Tecnologia da Informática

Licenciatura em Engenharia Informática

Universidade de Coimbra

2023/2024

Resolução dos exercícios 1.2 e 1.3

Exercício 1.2 - O Arduino suporta uma função *random()*, que permite gerar números pseudo-aleatórios, e cuja sintaxe está descrita na documentação *online*[1]. Crie um programa que produza números aleatórios entre 10 e 100, imprima esses números e indique se são pares (imprimindo “1” para par e “0” para ímpar).

Exercício 1.3 - Há uma variante da função *Serial.println* que aceita dois argumentos (confira na referência oficial[2]). Utilize esta variante para criar um programa que conte números inteiros, um a um e imprima os valores em decimal, hexadecimal e binário.

Conteúdo

- Execução condicional
- *Digital Output*: recordando a primeira aula
- *Digital Input*
- Exemplo prático
- Leituras adicionais

Execução condicional

Estruturas de controlo: parte I

Execução condicional

Existe uma instrução ***if*** para execução condicional.

Formato:

```
if (condição) {
  //Instruções a executar se a condição for verdadeira (TRUE)
} else {
  // Instruções a executar se a condição for falsa (FALSE)
}
```

Exemplo 1 com instrução if

```
int a = 12, b = 12, c = 13;

void setup(){
  Serial.begin(9600);

  if (a == b){
    Serial.println(“a é igual a b ”);
  }

  if (a == c){
    Serial.println(“a é igual a c ”);
  } else {
    Serial.println(“a é diferente de c ”);
  }

  c = 12;
  if (a == c){
    Serial.println(“a é igual a c ”);
  } else {
    Serial.println(“a é diferente de c ”);
  }
}

void loop(){ }
```

Exemplo 2 com instrução if

```
int a = 12, b = 12, c = 13;

void setup(){
  Serial.begin(9600);

  if (a <= c){
    Serial.println(“a é menor que c ”);
  }

  if (c > a){
    Serial.println(“c é maior que c ”);
  } else {
    Serial.println(“a é maior ou igual que c ”);
  }

  c = 12;
  if (c > a){
    Serial.println(“c é maior que a ”);
  } else {
    Serial.println(“a é maior ou igual que c ”);
  }
}

void loop(){ }
```

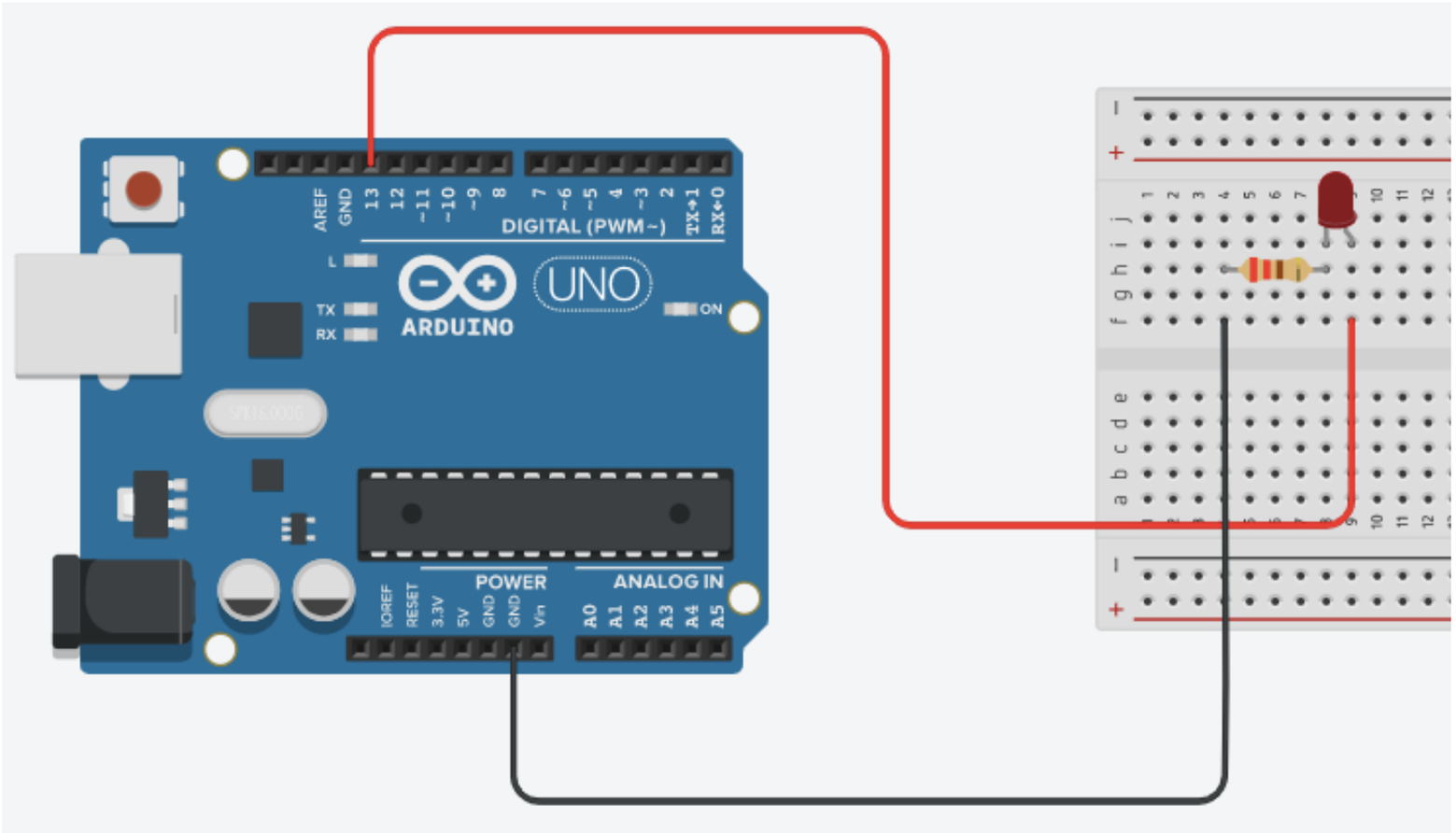
E se quiser executar condicionalmente um bloco de instruções?

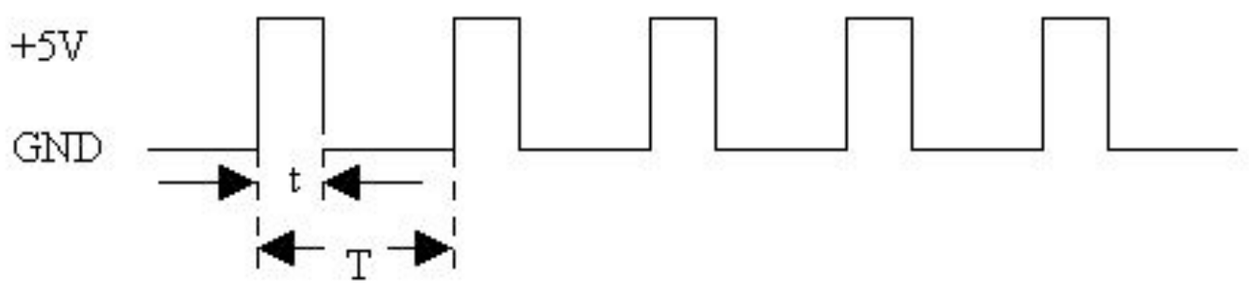
```
int a = 1, b = 13;

void setup(){
  Serial.begin(9600);
}

void loop(){
 if (b > a){
    Serial.println(“b é maior que a ”);
     a=a+1;
  } else {
    Serial.println(“a é igual a b ”);
    Serial.println(a);
    Serial.println(b);
  }
}
```

Input/Output (I/O) digital

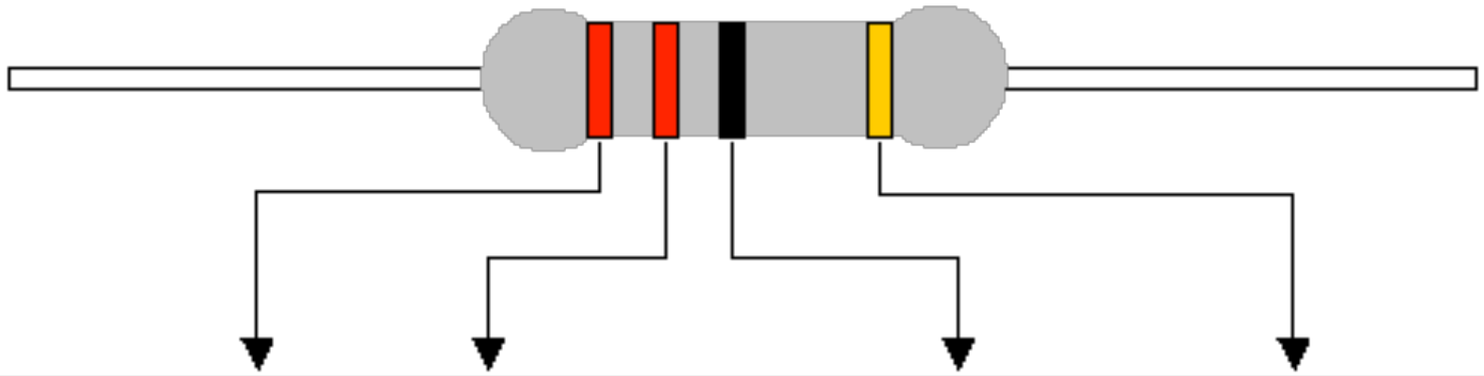
Recapitulando

Nas aulas anteriores, trabalhámos com um exemplo de output digital !

