* A breadboard
* A push-button
* An LED of your chosen colour

To build the assembly:

1. First, connect pin 13 to an LED and a resistor (in series) and then to GND.
2. Then connect the switch as shown, with one leg connected to +5V, and the other leg of the switch connected to Pin 2. If you are using a four-pin push button as in the diagram above, make sure you place it across the central dividing line of the breadboard as shown.
3. Finally, connect Pin 2 (and that second leg of the switch) to a resistor then to GND.

*The resistor in step 3 is called a 'pull-down' resistor because it 'pulls' the voltage of Pin 2 down to 0V (GND) when the switch is open. We have included a short additional explanation of this at the bottom of the page, but this isn't part of the main course content.*

For more information on attaching push buttons, look up [push-button](https://www.arduino.cc/en/Tutorial/Button) on the Arduino webpage.

**Code**

This set of instructions will light up an LED attached to pin 13 when the button attached to pin 2 is pressed. When you use the Arduino IDE program, the code can be found by clicking *File→Examples→02.Digital→Button*.

/\*  Button

 Light up an LED on pin 13 while the button on pin 2 is pressed.

 \*/

// Declaring the constants :

const int buttonPin = 2; // Number of the pin connected to the push-button

const int ledPin = 13; // Number of the pin connected to the LED

// Declaring the variables :

int buttonState = 0; // This variable 'buttonState' will be used to store the state of the button

// This code is executed once on boot-up

void **setup**() {

  // ledPin is an output:

  pinMode(ledPin, OUTPUT);

  // buttonPin is an input:

  pinMode(buttonPin, INPUT);

}

// This code is executed over and over again as a loop

void **loop**(){

  // read the state of the button and store in buttonState  
  buttonState = digitalRead(buttonPin);

  // Check to see if buttonState is HIGH (5V), i.e. if the button is pressed

  if (buttonState == HIGH) {

    // light up the LED

    digitalWrite(ledPin, HIGH);

  }

  else {

    // if the button is not pressed, turn off the LED

    digitalWrite(ledPin, LOW);

  }

}

**Instructions**

* **Declaring a constant:** as with a [variable](https://www.arduino.cc/en/Reference/VariableDeclaration), we come with this line to store the value to the right of the equal sign in led.

const int led = 13;

The key word [const](http://arduino.cc/en/Reference/Const" \o "const" \t "_blank) indicates that we don't want the value of 'led' to be able to changed by the program.

* **New instructions**:
* [digitalRead](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.PinMode) reads the state of the pin and returns HIGH if the voltage is at the power supply voltage (in this case 5V) or LOW if the pin is at 0V.

digitalRead(buttonPin);

The value returned by [digitalRead](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.PinMode" \o "pinMode" \t "_blank) can be stored in a variable as follows:

buttonState = digitalRead(buttonPin);

* [if](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.If) lets us test a statement to see if it's true, and then run some code if it is. In our program *Button*, we want to do something "if" the button is pressed, so we're going to compare buttonState with the value HIGH as follows:

if(buttonState == HIGH)

* [else](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.IfElse): the code after this key word will be executed if the preceding test is false. In *Button*, if the button**is not pressed** we want the LED off.

**Pull-down and Pull-up resistors: brief explanation**

*The resistor connected to the switch is called a 'pull-down' resistor: it ensures the voltage we read at Pin 2 is 0V whenever the switch is 'open' (not pressed).*

*When we press the switch, we make an electrical connection between +5V and Pin 2. If we then release the switch, that connection to GND through a resistor allows the voltage at Pin 2 to return to 0V. Without this resistor, Pin 2 might retain an electrical charge we don't want it to, so it might not always read 0V if the switch is open. For more information, see the Arduino Playground page on*[*Pull-up and Pull-down resistors*](https://playground.arduino.cc/CommonTopics/PullUpDownResistor)*.*

**References**

* [Official Arduino reference pages](http://arduino.cc/en/Reference/HomePage) by the Arduino team
* [SparkFun tutorial - blinking an LED](https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/experiment-1-blinking-an-led) by SparkFun