Introdução ao Arduino

Prof. Simão

simao@ufpr.br - 2025

Dúvidas

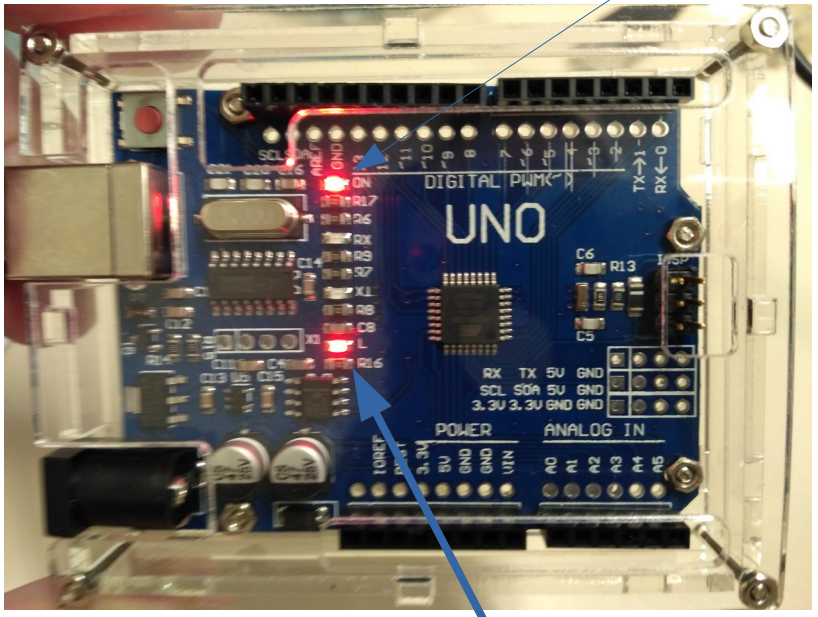
Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

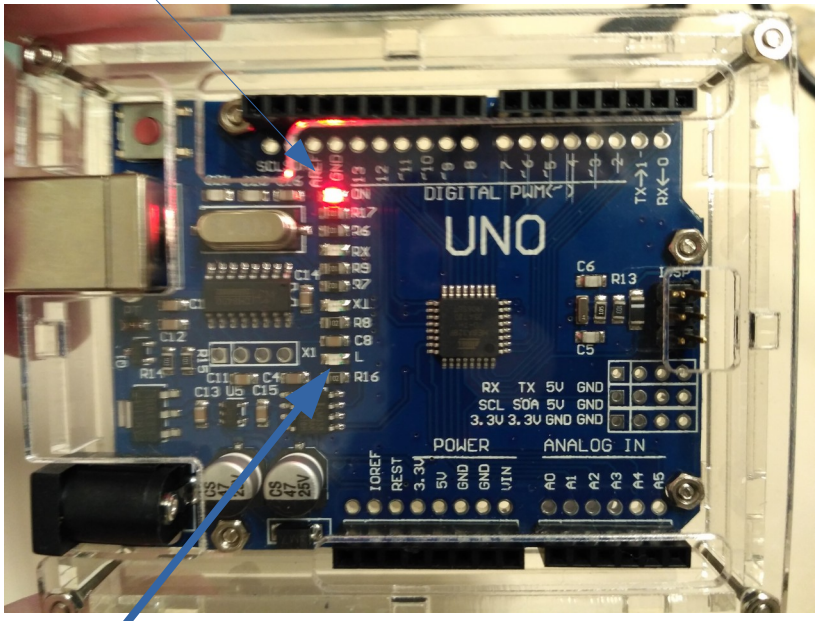
Caso contrário, vamos em frente...

simao@ufpr.br - 2025

Já temos um led piscando...

LED de energia – sempre aceso





LED da placa (pino 13) deverá piscar

Imagens: fotos do autor

simao@ufpr.br - 2025

Vamos fazer isto fora da placa

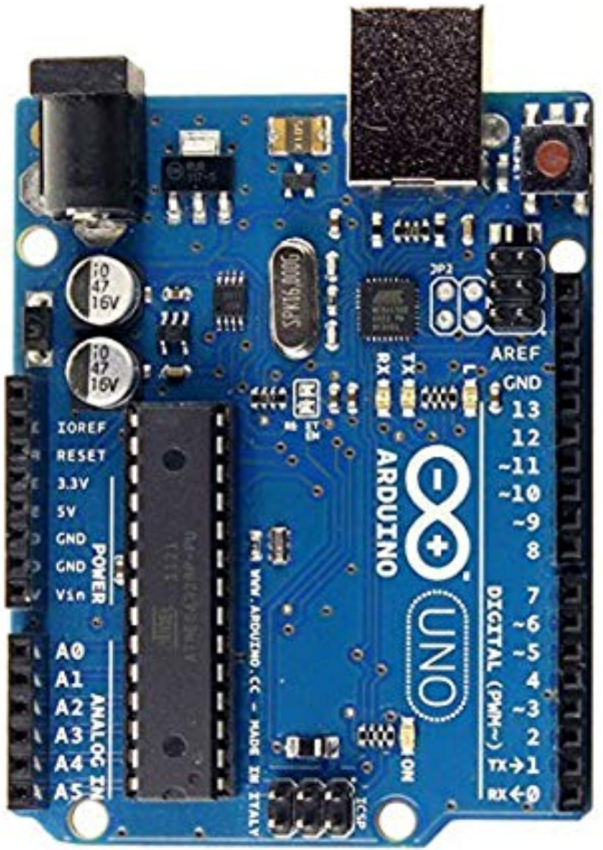
- A placa possui pinos configuráveis como entradas ou saídas
  - Digitais: binário / ligado e desligado
    - 0 ou 1 LOW ou HIGH 0v ou 5v
  - Analógicas: uma gama de valores entre mínimo e máximo

simao@ufpr.br - 2025

Posição

Pinos de energia *POWER*

Entradas Analógicas



Entradas / Saídas Digitais

Imagem: A posteriori

simao@ufpr.br - 2025

Ligar led fora da placa

- Vamos usar e reusar o código já testado, mudando o número da porta que usaremos como saída

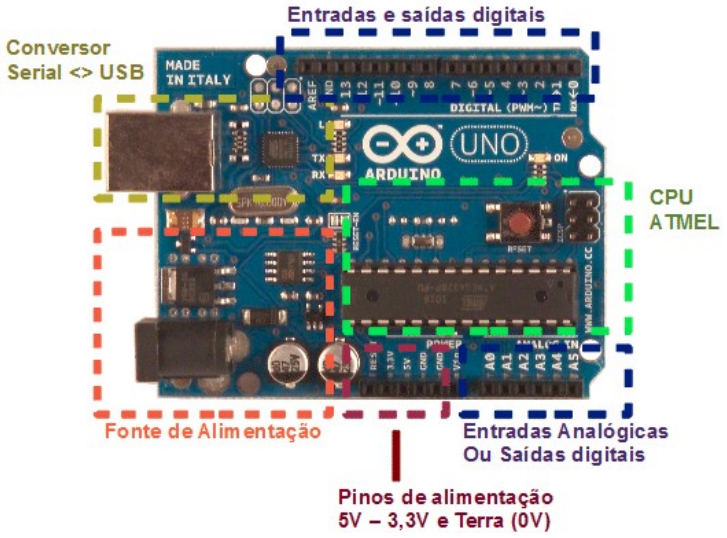


Imagem: https://www.researchgate.net/figure/Figura-1-Placa-de-prototipagem-eletronica-Arduino-UNO-Fonte-wwwrobotizandocombr_fig1_307593728