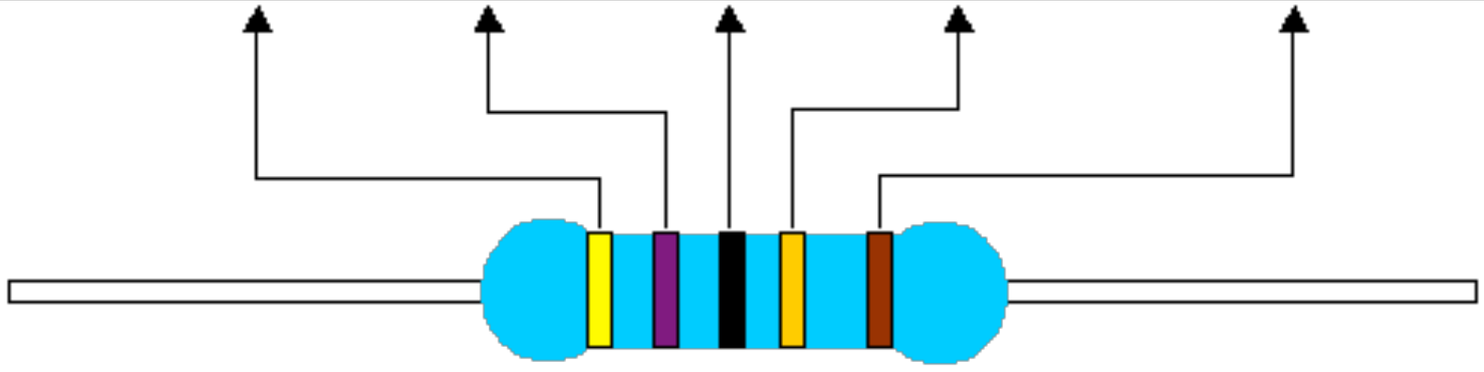




Códigos das resistências



COLOR	1ST BAND	2ND BAND	3TH BAND	MULTIPLIER	TOLERANCE	
BLACK	0	0	0	1		
BROWN	1	1	1	10	± 1%	F
RED	2	2	2	100	± 2%	G
ORANGE	3	3	3	1K		
YELLOW	4	4	4	10K		
GREEN	5	5	5	100K	± 0.5%	D
BLUE	6	6	6	1M	± 0.25%	C
VIOLET	7	7	7	10M	± 0.10%	B
GREY	8	8	8		± 0.05%	A
WHITE	9	9	9			
GOLD				0.1	± 5%	J
SILVER				0.01	± 10%	K
PLAIN					± 20%	M



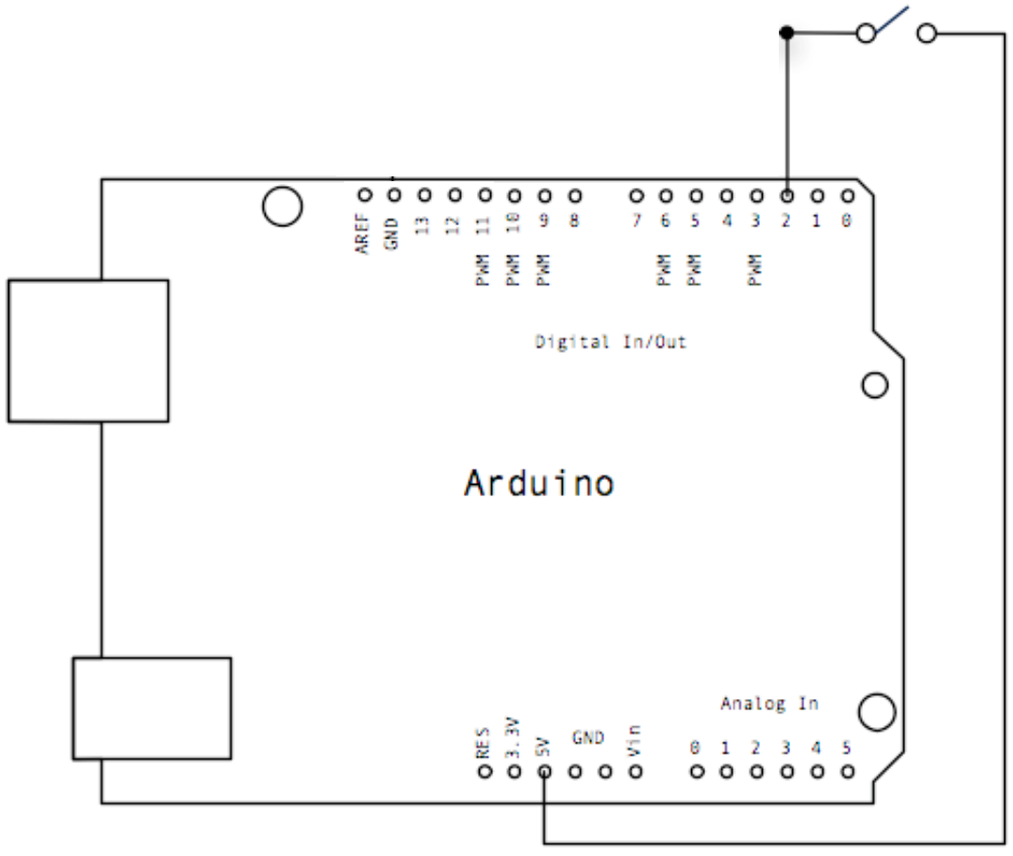
Nesta aula vamos fazer algo diferente

Vamos ligar um botão de pressão ao Arduino, para um exemplo de *Entrada (Input) Digital* .

Um botão de pressão pode ser ligado ao Arduino, usando os pinos digitais para obter o seu estado (verificar se foi premido).

Mas há um problema: não podemos criar um circuito convencional entre o pino de +5V e a entrada digital.

A entrada digital flutua quando não há sinal sendo aplicado - ou seja, retornará aleatoriamente HIGH ou LOW.



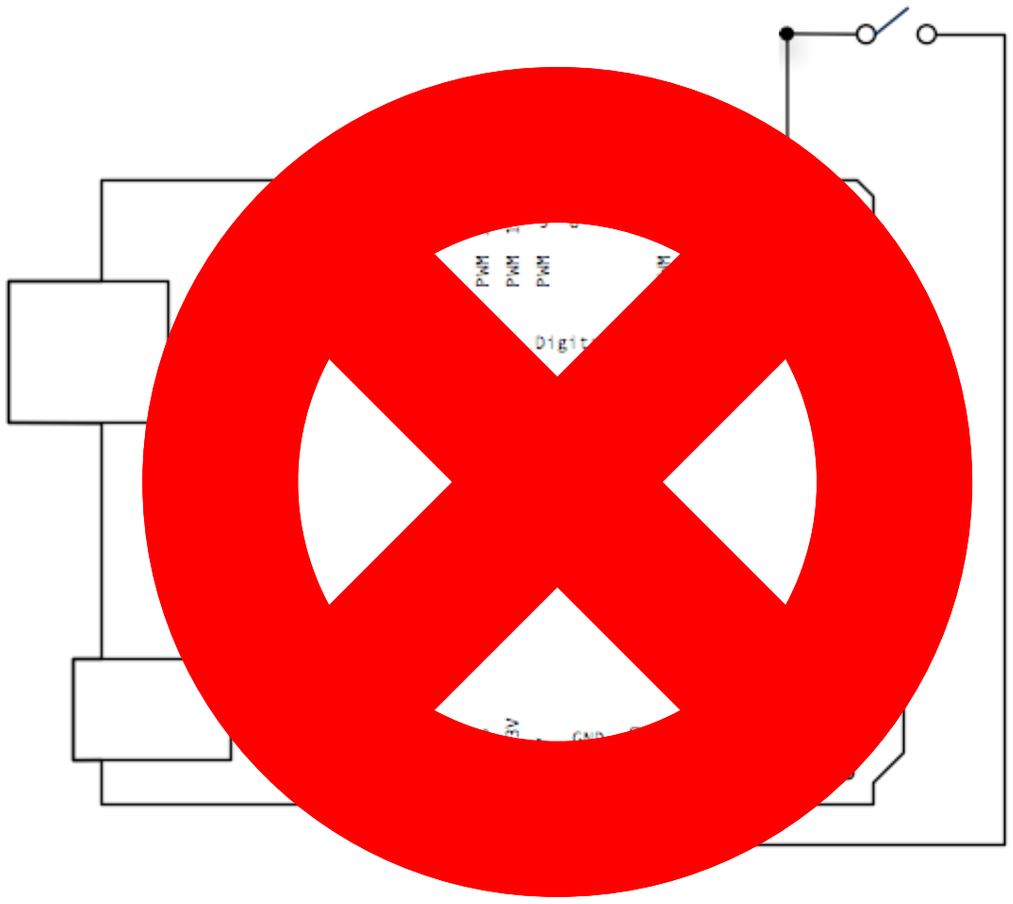
Nesta aula vamos fazer algo diferente

Vamos ligar um botão de pressão ao Arduino, para um exemplo de *Entrada (Input) Digital* .

Um botão de pressão pode ser ligado ao Arduino, usando os pinos digitais para obter o seu estado (verificar se foi premido).

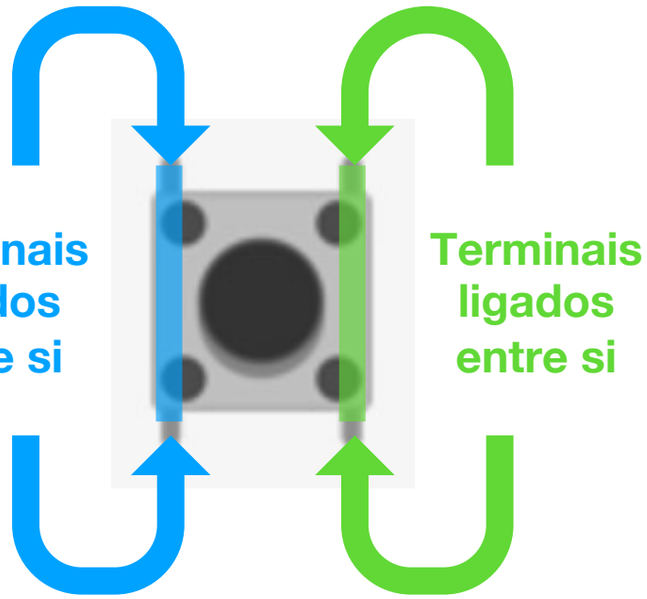
Mas há um problema: não podemos criar um circuito convencional entre o pino de +5V e a entrada digital.

