ROBOTICS AND CODING

ARDUINO

PATRA 2023

After installing the Arduino IDE we can write our first pieces of code. Arduino logic is very simple - there are basically two main functions : setup() , and loop(), which work as follows:

- **setup()** - here we put all the commands that must be run once, when our arduino is activated (that is, when we supply power or when the reset button is pressed). Initialization of variable values are usually included and definitely the characterization of the inputs/outputs we will use (when one specific Pin will be input or output).

- **loop()** - this is where we write our program. The existing commands will be run, when it reaches the end, the loop() will be activated again, continuing from the beginning, and again. This will happen continuously as long as the arduino has power or until the button reset is pressed.

So the basic operation of the arduino is that it runs the setup() function once at the beginning and then the loop() function, over and over again until we shut it down (no power) or to press the reset button. In the case of Reset, the setup() function is rerun once and then the loop() again and again, just like when the microcontroller is initially powered up.

A typical program has the following structure:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

The use of breadboard for connections and short circuits

Because circuits usually include several elements (wires, leds, sensors, resistors, etc.) it is recommended to use a **breadboard** so that they can be easily connected the elements through it. Also, functionally the breadboard can short-circuit between them cables, which is very useful when we have a lot of cables to short. In Figure 1 shows a typical breadboard.

Notice that the horizontal + and - lines on each side (top and bottom, in red and blue color as we see it) are short-circuited with each other, while in the columns (which are usually numbered 1 through 30) the five vertical slots are shorted (usually with letters a, b, c, d, e as well as f, g, h, i, j) to each other in each column as we look.

So, for example we can connect to – the GND of the Arduino and to the breadboard we connect all the GND returns of our circuits.

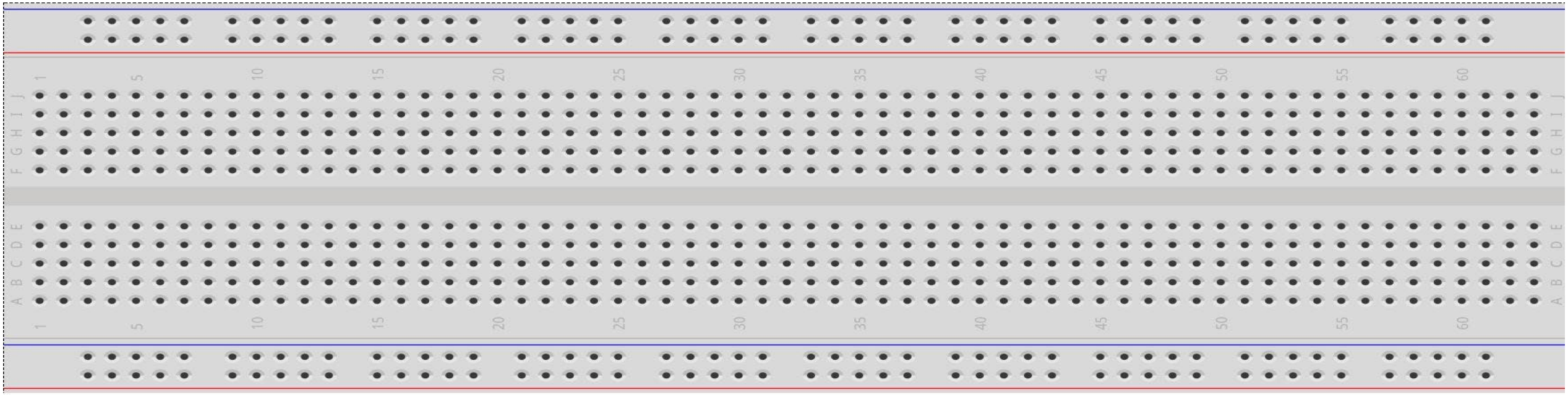

Figure 1 – Breadboard

Also, through the short-circuited letters (5 in each column as we look), we can connect the parts of our circuit. Figure 2 shows a simple example of connection two LED.