Breadboard / Protoboard

- Placa para montagens experimentais

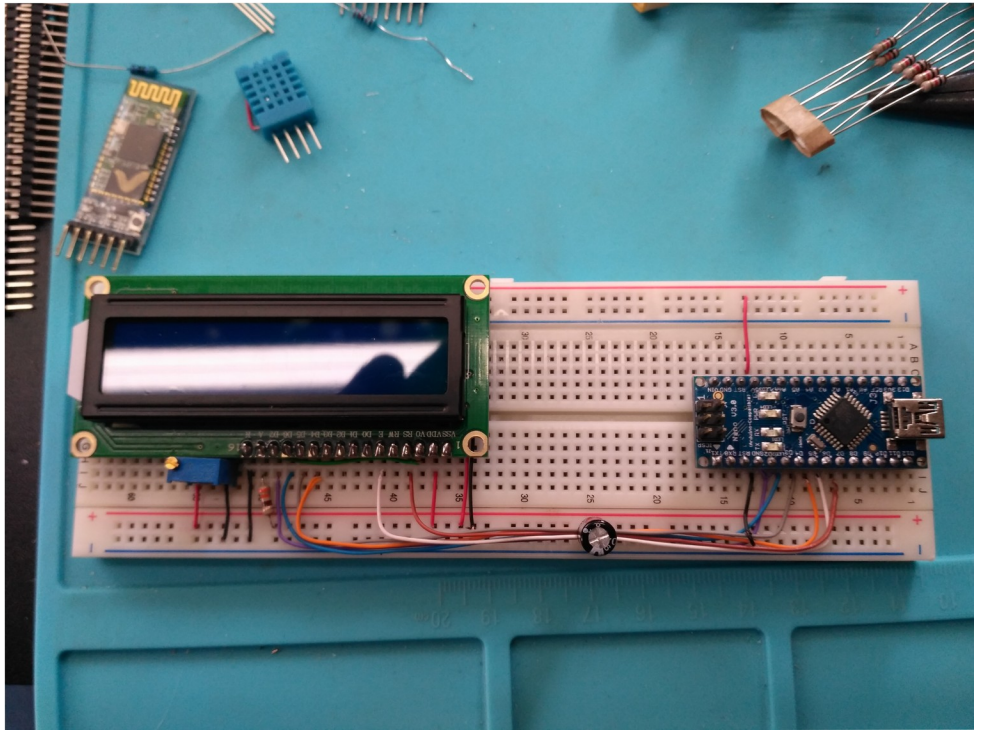


Imagem: foto do autor

simao@ufpr.br - 2025

Breadboard / Protoboard

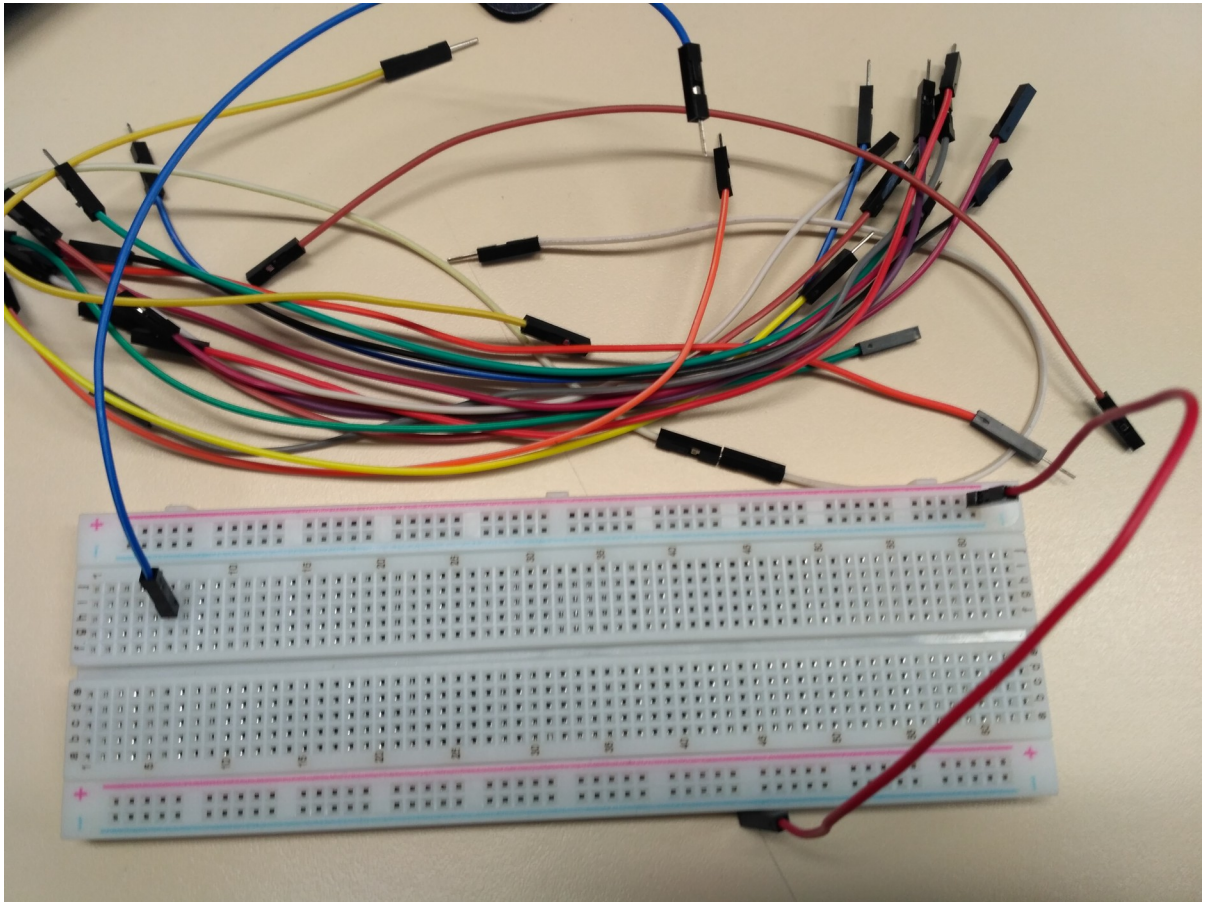


Imagem: foto do autor

simao@ufpr.br - 2025

Breadboard / Protoboard

Nas barras horizontais todos os pinos são ligados juntos (mas o '+' é independente do '-'). Nas verticais, os pinos são ligados juntos, mas a barra de cima não é ligada na de baixo, e cada coluna é independente.

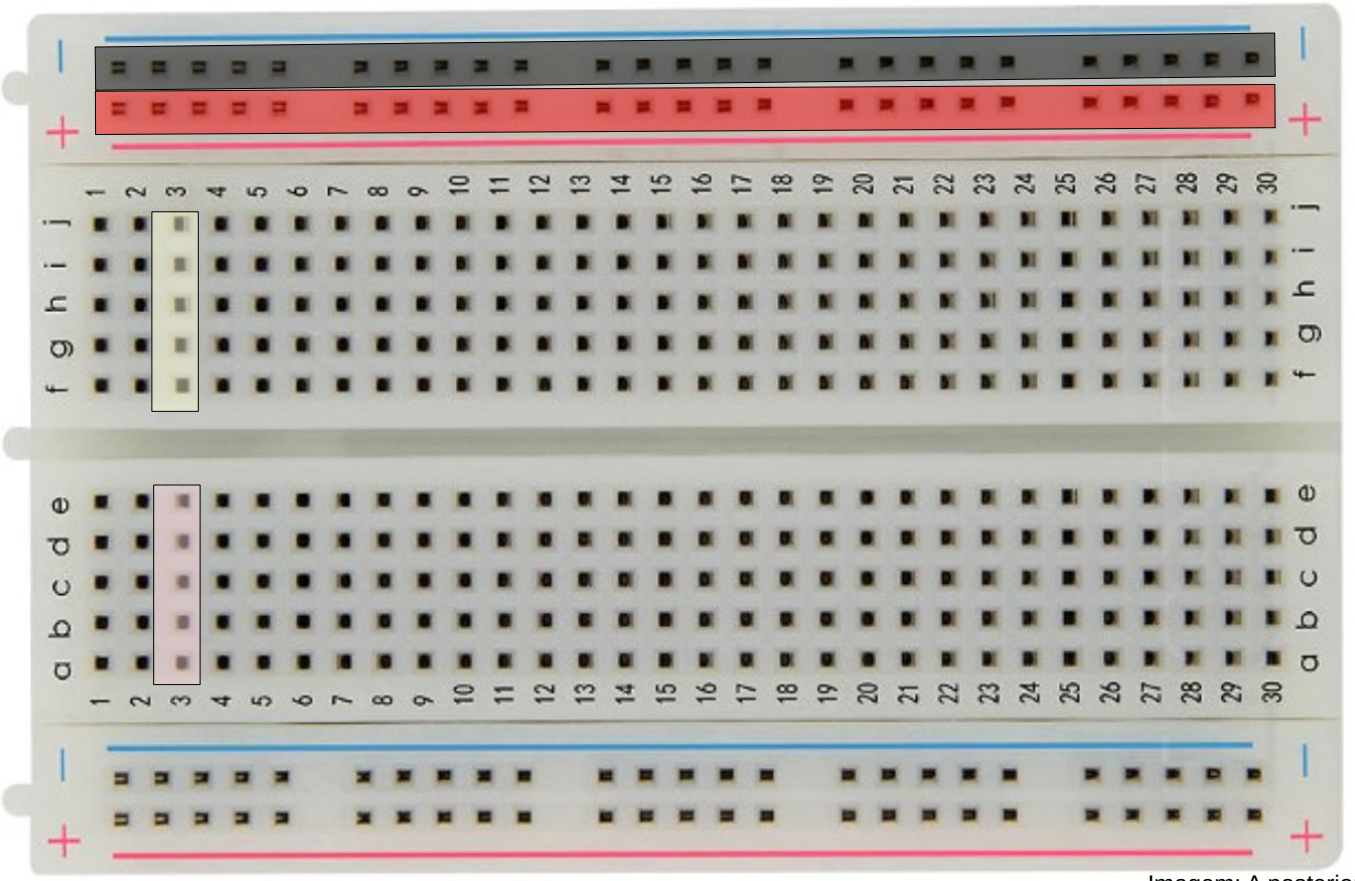
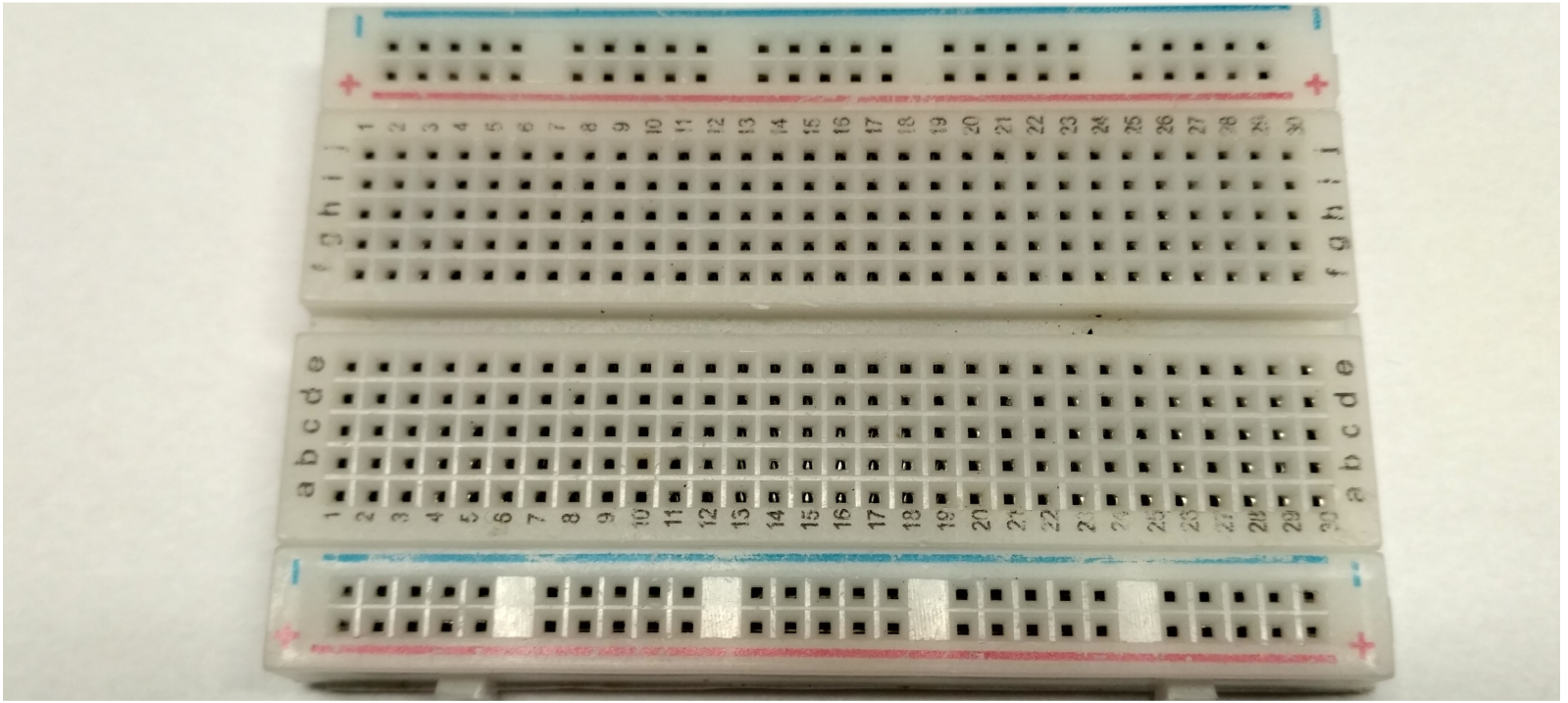