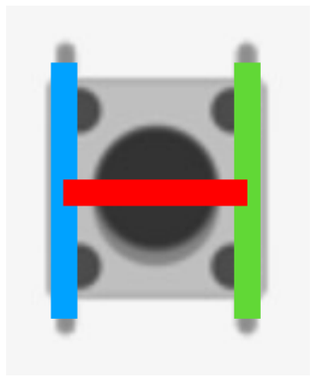
A entrada digital flutua quando não há sinal sendo aplicado - ou seja, retornará aleatoriamente HIGH ou LOW.



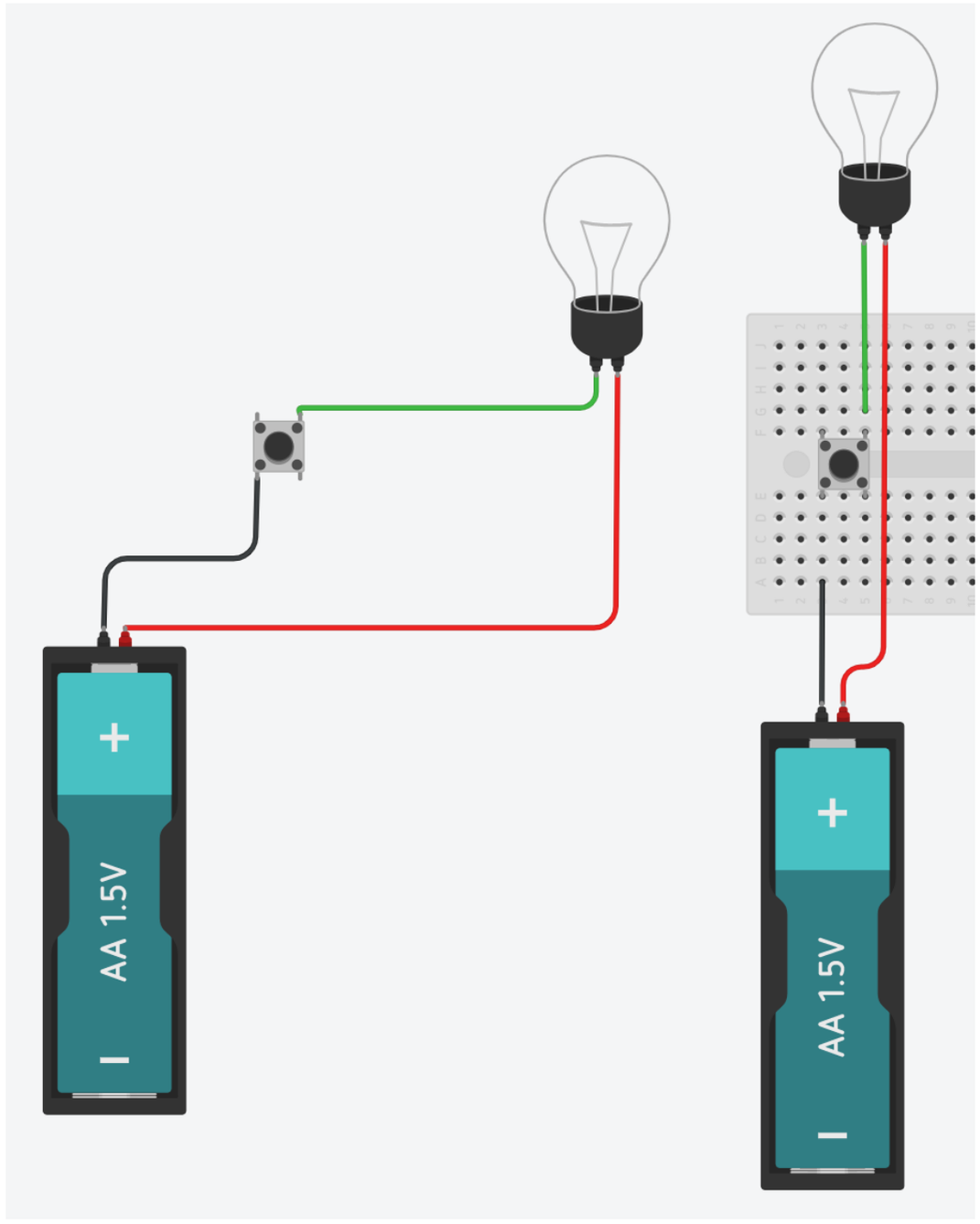
Uma nota breve (de passagem)

Como funciona um botão de pressão (*pushbutton*) – uma demo no tinkercad





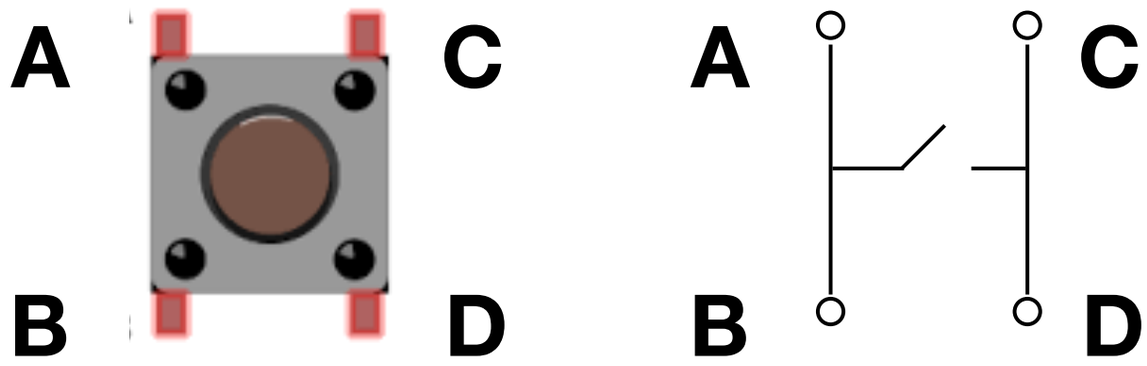
Quando o botão é premido, os terminais da direita são ligados aos da esquerda.



