

The button: let Arduino decide

Arduino listens to a button, switches on the LED and writes to you on the computer. With the basic commands explained one by one, plus a bonus: turning the button into a switch.

By **Ada Intelisano** · informaticait.it technical team



Online guide,
always up to date

IN SHORT

- We add a button to the circuit from guide no. 34.
- The sketch reads the button with `digitalRead` and decides with `if ... else`.
- The Serial Monitor shows on the computer what is happening.
- At the end: the notebook with all the basic commands.

Time: 30 minutes

Level: day two

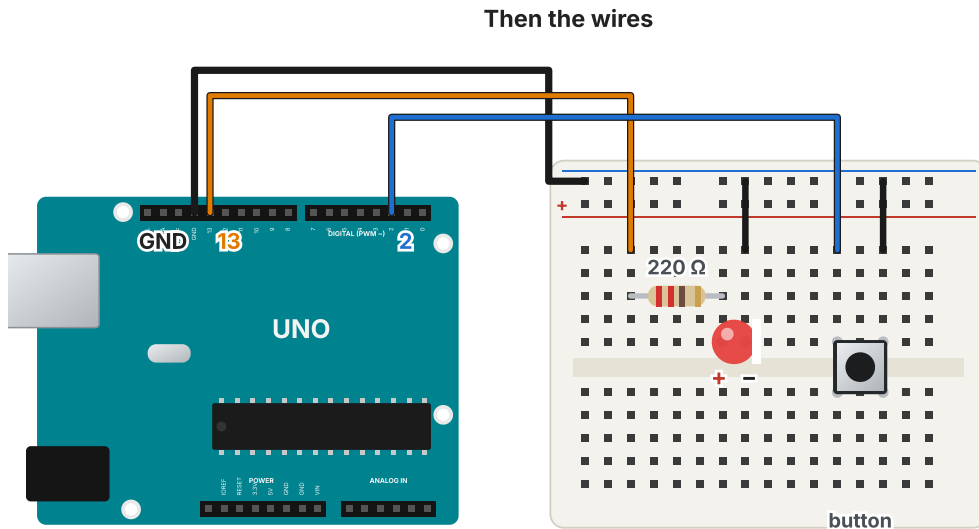
Before: guide no. 34 (your first LED)

In guide no. 34 Arduino switched the LED on all by itself. Now it is the one **listening**: when you press a button it notices, switches on the LED and writes it on the computer. It is the first step for any project that reacts to something.

1 Get the parts ready

- the circuit from guide no. 34 (Arduino, breadboard, LED with the 220 Ω resistor)
- a 4-pin **pushbutton**, the small square kind
- **three more jumper wires**

2 Fit the button and the wires



informaticait.it

The **button** sits across the center channel, like a bridge. Compared with guide no. 34 the ground connection changes: GND goes to the **long row at the top marked –** (the – rail), and from there two short jumpers bring the minus to the LED and to the button.

Orange wire: pin 13 → resistor (as before). **Blue wire:** pin 2 → column of the button's left pin. **Black wires:** GND → – rail; – rail → short leg of the LED; – rail → right pin of the button.

✓ Check before moving on

Following it with your finger: pin 2 → button → – rail → GND. When you press, pin 2 is connected to GND.

3 Write the sketch

```
// The button: as long as you hold it down the LED stays on and the computer says so.
#include <Arduino.h>

const int LED = 13;           // with the ESP32 write 23
const int BUTTON = 2;         // with the ESP32 write 27

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP); // input: at rest it reads HIGH
  Serial.begin(115200);          // opens communication with the computer
  Serial.println("Ready! Press the button");
}

void loop() {
  if (digitalRead(BUTTON) == LOW) { // LOW = pressed
    digitalWrite(LED, HIGH);
    Serial.println("Pressed");
  } else {                          // otherwise: released
    digitalWrite(LED, LOW);
  }
  delay(200);                       // a short pause
}
```

New word: INPUT_PULLUP

It tells Arduino that the pin is an **input** and turns on an internal resistor that holds it HIGH. When you press, the button connects it to GND and the pin reads **LOW**. So: **pressed = LOW**.