Imagem: A posteriori

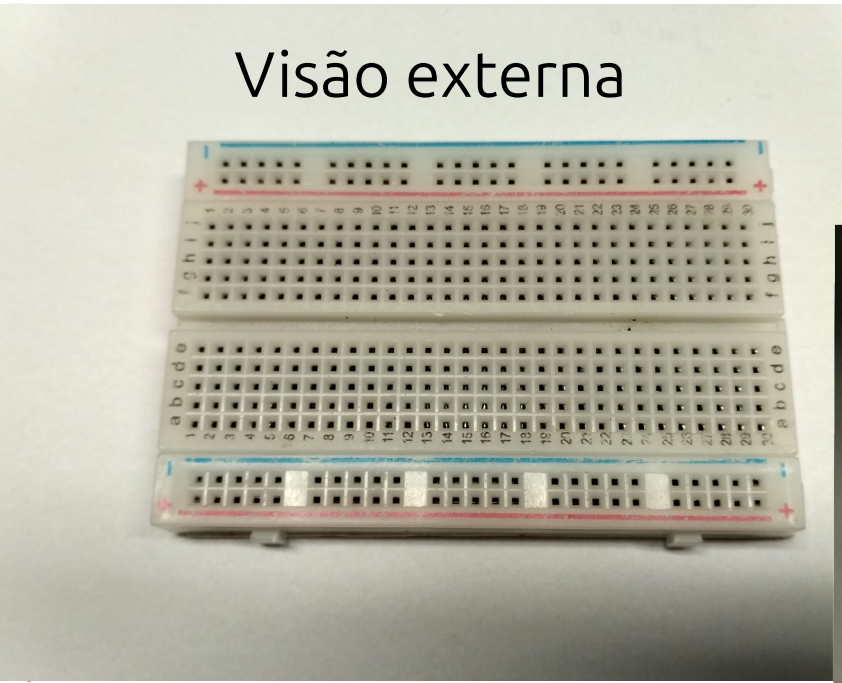
Organização matricial, identificável por linhas alfabéticas e colunas numéricas para facilitar a disposição de componentes.



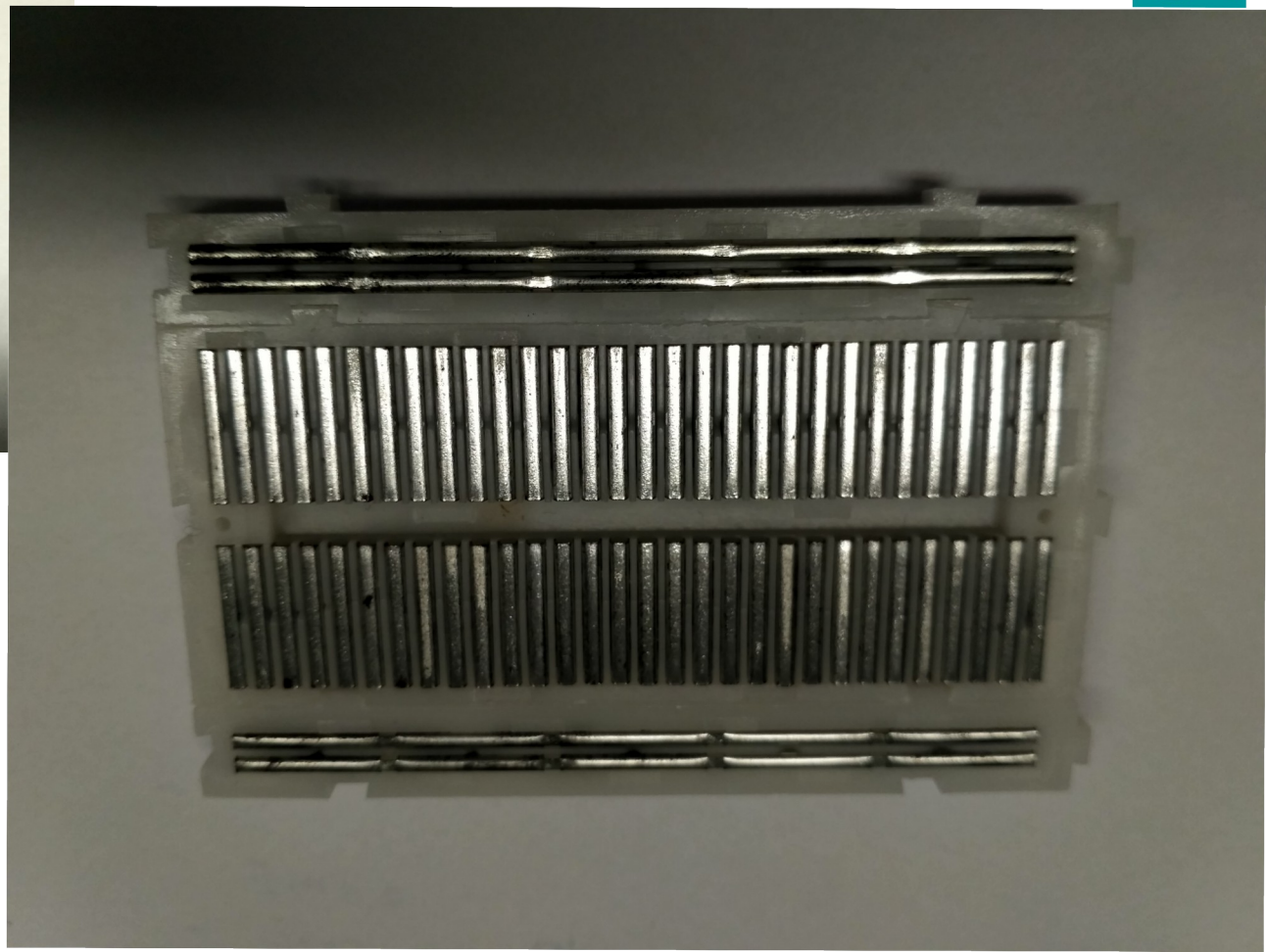
As barras horizontais identicando terminais **negativo (-)** e **positivo (+)** são sugestões. Não são polarizadas, servem para organização