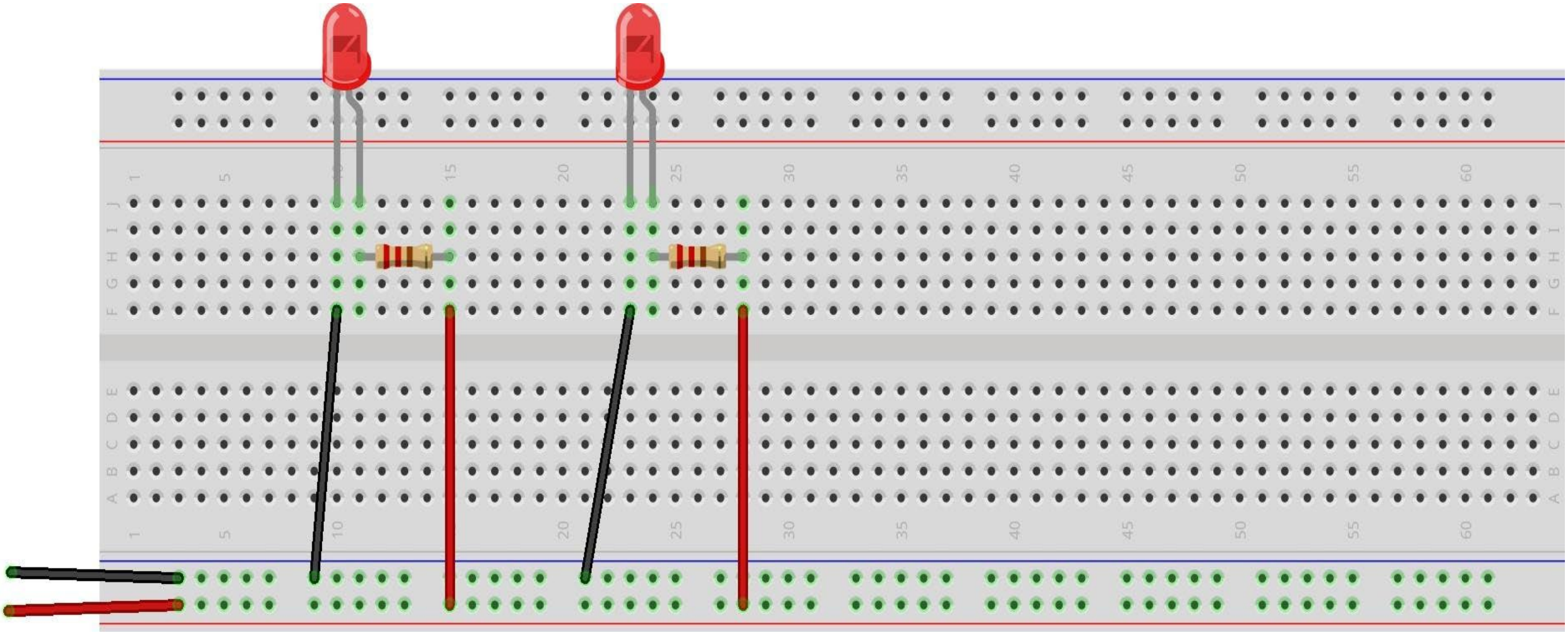

Figure 2 – Breadboard usage example

digitalRead()

Description: Reads the value from a specified digital pin, either HIGH or LOW.

Syntax: **digitalRead(pin)**

Parameters: **(pin)** the Arduino pin number you want to read

Returns: **HIGH** or **LOW**

IF statement

In programming, many times we will need to check some condition to decide whether to execute a piece of code or to execute something else instead it. We achieve this by using the “If” statement, which is drawn up:

if (condition) { commands }

where, in the (condition) we have the checking we want to be done, usually using the comparison operators:

(>, <, =, <>, >=, <=) e.g. potVal > 500.

The condition can be more complex, using the logical operators (|| for OR, && for AND),

e.g. (potVal > 500) && (timePass >= 1000).

In the { commands } clause, the commands we want to be executed.

If <condition> is TRUE, <commands > will be executed, if <condition> is FALSE, the arduino will execute the first command after the close curly bracket of the “If” statement.

Time logging function - millis()

The Arduino has a built-in clock, which counts the time from the moment it is turned on (or it is reset). This information is available to us at any point by calling the function millis(), which returns the time in milliseconds (1/1000 sec) it has gone from the activation of our unit. This helps us measure time in our programs, especially in cases where we want to "remember" things.

For example, if we want to check if a certain amount of time has passed since something happened (e.g. a key was last pressed), we can log the event to a time variable and subtract this time from the next one, etc.

For example:

```
lastPress = millis();

if (lastPress – millis() > 1000) {...}
```