New word: if ... else

It is a question: *if* the button is pressed switch on the LED, *otherwise* switch it off. The double equals sign == is used to compare.

New word: Serial

It is the communication line with the computer, through the USB cable. With `Serial.println` the board writes messages to you.

4 Upload and open the Serial Monitor

Upload as in guide no. 34 (right-pointing arrow). Then open the **Serial Monitor**: it is the magnifying glass icon at the top right. At the bottom of the monitor, choose the speed **115200**, the same as in the sketch.

✓ What you should see

The monitor shows "Ready! Press the button". Hold the button down: the LED comes on and lots of "Pressed" lines appear. Let go: the LED goes off.

If it doesn't work

- The LED is always on: the button is turned the wrong way and always connects. Rotate it a quarter turn.
- Nothing happens when you press: check the blue wire on pin **2** and that the button sits across the channel.
- The monitor shows strange symbols: the speed at the bottom is not 115200.
- The monitor is empty: press the RESET button on Arduino to start over.

5 Bonus: one press on, one press off

Now the button becomes a switch, like a light switch. You need a **variable** that remembers whether the LED is on.

```
// One press switches on, one press switches off: the button becomes a switch.
#include <Arduino.h>

const int LED = 13;           // with the ESP32 write 23
const int BUTTON = 2;         // with the ESP32 write 27
bool ledOn = false;           // remembers whether the LED is on

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
}

void loop() {
  if (digitalRead(BUTTON) == LOW) {           // you pressed it
    ledOn = !ledOn;                           // the exclamation mark flips it: on becomes
off and vice versa
    digitalWrite(LED, ledOn ? HIGH : LOW);
    delay(300);                               // pause: one press counts only once
    while (digitalRead(BUTTON) == LOW) {}      // wait until you let go of the button
  }
}
```

New word: variable

A box with a name that holds a value. `bool ledOn = false;` creates a box that is either true or false, and at the start it is false (off).

✓ What you should see

Every press changes the LED: on, off, on...

6 Challenge yourself

Try it yourself: three challenges

- Do the opposite: LED **on** when the button is released and off when you press it.
- While you hold the button down, make the LED **blink** instead of staying on (you need delay from guide no. 34).
- Count the presses and write them in the monitor: "Press number 1", "2"... (you need an int variable).

7 And with the ESP32?

Same circuit, but: LED on pin **23**, button on pin **27**, ESP32 GND to the – rail. In the sketch change the two lines: `const int LED = 23;` and `const int BUTTON = 27;`. Board to choose: **ESP32 Dev Module**.

The command notebook

All the commands from guides no. 34 and 35, to keep at hand:

Command	What it does
<code>pinMode(pin, OUTPUT)</code>	the pin gives out current: for LEDs, buzzers, relay modules
<code>pinMode(pin, INPUT_PULLUP)</code>	the pin listens: for buttons and switches connected to GND
<code>digitalWrite(pin, HIGH) / LOW</code>	switches on / off
<code>digitalRead(pin)</code>	reads the pin: HIGH or LOW
<code>delay(ms)</code>	waits, in thousandths of a second (1000 = 1 second)
<code>if (...) { ... } else { ... }</code>	if it is true it does the first part, otherwise the second
<code>Serial.begin(115200)</code>	opens communication with the computer
<code>Serial.println("text")</code>	writes a line in the Serial Monitor

✓ What you learned

- a button is connected between a pin and GND, with INPUT_PULLUP: pressed = LOW;
- `digitalRead` reads, `if ... else` decides;
- the Serial Monitor lets you see what the board is thinking;
- a variable remembers things from one round of `loop` to the next.

Sources: Arduino, "Digital Pins" documentation (INPUT_PULLUP) and language reference · informaticait guides no. 20 (pull-ups and bounce), no. 26 (ESP32 pins), no. 34 (your first LED)