simao@ufpr.br - 2025

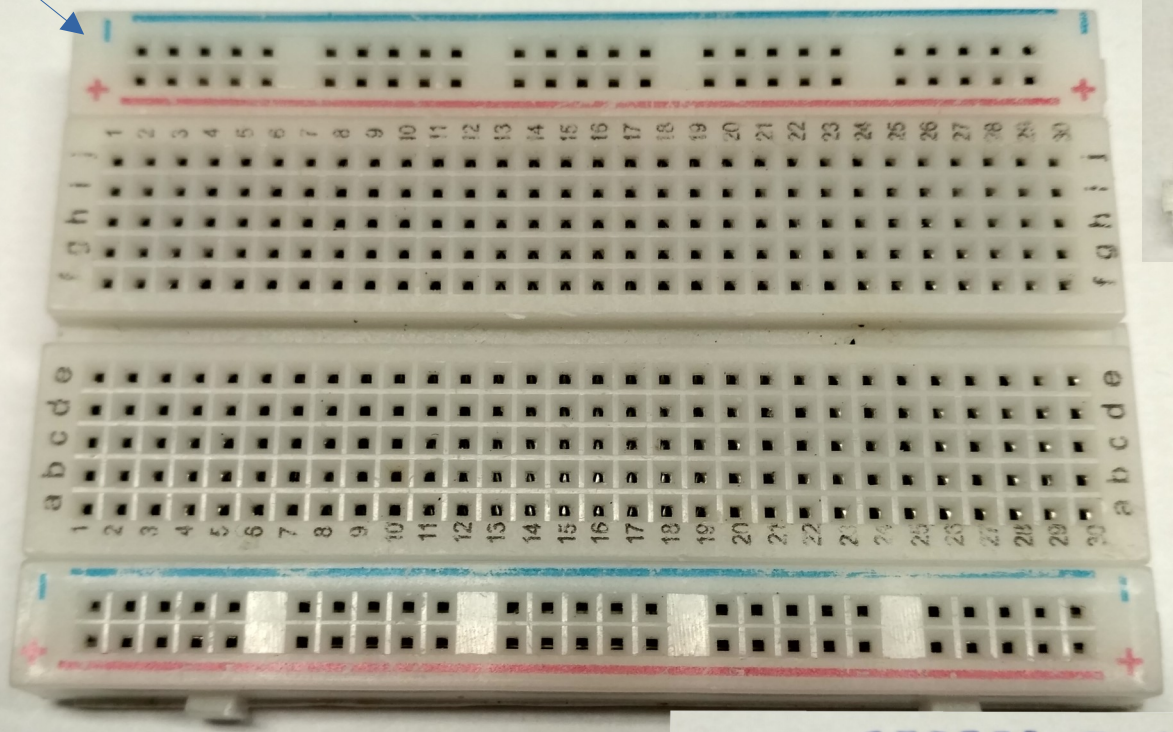
Visão externa

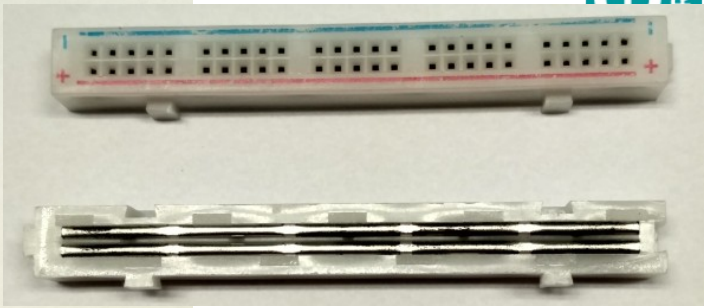


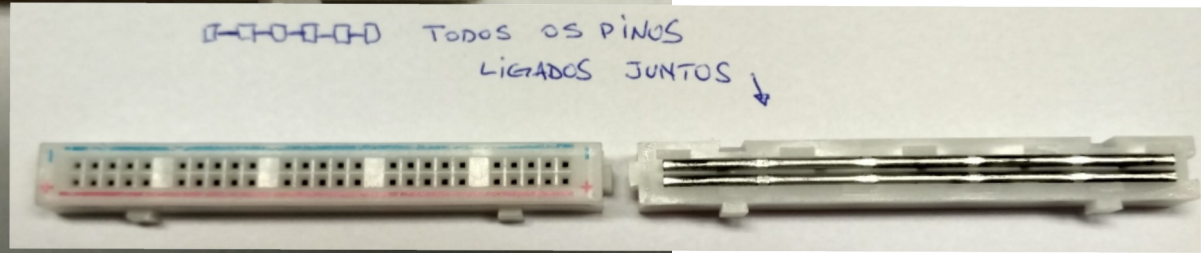
Visão interna – conexões elétricas



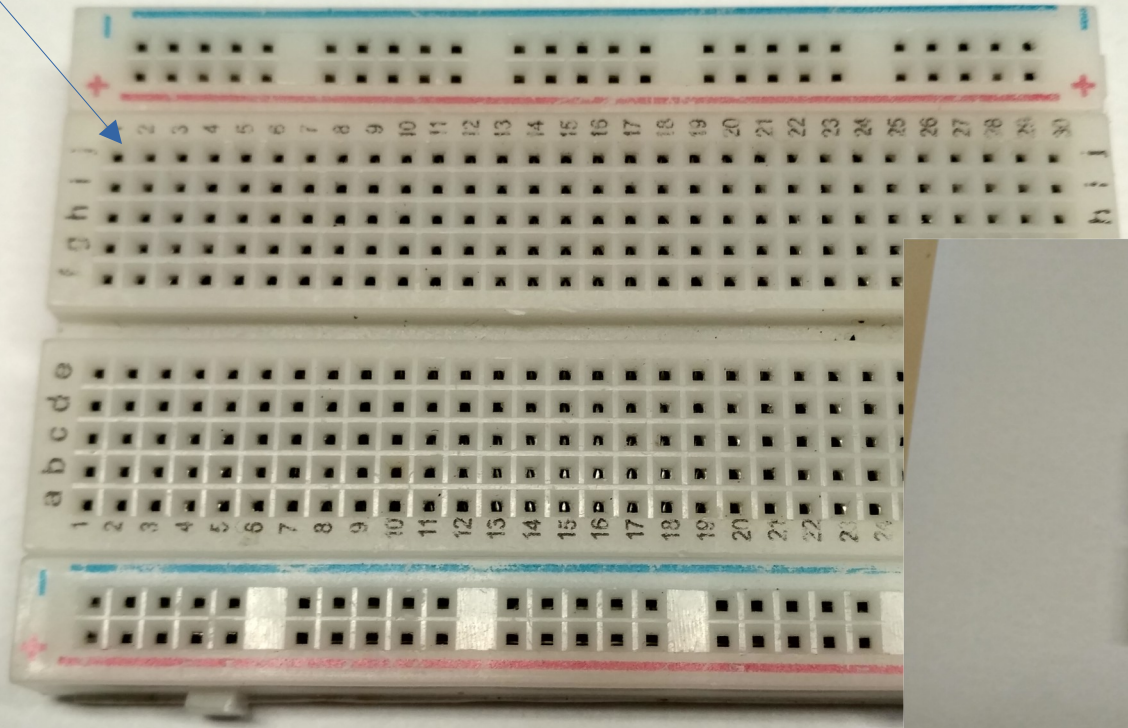
Barras de conexão horizontal de distribuição de energia: todos os pinos da mesma barra estão interconectados

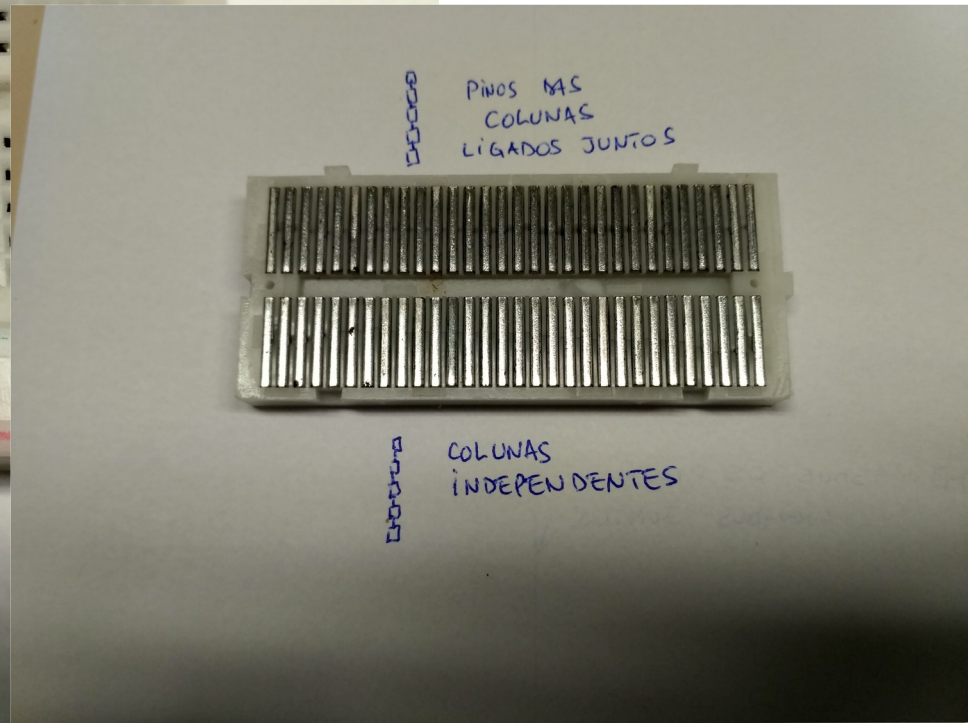






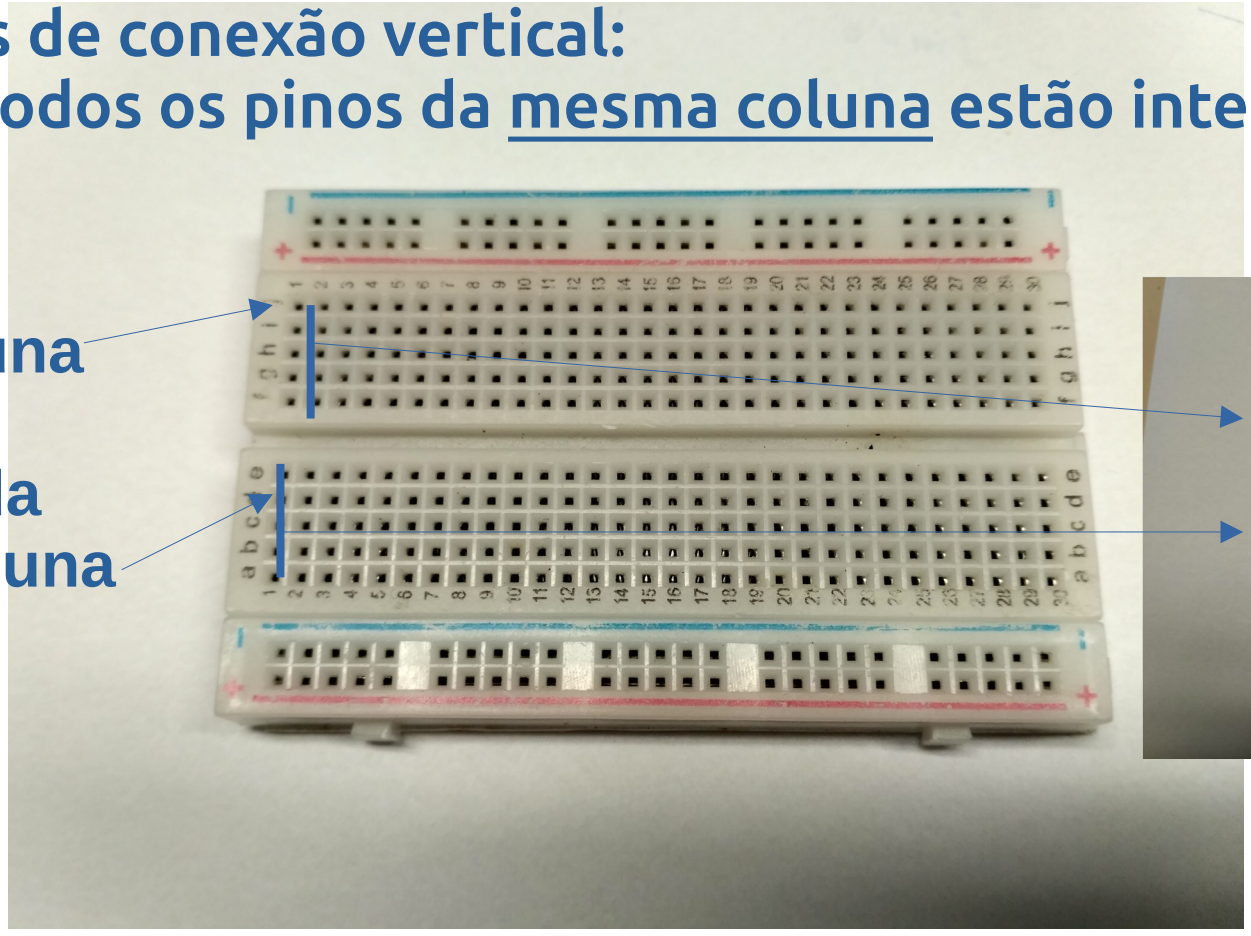
Colunas de conexão vertical:
todos os pinos da mesma coluna estão interconectados