Podem experimentar a demo em:
https://www.tinkercad.com/things/gEnDcJeuwQD

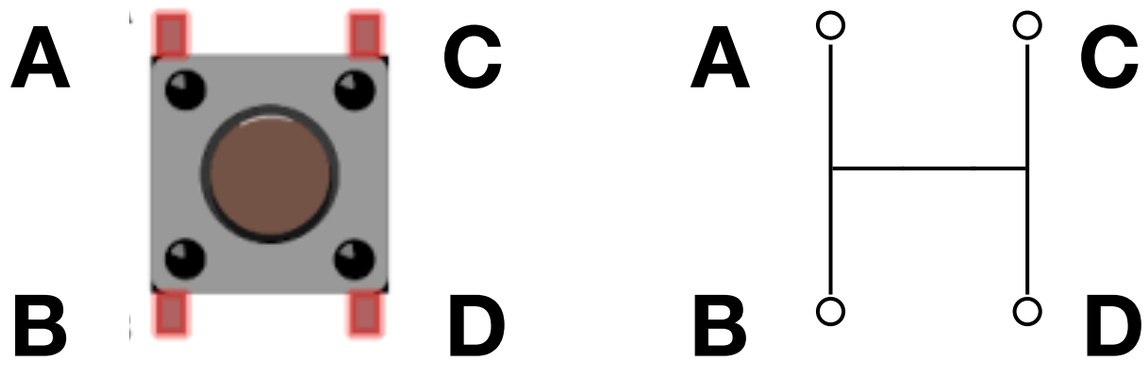
Operação do *pushbutton*

Aberto / Não premido



Operação do *pushbutton*

Fechado / premido

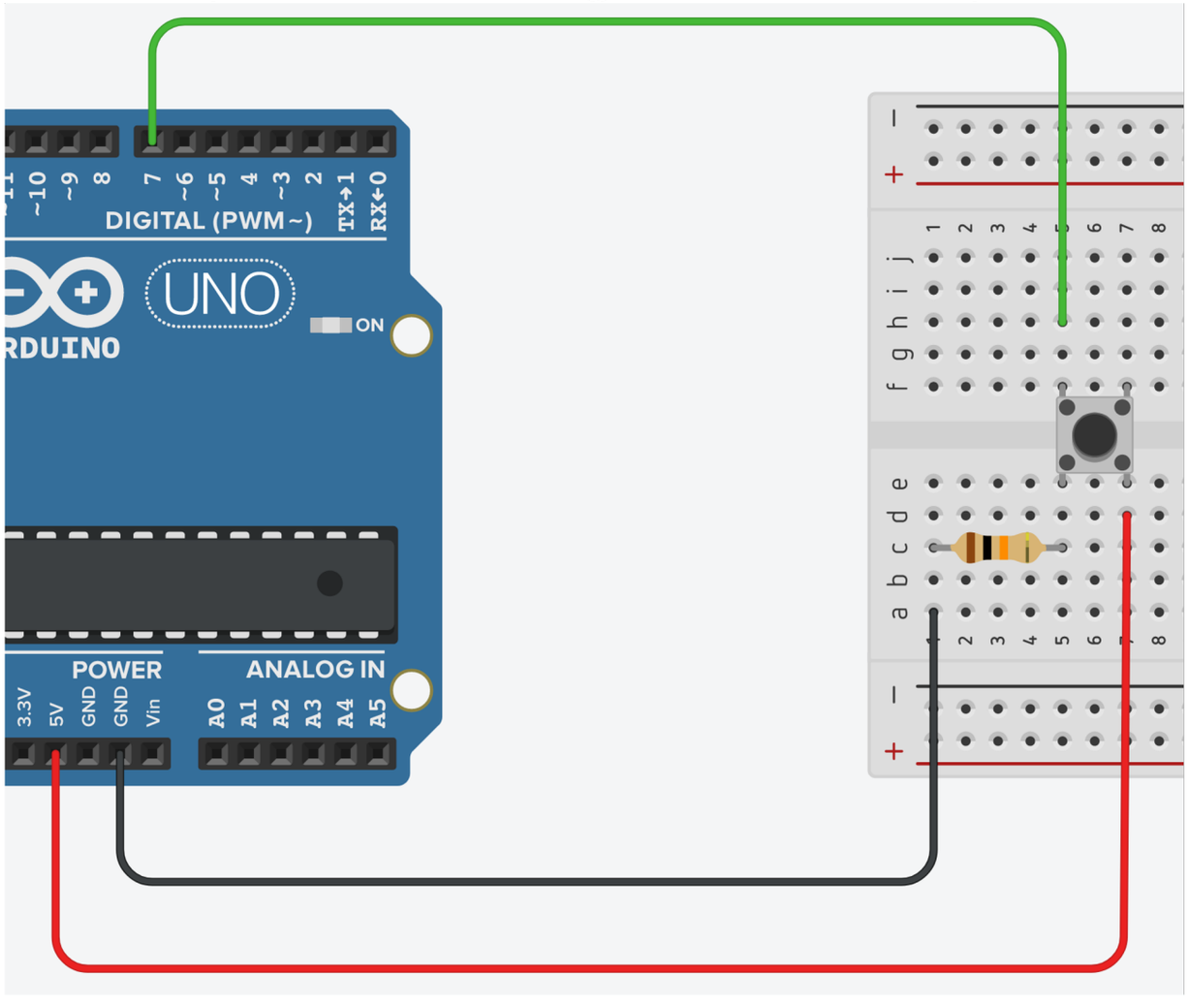


Pull-Down and Pull-Ups

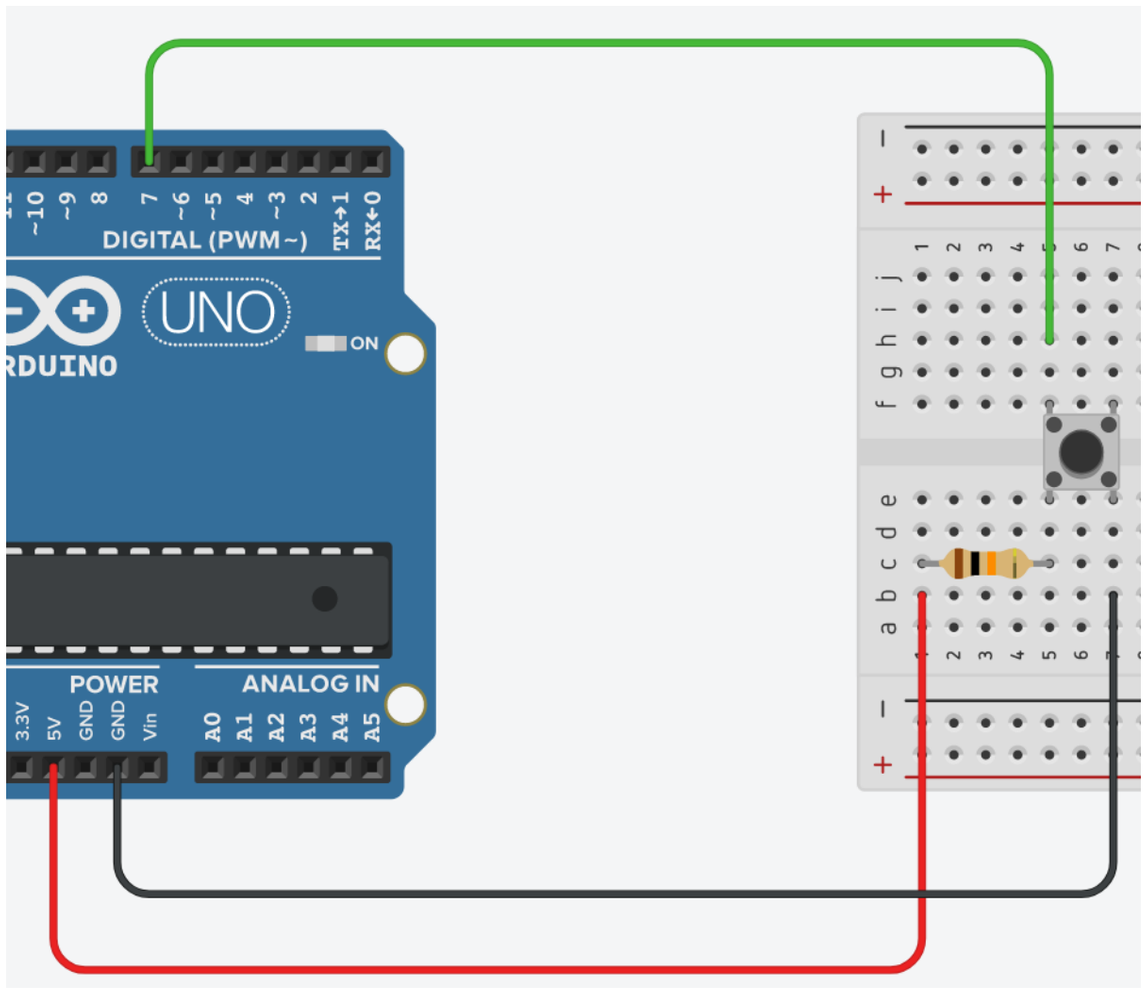
Pushbutton pode ser ligado com resistências configuradas em:

- *Pull-Down*
- *Pull-Up*

Pull Down
(resistência ligada ao GND)



Pull UP
(resistência ligada aos +5V)



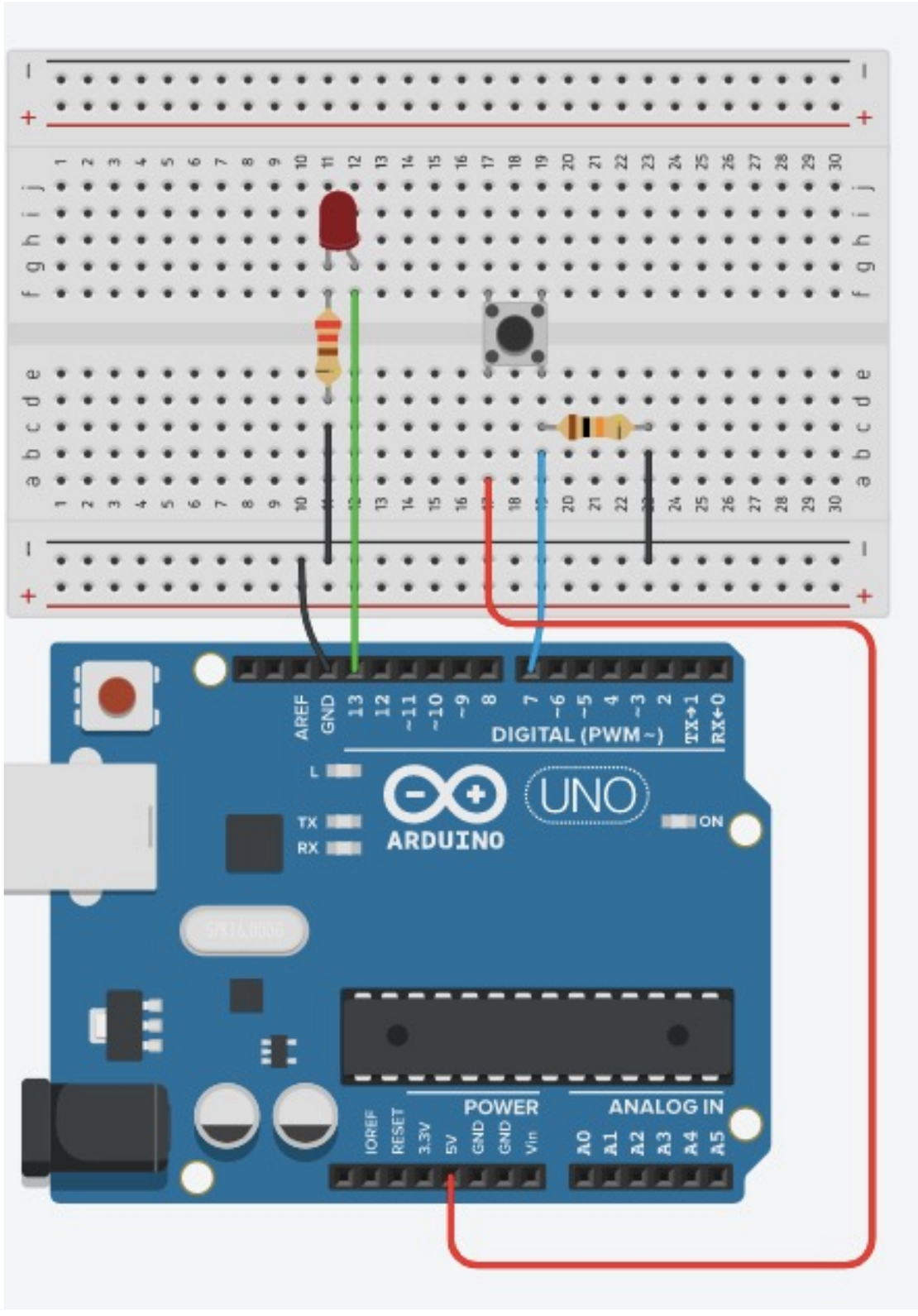
Pushbutton/Digital Input

Esta é a maneira *by the book* para ligar um botão ao Arduino (não necessariamente a que vamos utilizar).