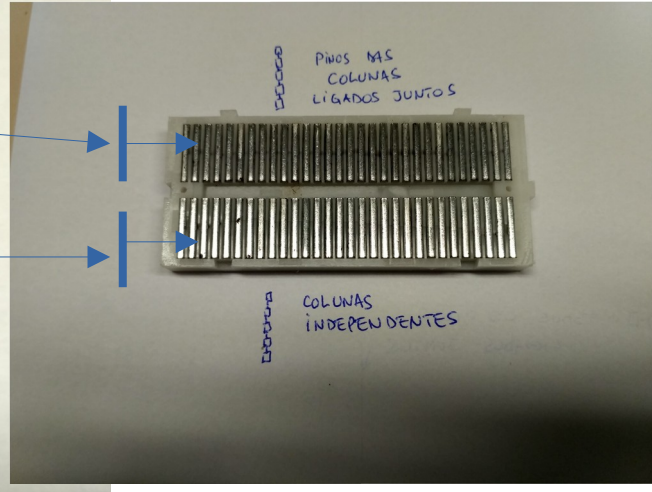




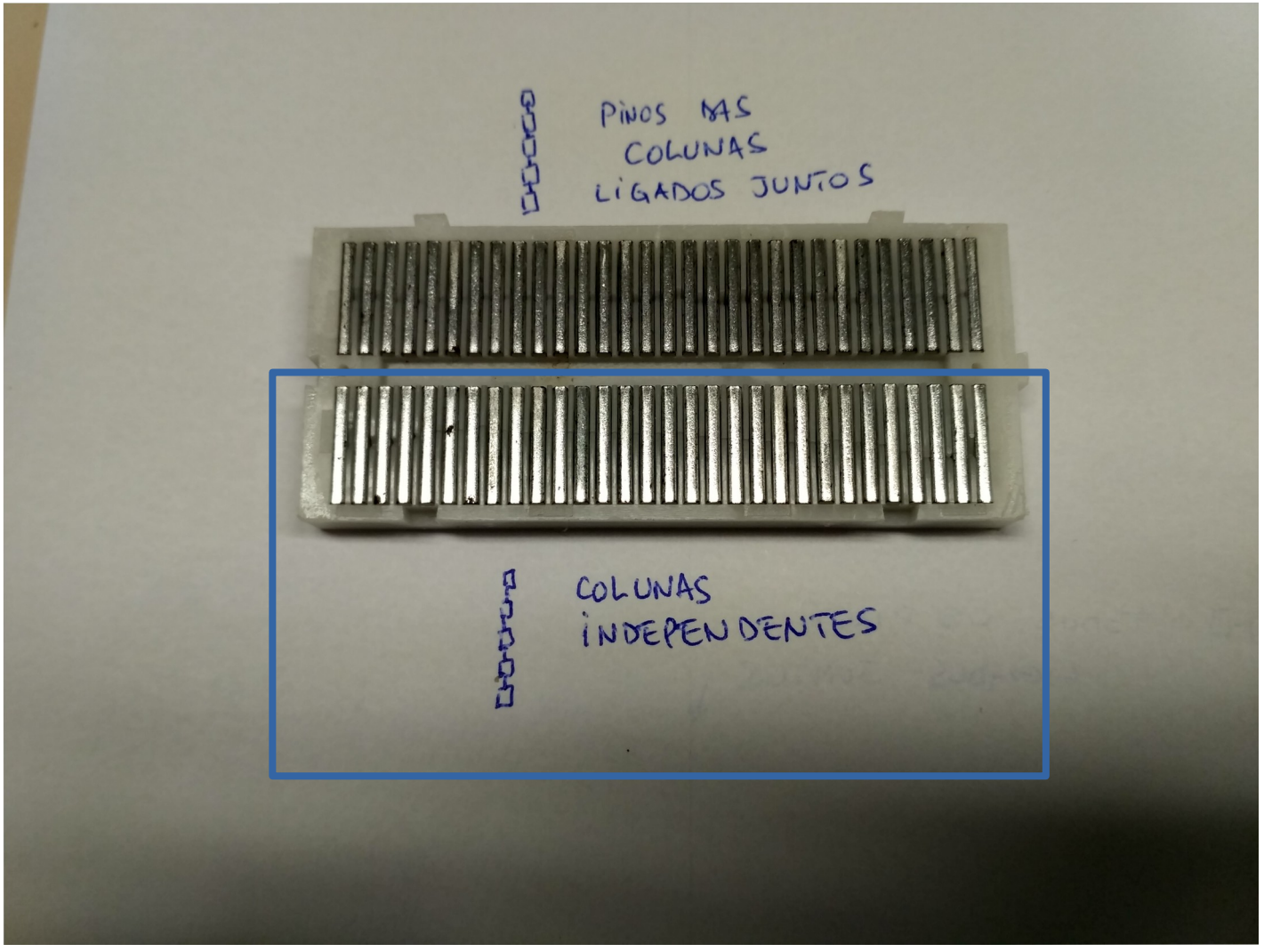
Colunas de conexão vertical:
todos os pinos da mesma coluna estão interconectados

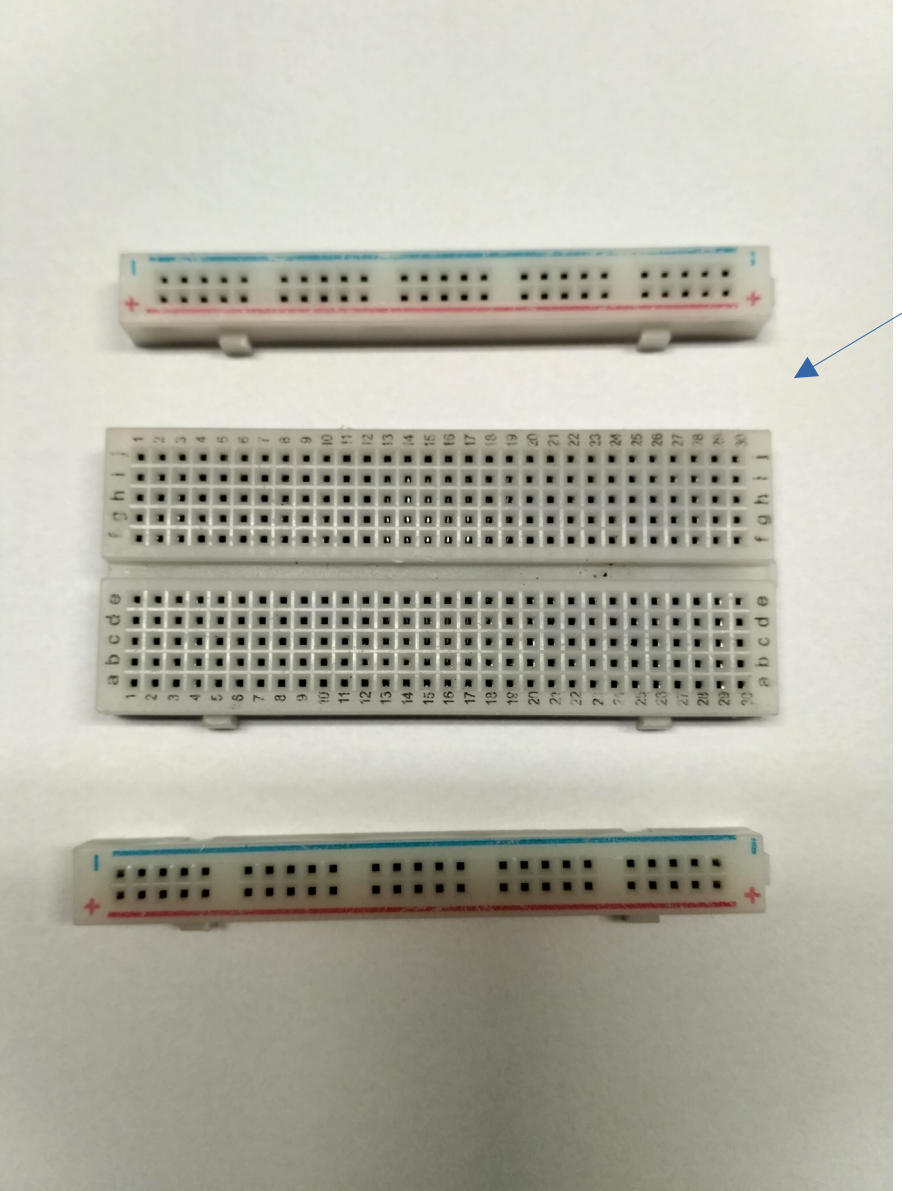
Esta coluna
não está
conectada
nesta coluna



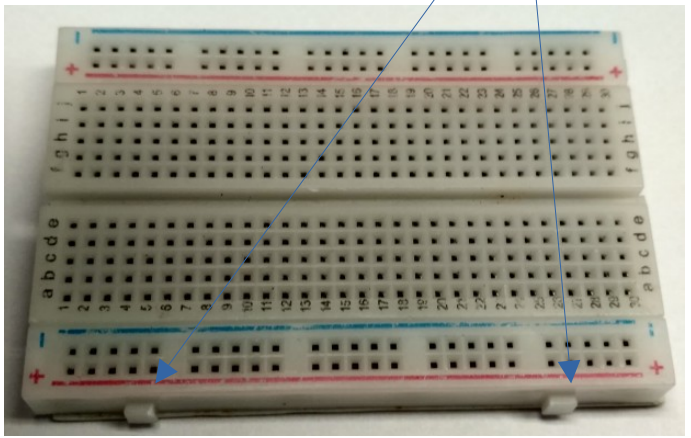


simao@ufpr.br - 2025

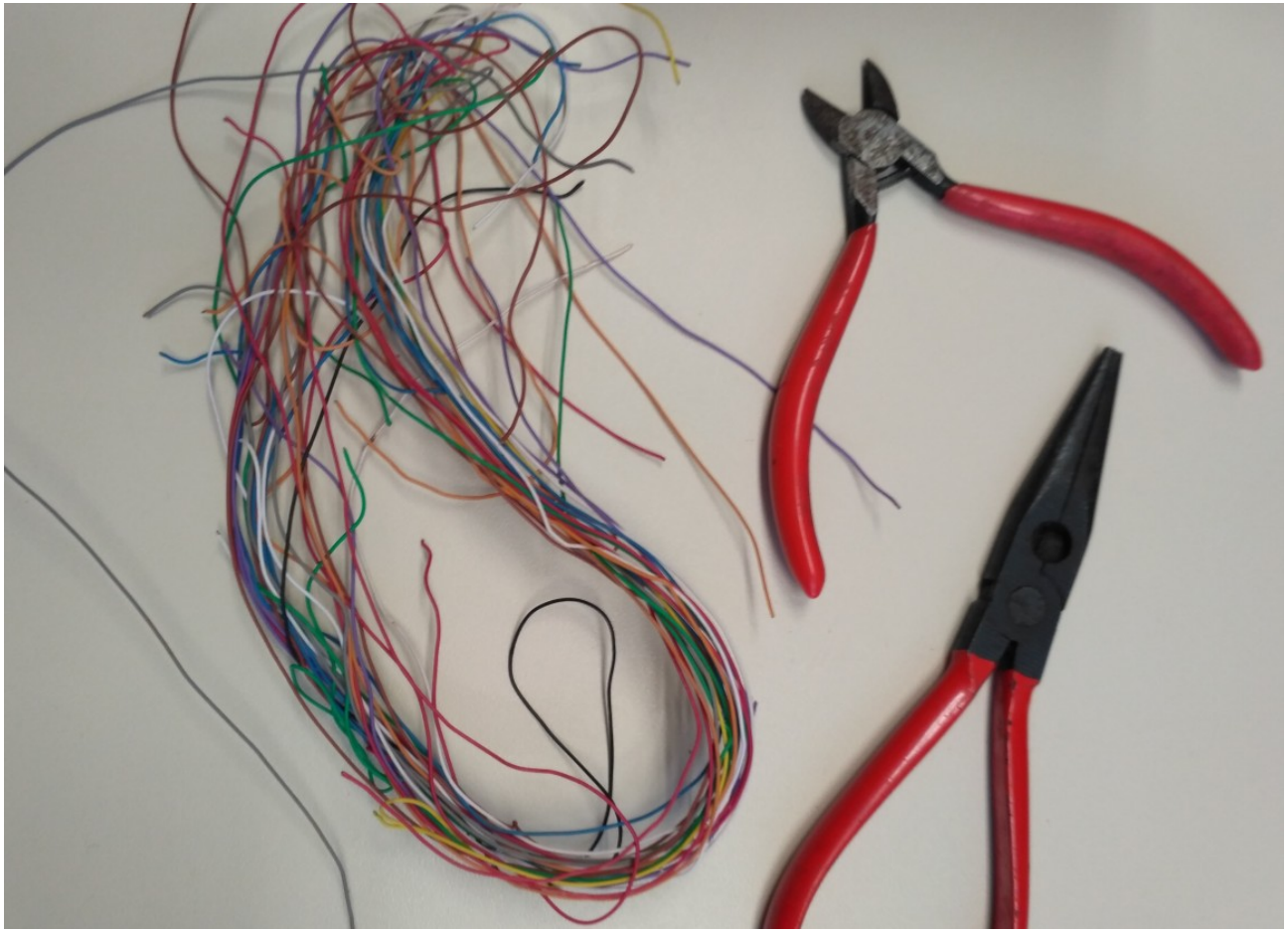




Algumas placas podem ser desmontadas quanto às barras e colunas e todas podem ser associadas para criar placas maiores por meio de encaixes



Ferramentas e fios



simao@ufpr.br - 2025

Placas padronizadas - soldagem

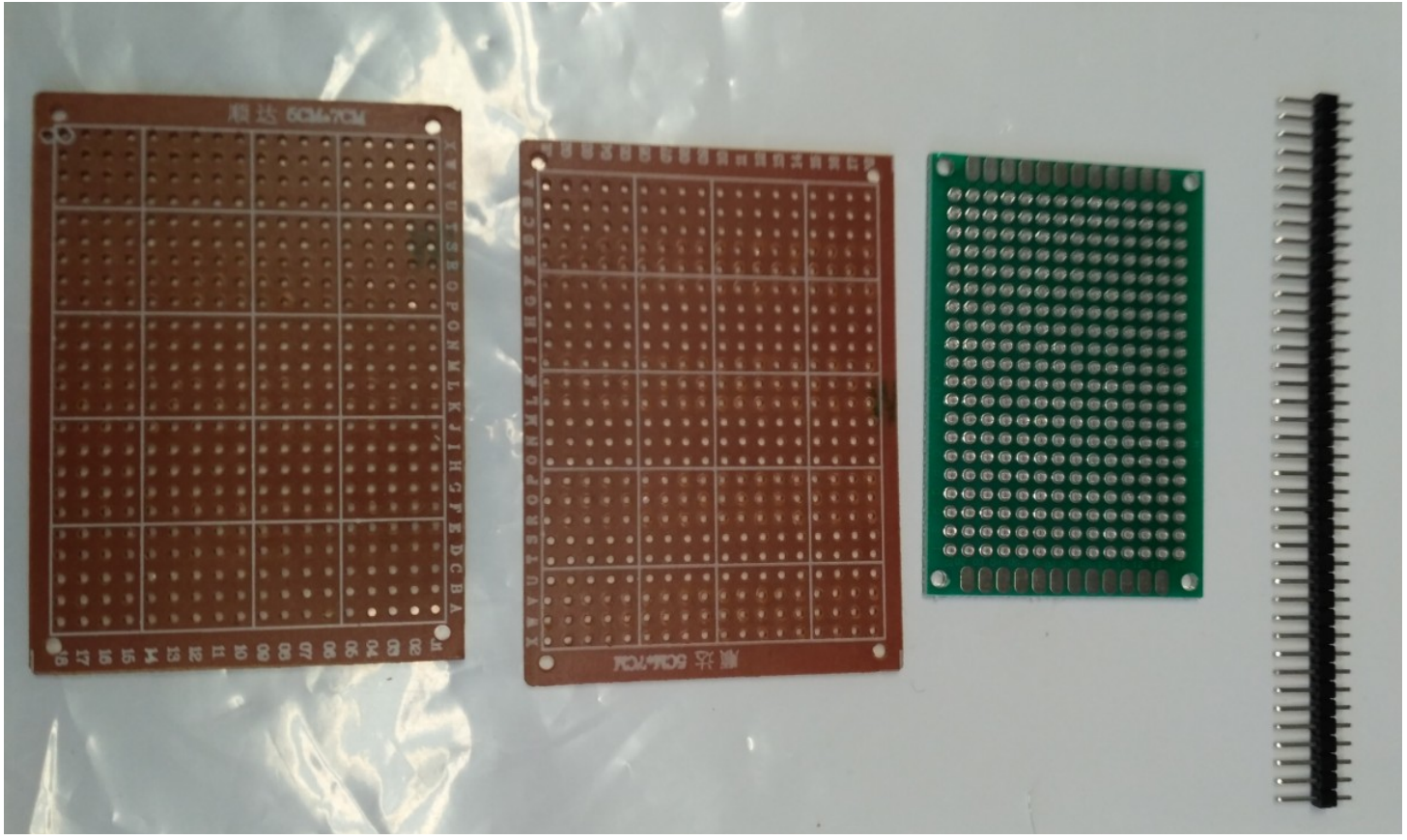


Imagem: foto do autor

simao@ufpr.br - 2025

Vamos trabalhar

simao@ufpr.br - 2025

O que precisamos?

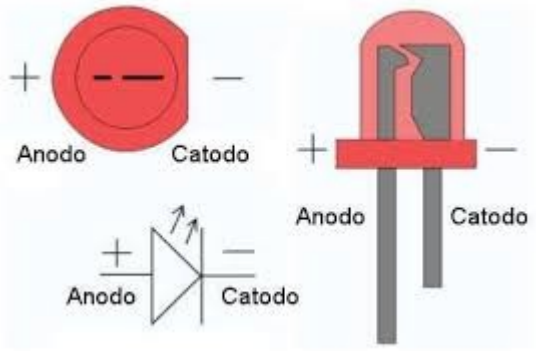
- Arduino e IDE configurados
- *Breadboard* e fios de conexão
- Um LED, um resistor
  - O resistor é necessário em função de que o LED trabalha com características elétricas diferentes do Arduino
    - Na sequência, veremos como determinar o valor do resistor

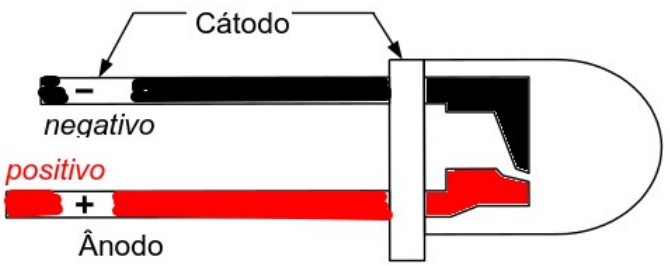
simao@ufpr.br - 2025

Início da eletrônica teórica

simao@ufpr.br - 2025

Diodo emissor de luz - LED





É um componente que possui **polaridade** definida, um lado vai ligado à tensão positiva e outro à negativa da fonte.

O Arduino trabalha com 5V; algumas placas trabalham com 3,3V. Os leds vermelhos trabalham com 1,7V, o amarelo com 2V e o verde com 2,1V. Azuis e brancos beiram 3V e infravermelhos 1,1V.

A corrente dos leds dificilmente ultrapassa 20mA.

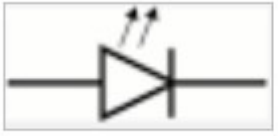
O símbolo do LED é  (ou  ou )

Imagem: Google

simao@ufpr.br - 2025

Polaridade

- Se tiver dúvidas quanto à polaridade do LED, use um multímetro ou uma pequena bateria (CR2032) para testar.
  - Regra geral – **vermelho é positivo** e **preto é negativo**

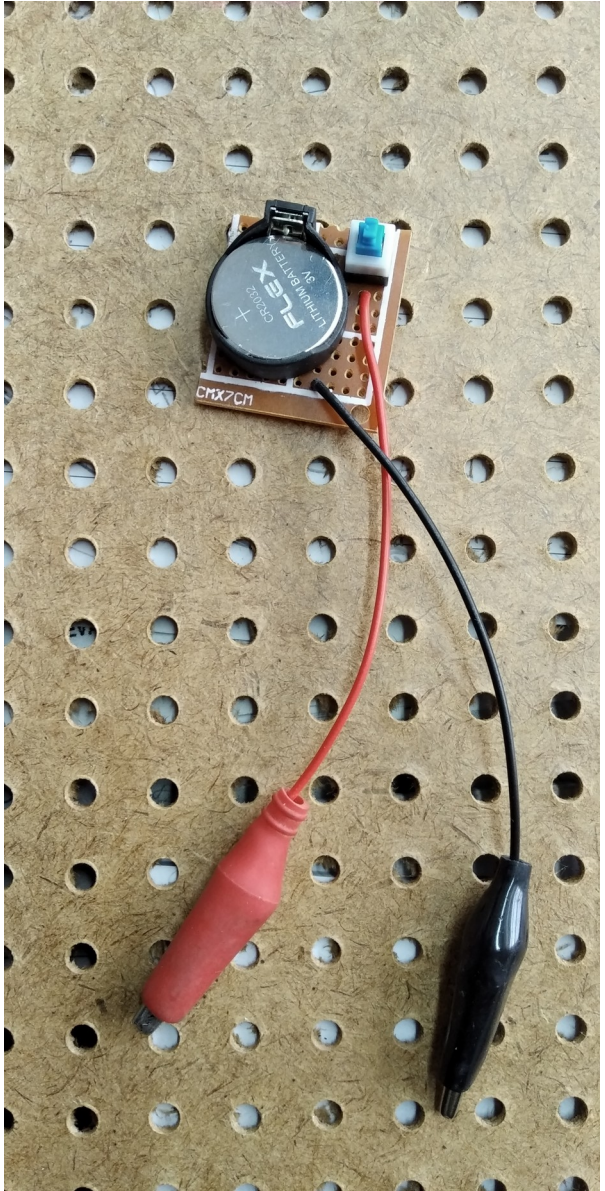


Imagem: do autor

simao@ufpr.br - 2025

Polaridade

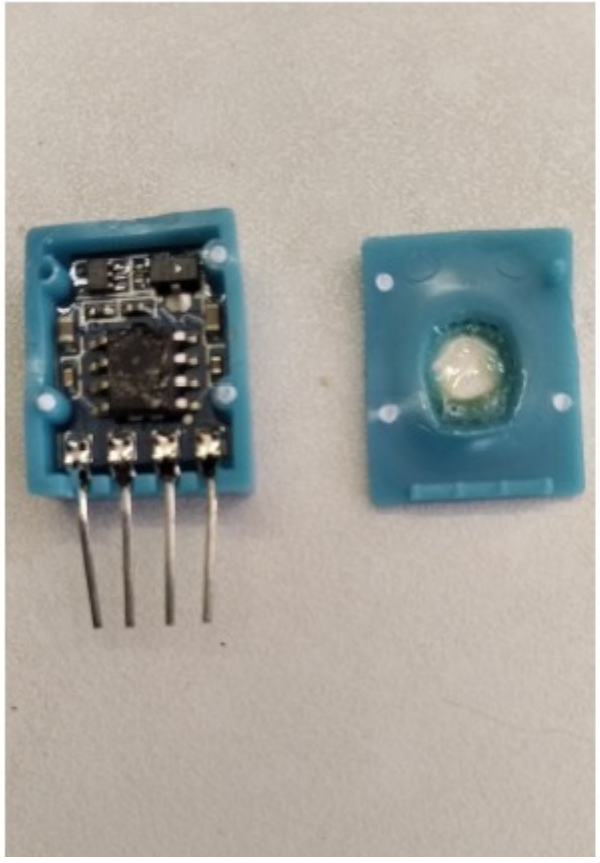
- Regra geral:
  - Positivo (**+**) é **Vermelho** e vai ligado no **5V**
  - Negativo (**-**) é **Preto** e vai ligado no **GND** (ou no **0V**)

(isto facilita a identificação, mas, obviamente, a cor do isolamento do condutor não altera o circuito elétrico; você pode utilizar os fios das cores que estiverem disponíveis – mas, preste atenção na polaridade, ligue positivo com positivo e negativo com negativo).