Componentes:

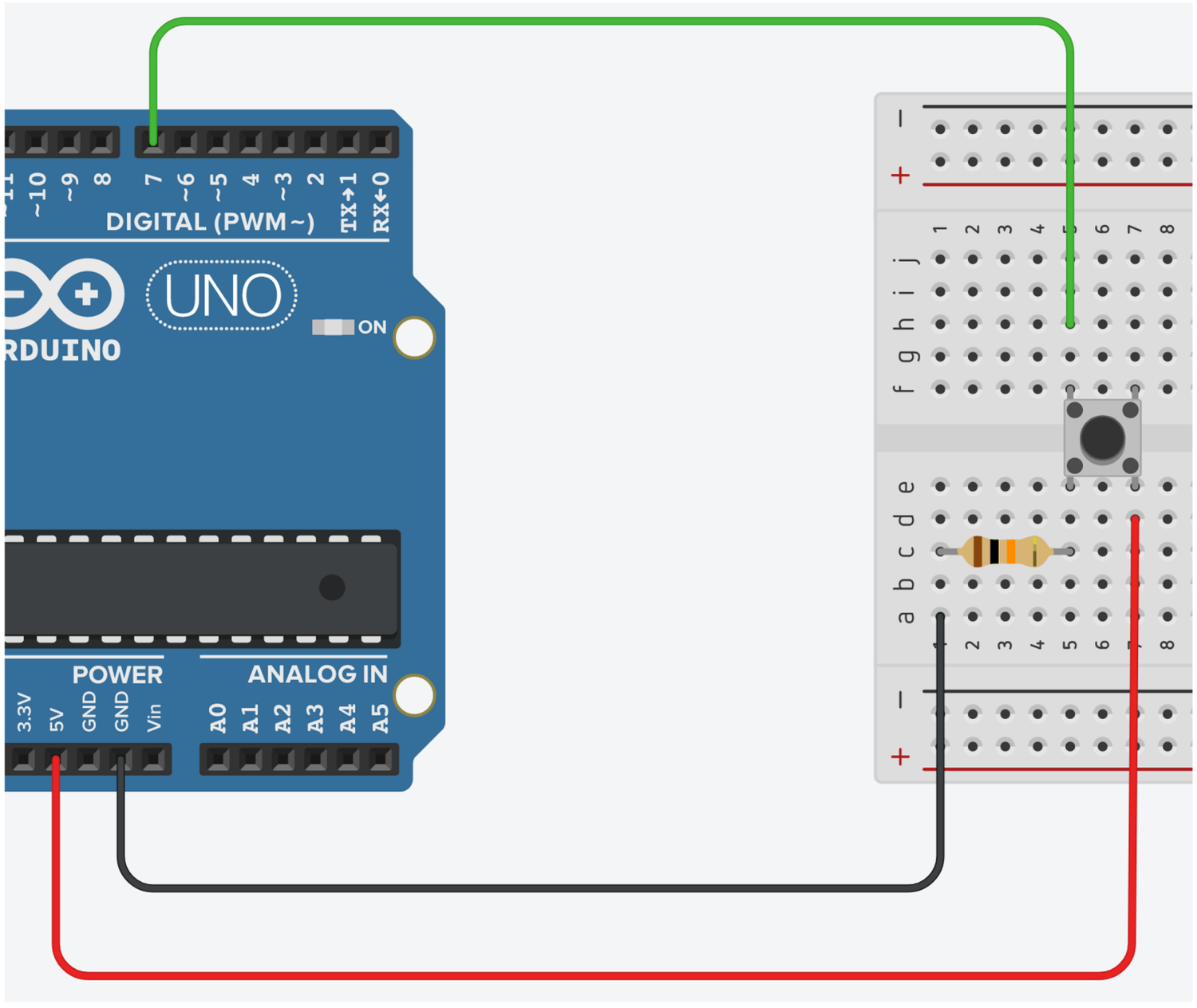
- PushButton
- Resistência *pull-down* de 10 KOhm – (castanho, preto, laranja)

A resistência *pull-down* mantém o nível de entrada digital BAIXO (LOW) quando o circuito está aberto.



Que faz a resistência de pull-down ?

Quando o botão não está premido, o nível de potencial no pino 7 é GND (LOW).

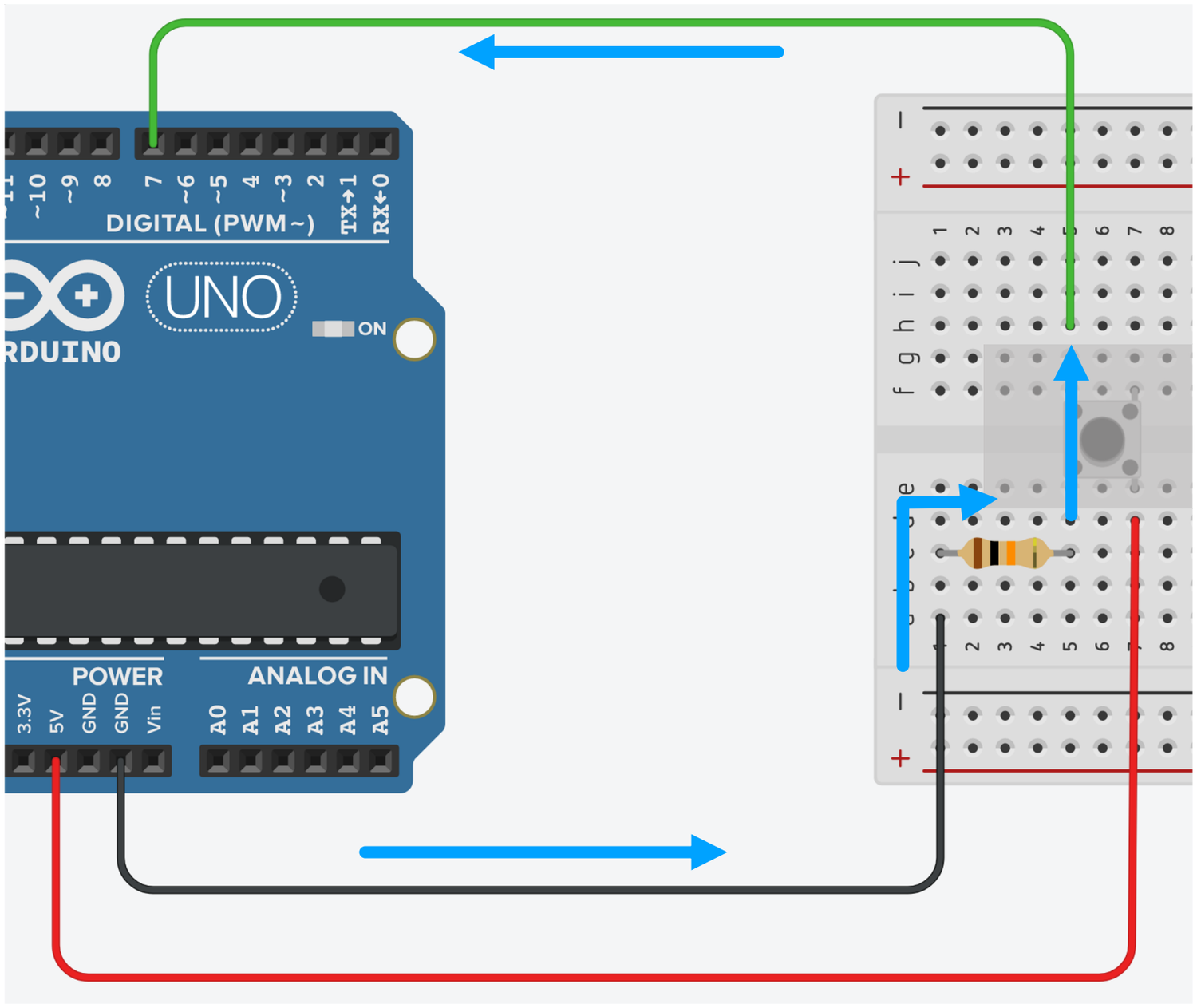


Que faz a resistência de pull-down ?

De facto, os dois terminais do lado esquerdo do pushbutton estão permanentemente ligados entre si...

Logo, o nível de potencial no pino 7 é o GND (LOW).