Há problemas em inverter?

- A inversão da polaridade em um componente polarizado, tal como o led, um transistor e alguns tipos de capacitores, poderá queimá-lo.
- Também poderá haver dano na placa do Arduino.

Há problemas em inverter?



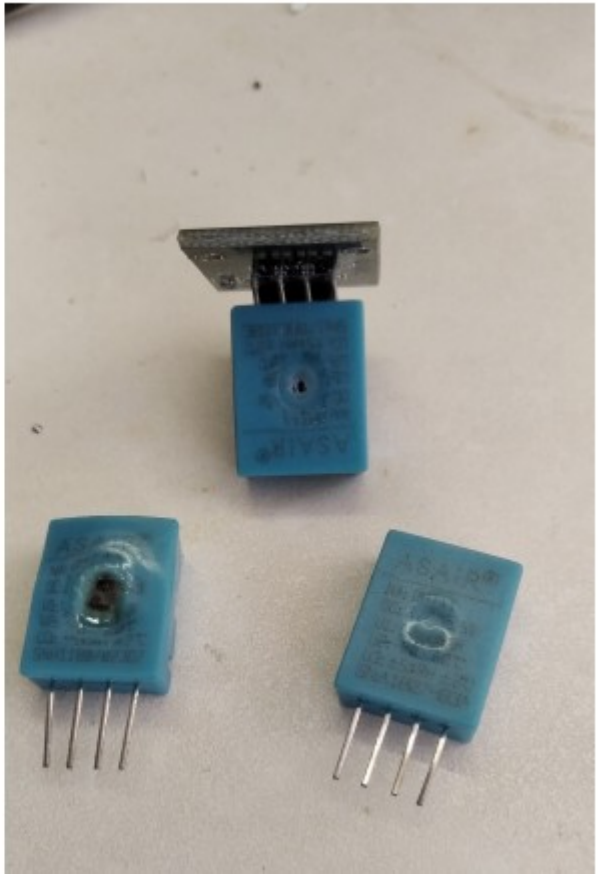


Imagem: do autor

simao@ufpr.br - 2025

https://tiaplicada.ufpr.br/wp-content/uploads/2023/06/destruirarduino.pdf

TI Aplicada https://tiaplicada.ufpr.br

Como destruir um Arduino?

Traduzido de: *10 Ways to Destroy An Arduino*

A ideia não é destruir, e sim saber o que fazer para **não** destruir...

Nem sempre tem fusível

ATENÇÃO

A placa do Arduino UNO suporta no máximo 20mA (0,02 A) de corrente em cada saída, e no máximo 200mA (0,2 A) no total (e com limitações a grupos de portas...).

Procure usar sempre menos do isto!

simao@ufpr.br - 2025

Como limitar a corrente/ tensão

- De forma a adequar as grandezas elétricas disponíveis na placa do Arduino às características do led, usamos resistores.

Resistores

- Como diz o nome, são componentes que resistem à passagem da corrente elétrica.
- Em função do esforço realizado para a passagem por um resistor será gerada uma queda na tensão disponível.

simao@ufpr.br - 2025

Lei de Ohm

- Estudada na cadeira de física do ensino médio, a Lei de Ohm estabelece uma relação entre tensão, corrente e resistência.
- A queda de tensão é dada pelo produto da corrente pela resistência:
  - E = R.I

Valor do resistor

- Vamos estabelecer uma baixa corrente para nosso resistor, 10mA (0,01A), e vamos calcular para que produza uma queda de 3V na tensão:
  - E = R.I
  - 3V = R.0,01
  - R = 300 Ω (300 ohms)

simao@ufpr.br - 2025

Séries comerciais de resistores

	A	B	C	D
1	**Valor**	**E24**	**E12**	**E6**
2	1	5,00%	10,00%	20,00%
3	1,1	5,00%		
4	1,2	5,00%	10,00%	
5	1,3	5,00%		
6	1,5	5,00%	10,00%	20,00%
7	1,6	5,00%		
8	1,8	5,00%	10,00%	
9	2	5,00%		
10	2,2	5,00%	10,00%	20,00%
11	2,4	5,00%		
12	2,7	5,00%	10,00%	
13	3	5,00%		
14	3,3	5,00%	10,00%	20,00%
15	3,6	5,00%		
16	3,9	5,00%	10,00%	
17	4,3	5,00%		
18	4,7	5,00%	10,00%	20,00%
19	5,1	5,00%		
20	5,6	5,00%	10,00%	
21	6,2	5,00%		
22	6,8	5,00%	10,00%	20,00%
23	7,5	5,00%		
24	8,2	5,00%	10,00%	
25	9,1	5,00%		

Existem várias séries de valores padronizados para os resistores.

Nos circuitos, escolhe-se o componente comercial de valor mais próximo do desejado.

Os resistores são fabricados/ vendidos baseando-se em um valor central, ao longo do qual temos variações para mais ou para menos (tolerância).

Os valores indicados ao lado terão multiplicadores dados por potências de dez (ex.: x 10, x10.000, etc).

Imagem: cópia de tela de planilha LibreOffice efetuada pelo autor

simao@ufpr.br - 2025

E o nosso resistor de 300Ω ?

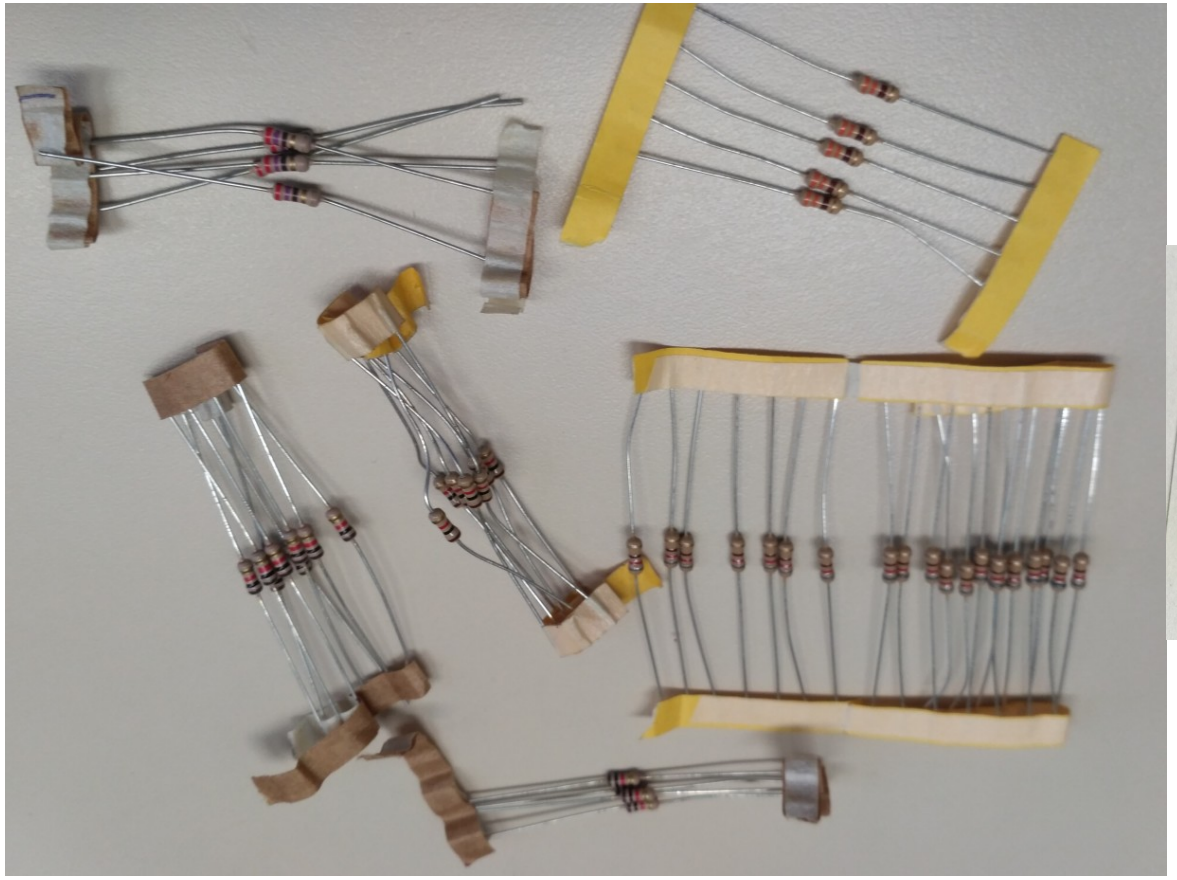
- Os valores mais próximos seriam 2,7; 3 e 3,3; todos multiplicados por 100.
- A escolha óbvia seria 3 x 100 = 300, porém nem sempre temos à mão o valor desejado.

12	2,7	5,00%	10,00%	
13	3	5,00%		
14	3,3	5,00%	10,00%	20,00%

Imagem: cópia de tela de planilha LibreOffice efetuada pelo autor

simao@ufpr.br - 2025

Aspecto de resistores



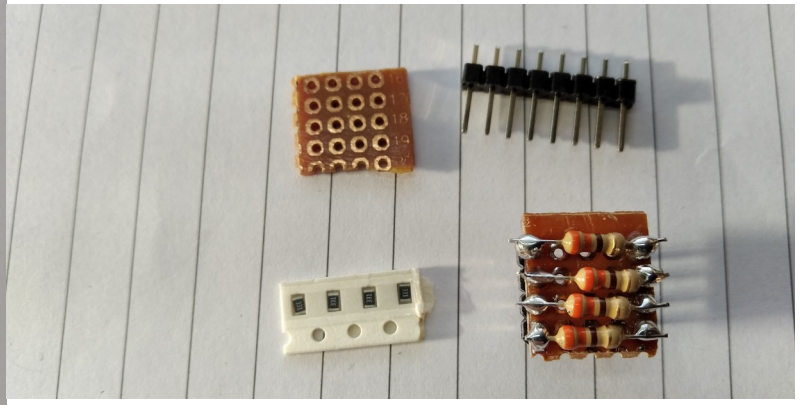


Imagem: foto do autor

simao@ufpr.br - 2025

Código de cores

Cor	1ª Faixa	2ª Faixa	N° de zeros/multiplicador	Tolerância
Preto	0	0	0	
Marrom	1	1	1	
Vermelho	2	2	2	
Laranja	3	3	3	
Amarelo	4	4	4	
Verde	5	5	5	
Azul	6	6	6	
Violeta	7	7	7	
Cinza	8	8	8	
Branco	9	9	9	
Dourado			x0,1	
Prata			x0,01	
Sem cor				± 20%

Resistores são identificados por códigos de cores, conforme mostrado ao lado.