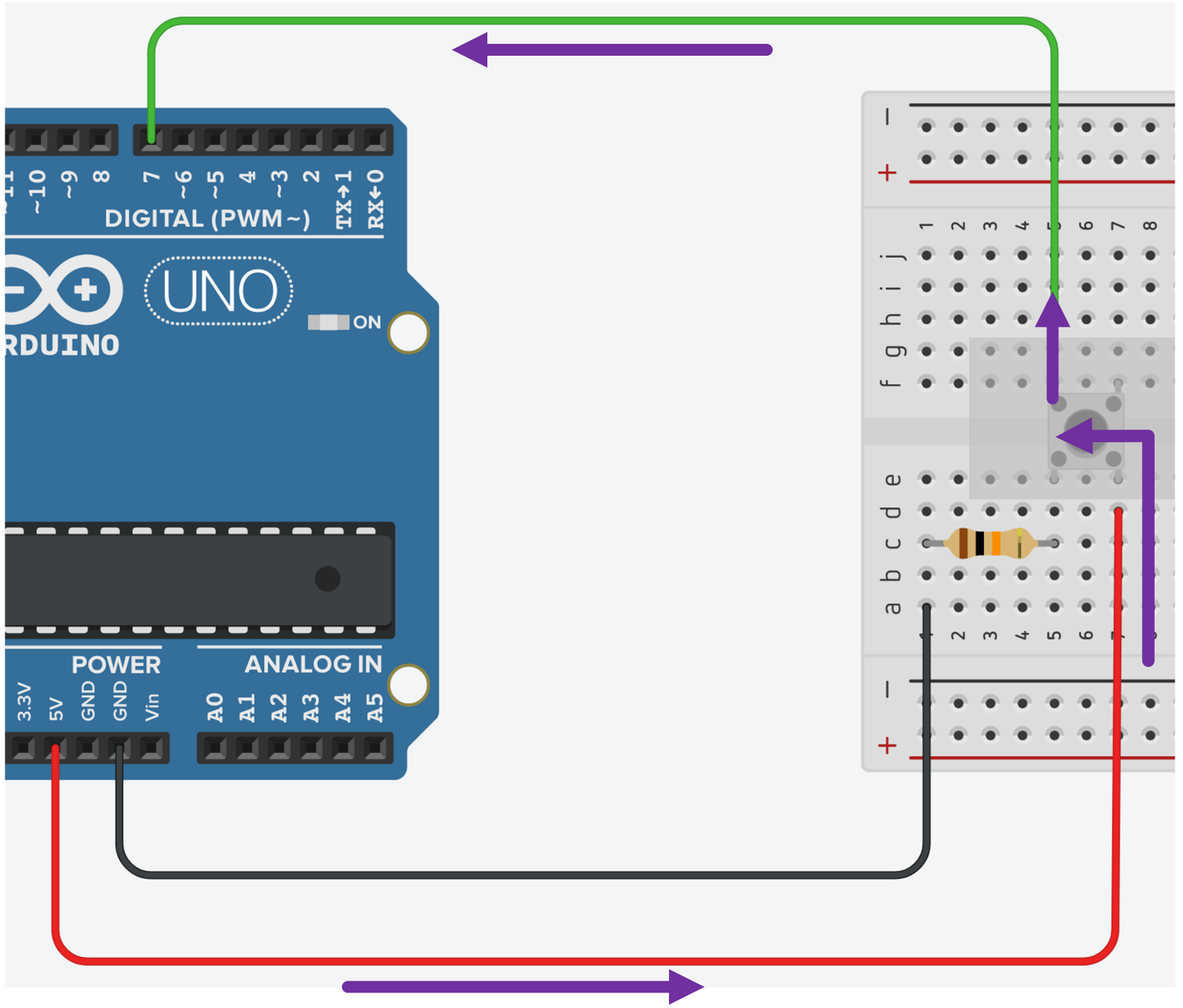
Ou seja, a resistência mantêm o nível de potencial no pino em LOW (puxa-lo para baixo, daí a sua designação).



Que faz a resistência de pull-down ?

Quando o botão é premido, os pares de terminais do lado direito e esquerdo ficam ligados entre si !

Logo, o nível de potencial no pino 7 é +5V (HIGH).

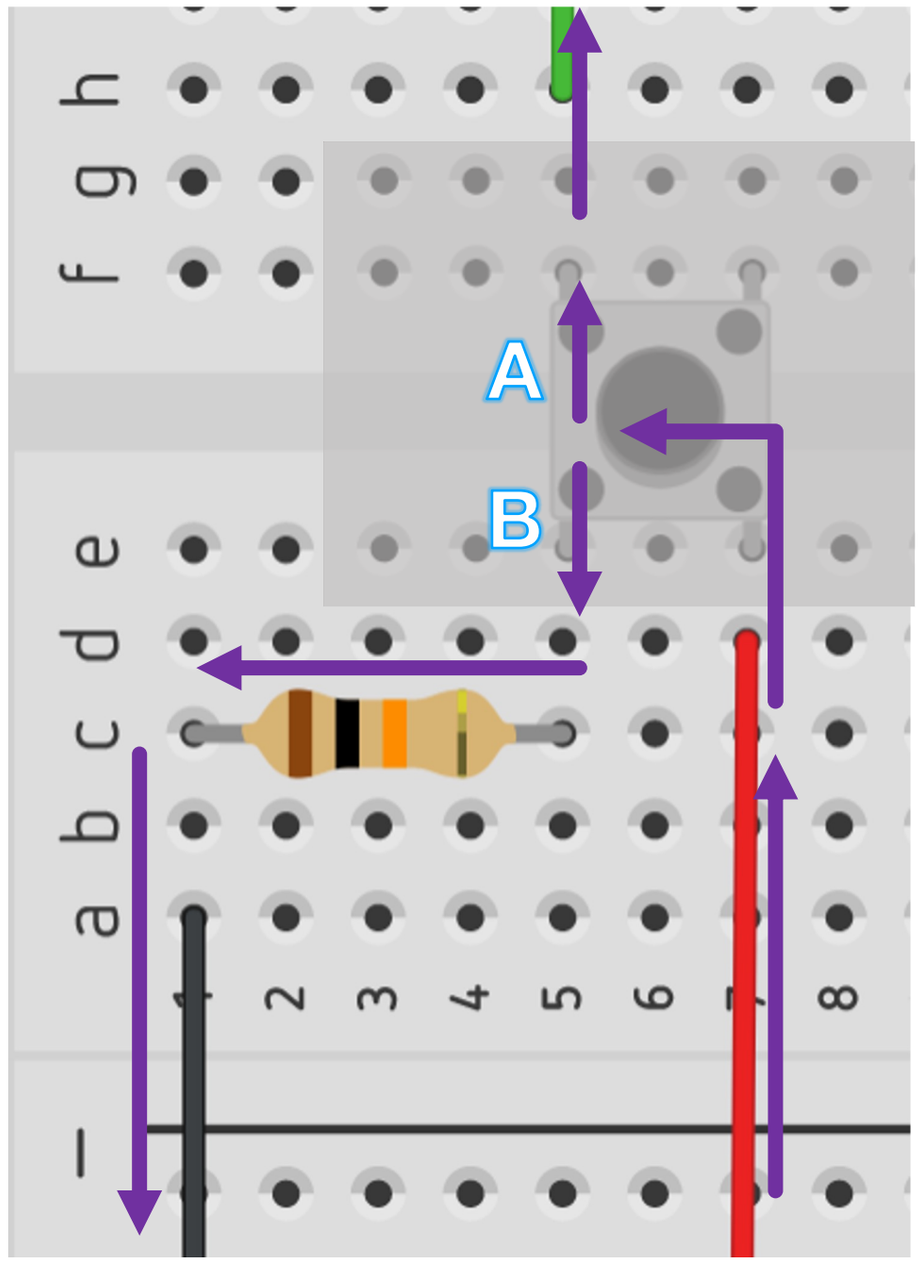


Que faz a resistência de pull-down ?

Sem a resistência de 10K, o que iria suceder ao premir o botão é que iríamos gerar um **curto circuito** !

Porque, ao premir o botão, na realidade é como se a corrente elétrica encontrasse dois caminhos: um para o pino 7 (A) e outro direto para o GND (B).

Sem pull-down, a resistência interna do pino digital do arduino (7, neste caso) seria sempre superior à do caminho que liga ao GND, criando-se assim uma ligação direta 5V -> GND, capaz de destruir o Arduino. Graças à resistência pull-down de 10K, isso não sucede.

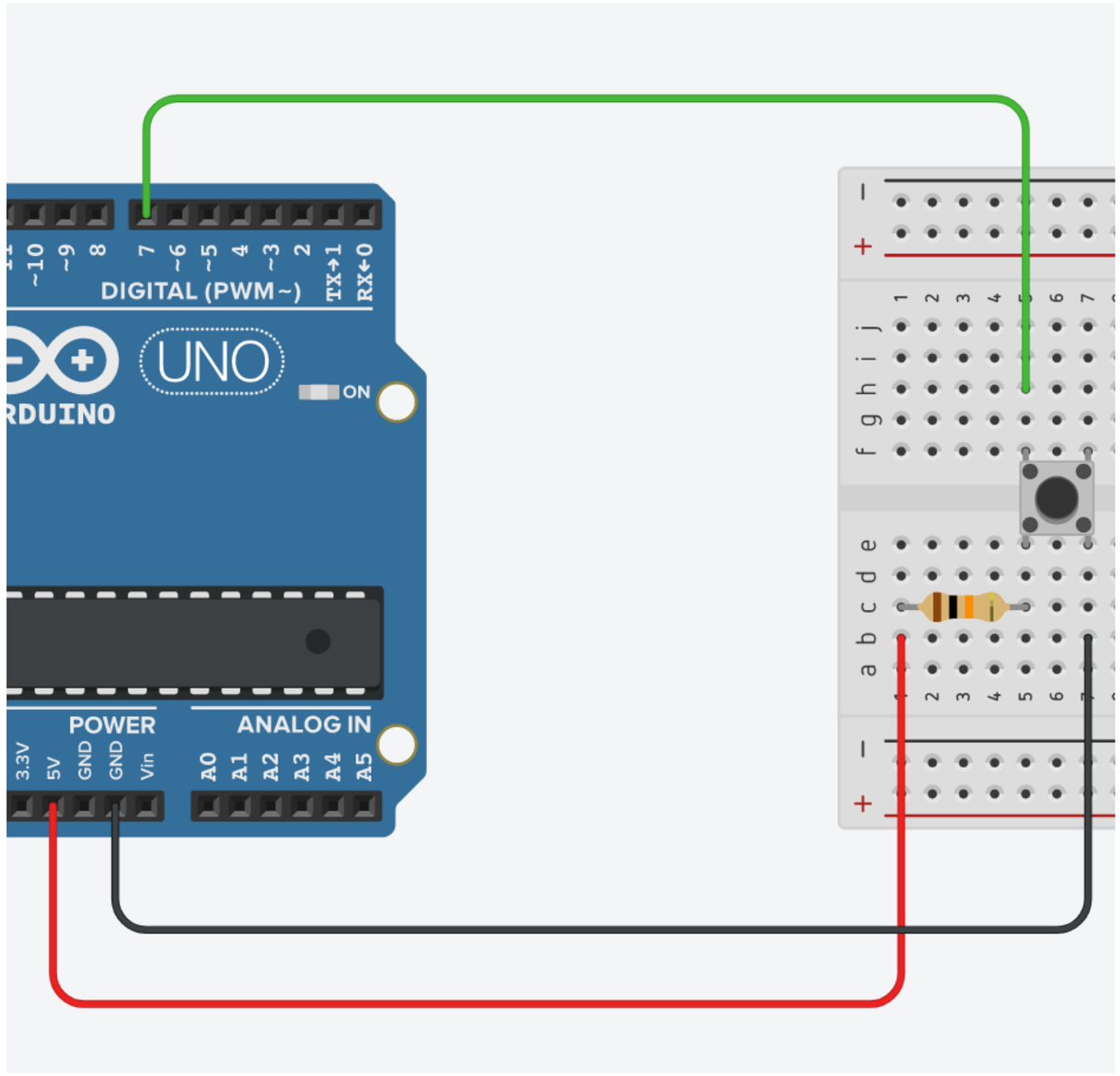


Curiosidade – pull-ups

Uma resistência *pull-up* desempenha uma função análoga ao *pull-down*, com a diferença de *puxar* o estado do pino para HIGH (daí o nome).

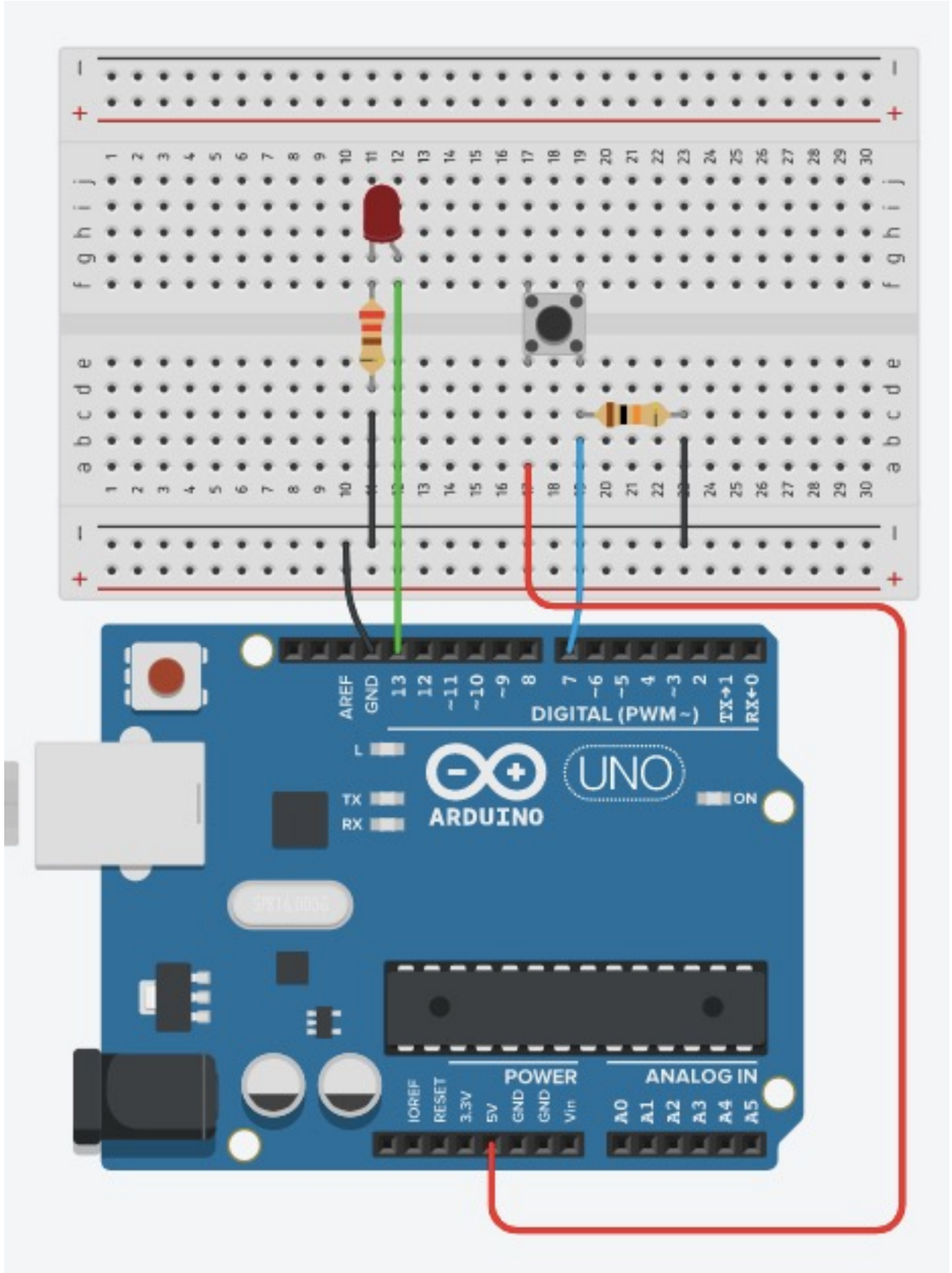
Nesta configuração, o pino de +5V passa a estar ligado à resistência (que, de facto, irá garantir que o potencial do pino estará ao nível HIGH quando o botão não for premido).

Quando o botão for premido, o potencial do pino 7 passará para LOW (GND).



Pushbutton/Digital Input (cont.)

```
int buttonPin=7;
int ledPin=13;

void setup()
{
     pinMode(ledPin, OUTPUT);
     pinMode(buttonPin, INPUT);
}

void loop(){
     int buttonState = digitalRead(buttonPin);
     digitalWrite(ledPin, buttonState);
}
```



Nota:

digitalRead(pino) devolve HIGH (1) ou LOW (0) conforme o valor lido corresponda a um 1 ou 0 lógico.

Consultando a referência...

digitalRead()

[Digital I/O]

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

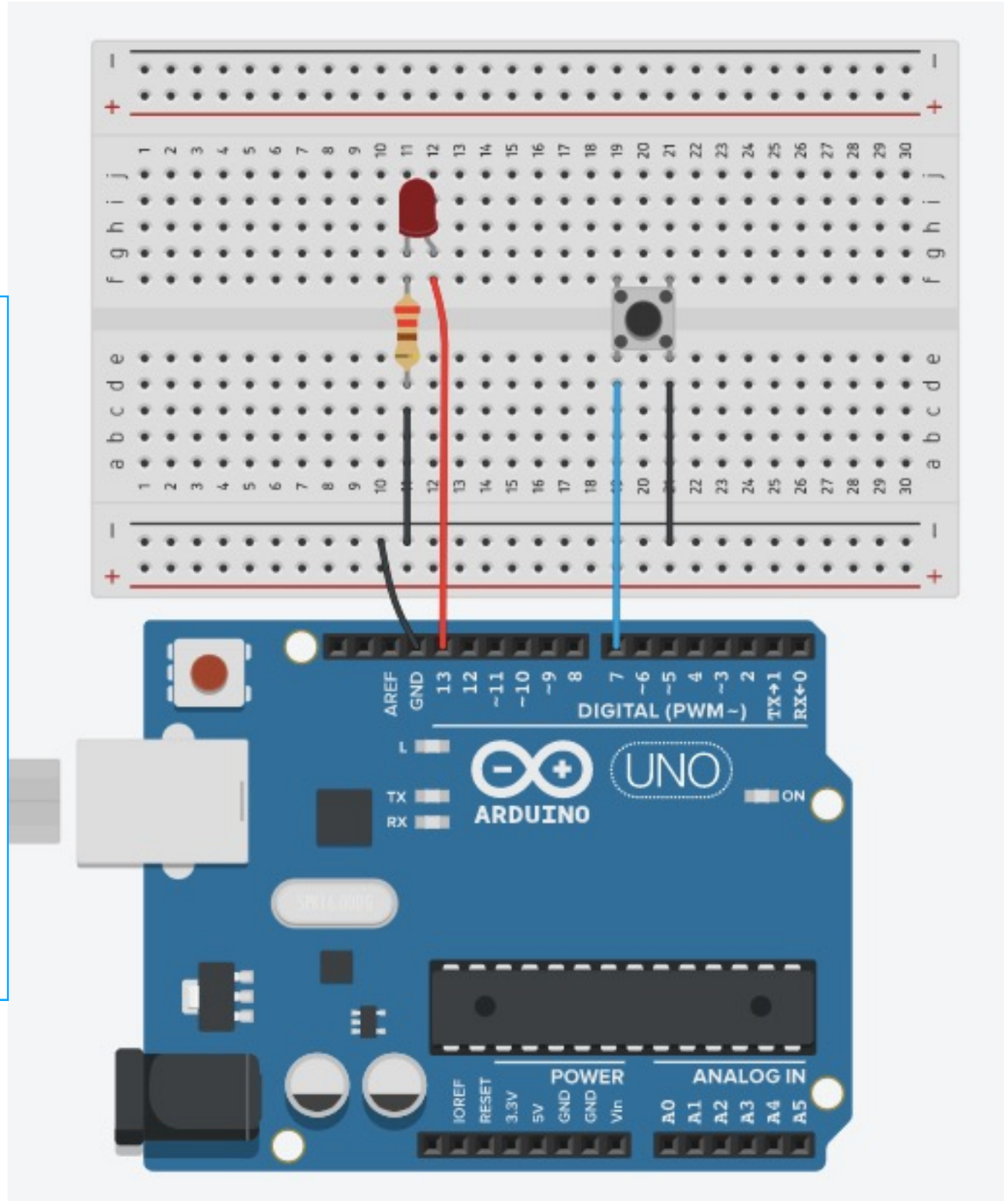
Returns

`HIGH` or `LOW`

Pushbutton/Digital Input – Montagem a realizar

O circuito pode ser simplificado, retirando a resistência *pull-down* e utilizando as **resistências *pull-up* embebidas no Arduino**.

```
int buttonPin=7;
int ledPin=13;

void setup()
{
     pinMode(ledPin, OUTPUT);
     pinMode(buttonPin, INPUT_PULLUP);
}

void loop(){
     int buttonState = !digitalRead(buttonPin);
     digitalWrite(ledPin, buttonState);
}
```

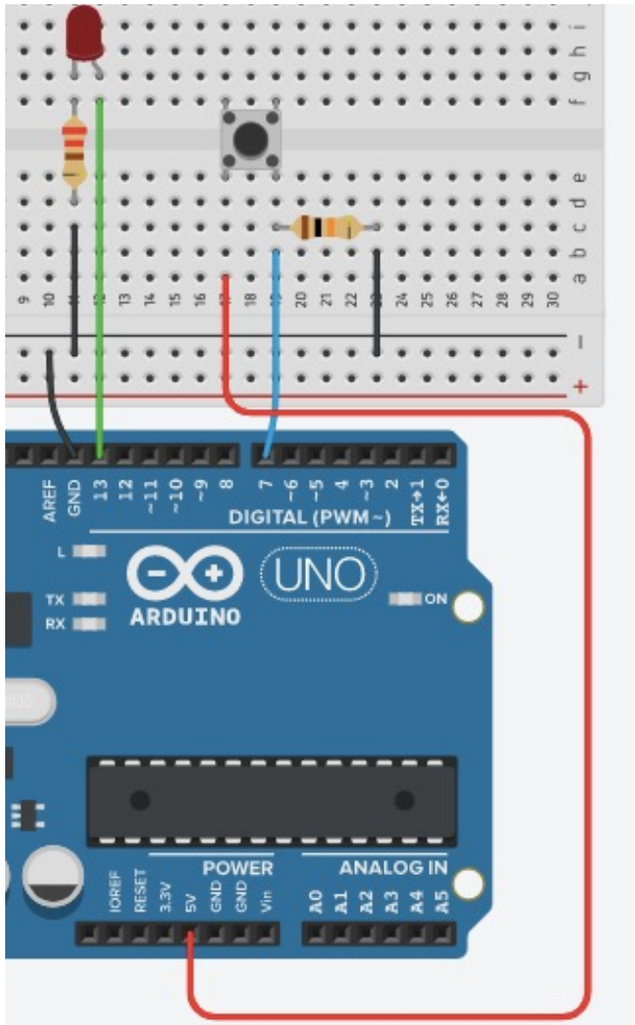


Mas há uma diferença no comportamento. Qual ?

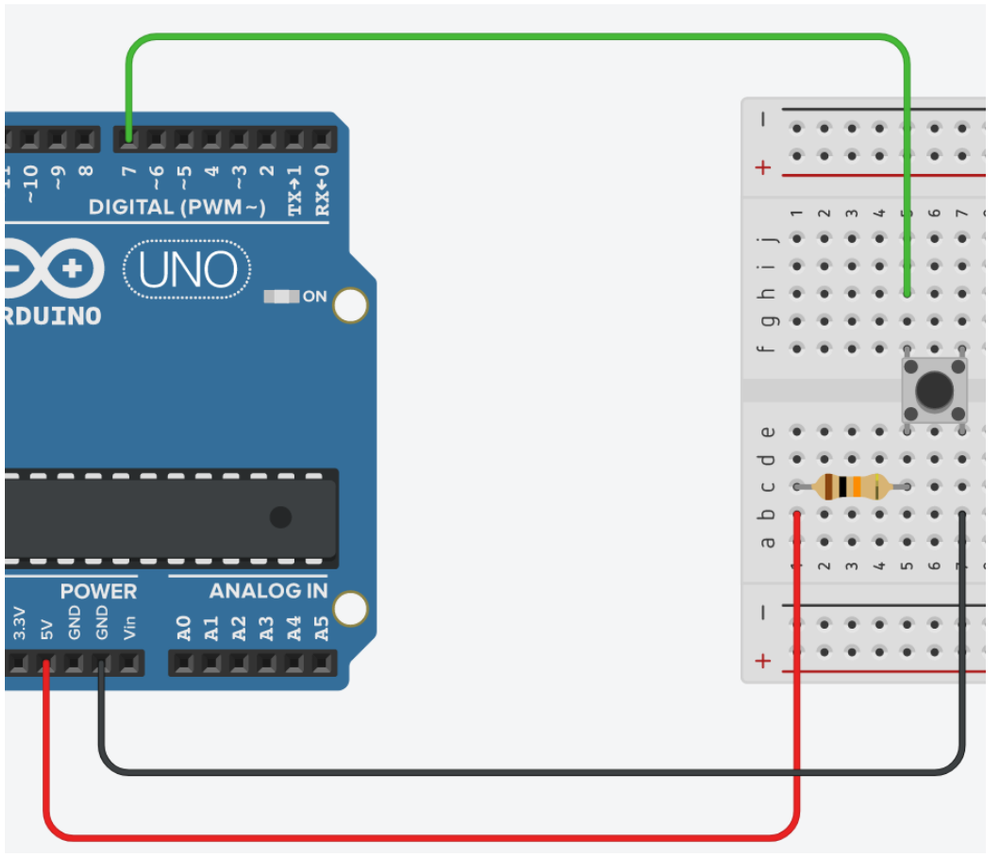
Pull-Down and Pull-Ups

Estado Botão	Pull-Down (valor lido no pino 7)	Pull-UP (valor lido no pino 7)
Premido	HIGH	LOW
Não Premido	LOW	HIGH

Pull Down



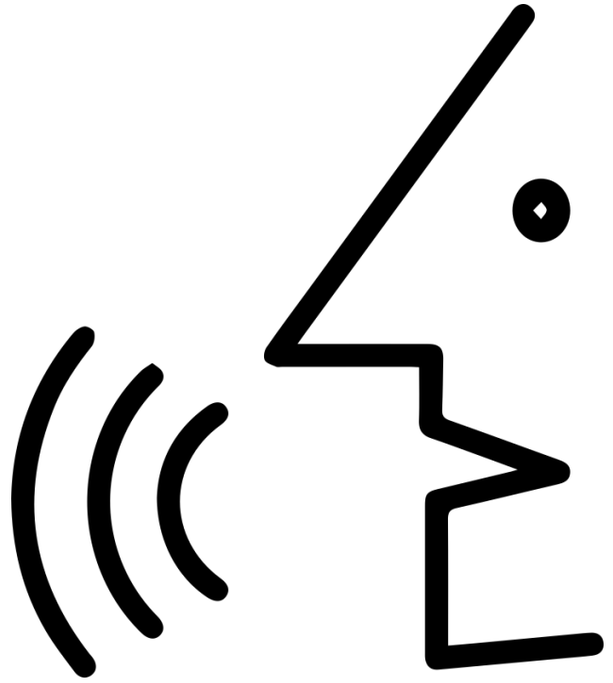
Pull UP



Conceitos: INPUT/Entrada vs. OUTPUT/Saída

Input/Entrada é um sinal / informação que entra no Arduino.

Output/Saída é um sinal / informação que sai do Arduino.

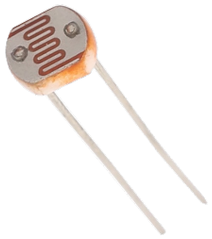


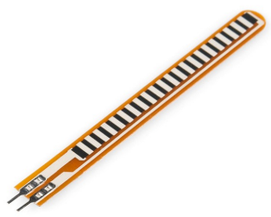
Conceitos: INPUT/Entrada vs. OUTPUT/Saída

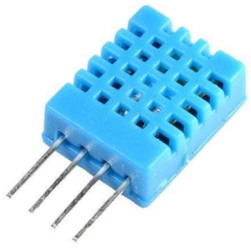
Input/Entrada é um sinal / informação que entra no Arduino.

Exemplos:

Botões, Interruptores, Sensores de Luminosidade, Sensores de Flexão, Sensores de Humidade, Sensores de Temperatura...







Output/Saída é um sinal / informação que sai do Arduino.

Exemplos:

LEDs, motor DC, servo motor, piezo buzzer, relé, LED RGB.

Resolução do exercício 1.5

Faça um programa que conte (e imprima no Serial Monitor) o número de execuções do bloco *loop* decorridas durante a pressão de um botão conectado no pino 7, utilizando os pull-ups internos.

Exercícios TPC:
(da ficha de exercícios)

Exercício 1.10 – Tendo presente o exercício 1.5, altere o programa de forma a ligar um LED, sempre que o número de execuções for múltiplo de 3. O programa deve informar também o número de vezes que acendeu o LED. Caso não seja múltiplo de 3, deve desligar o LED.

Exercício 1.6 - Assuma que montou um circuito com dois LED (um ligado no pino 13 e outro no 12) e um botão de pressão (ligado ao pino 7). Faça um programa para alternar os dois LEDs conforme o estado do botão.

Para os/as mais corajosos/as

Para os/as mais corajosos/as

Exercício 1.7 - *Debouncing*

a) - Analise o exemplo *debounce* da categoria *Digital*, incluído com o Arduino IDE e tente compreender em que é que consiste o fenómeno de *bouncing*. Averigue como funciona a função *millis()*.

b) - Partindo de um circuito idêntico ao do exercício 1.6, mas agora reduzido a apenas um LED, escreva um programa que acenda e apague um LED sempre que o botão seja premido 3 vezes seguidas.