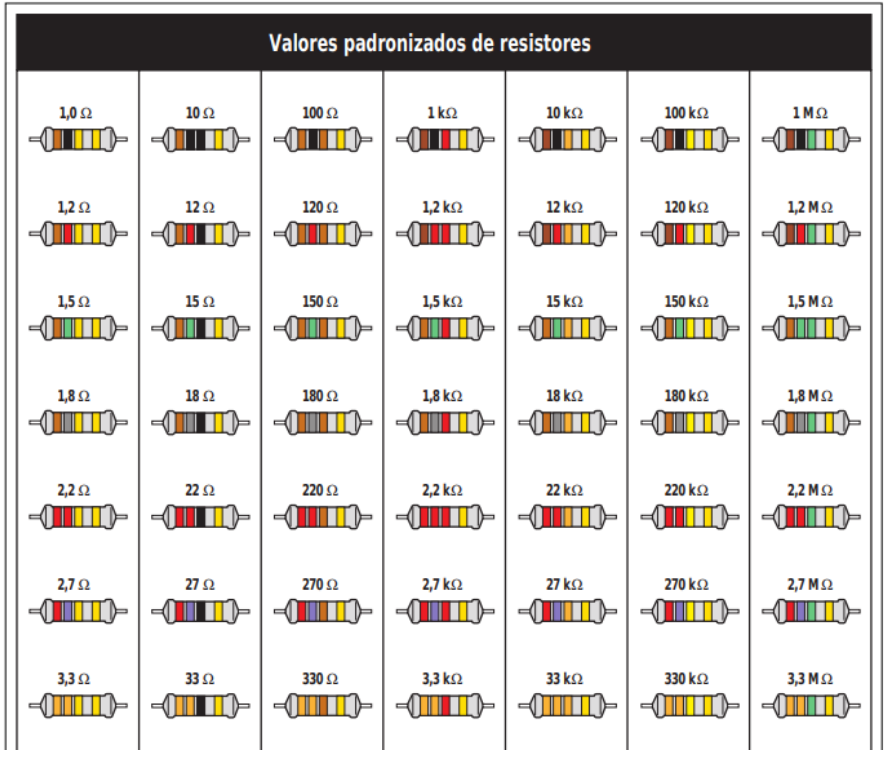
O resistor que aparece abaixo possui (marrom=)1 (preto=) 0 (vermelho=) 00 ohms, ou 1000 ohms, ou 1000 Ω (normalmente identificado como sendo de 1kΩ).

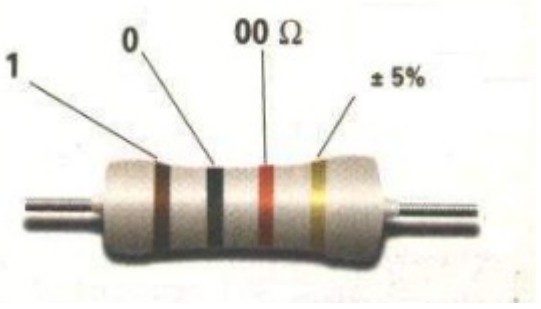
Imagem: https://www.mundodaeletrica.com.br/i/766/3-faixas-600.webp

simao@ufpr.br - 2025

Código de cores

https://tiaplicada.ufpr.br/wp-content/uploads/2025/06/coresresistores.pdf

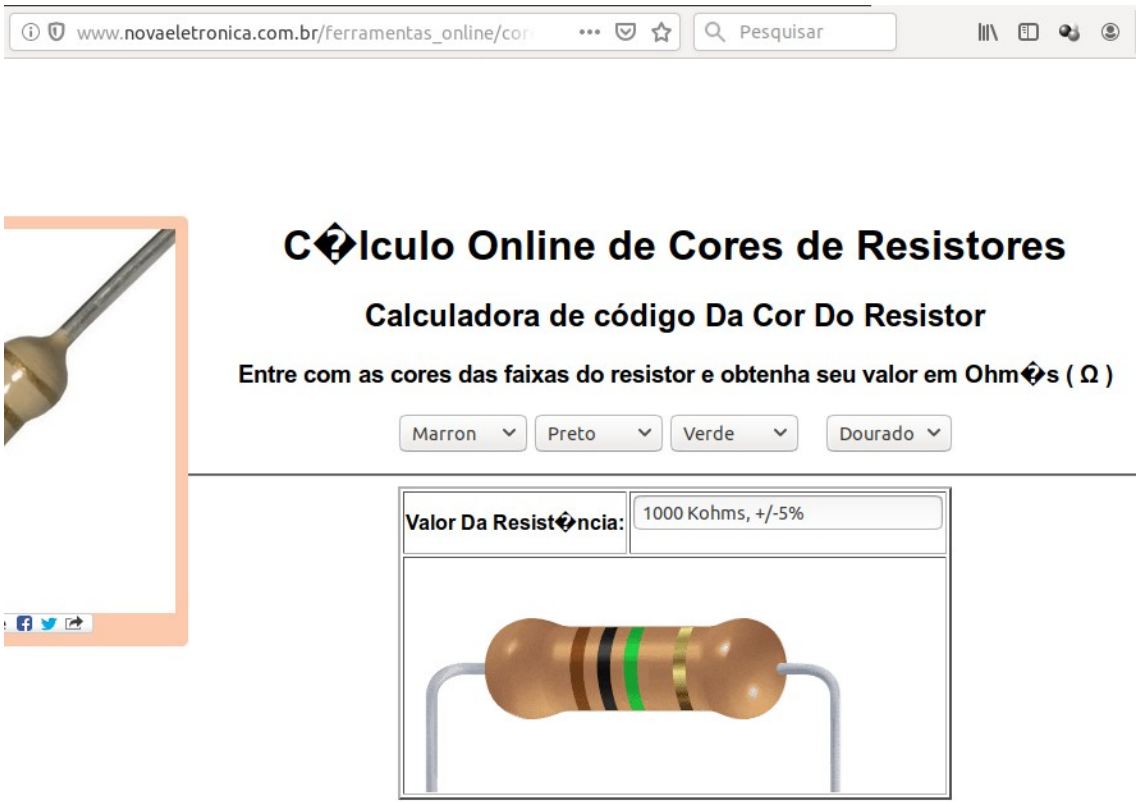




simao@ufpr.br - 2025

Calculadora *online*

http://www.novaeletronica.com.br/ferramentas_online/cores-de-resistor-online.php



Finalmente, nossas opções usuais

Vermelho (2), violeta (7), marrom (1 zero, x10) = 270 Ω

Laranja (3), Preto (0), marrom (1 zero, x10) = 300 Ω

Laranja (3), Laranja (3), marrom (1 zero, x10) = 330 Ω

Imagens: Google

simao@ufpr.br - 2025

Pode usar outro valor?

- Pode, MAS:
  - Se for menor do que 100 ohms a corrente pode aumentar em demasia, queimando a placa
    - Em especial se forem vários leds ligados
  - Se for superior a 1000 ohms (1k), a corrente pode não ser suficiente para que o led acenda
  - Procure utilizar valores entre 150 ohms e 1000 ohms

Fim da eletrônica teórica

simao@ufpr.br - 2025

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

Ligar led externo

- Primeiramente vamos levar a energia para a *breadboard* e testar
- Depois, iremos mudar o código que faz piscar o led da placa do Arduino para que pisque o led externo

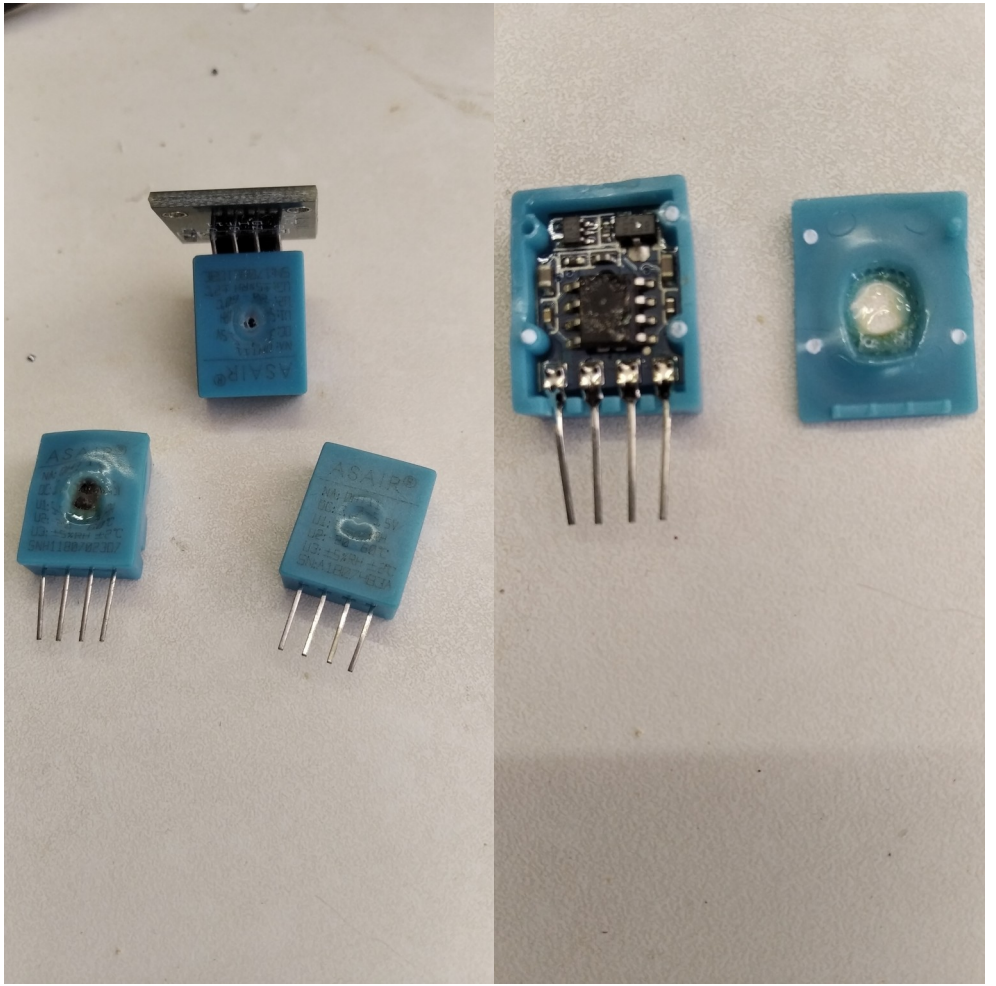
simao@ufpr.br - 2025

Evite problemas.

Trabalhe com a energia desligada
(desconecte o cabo USB).

Evite problemas.

Atenção à **polaridade*** dos componentes. Se você inverter (o positivo com o negativo)...

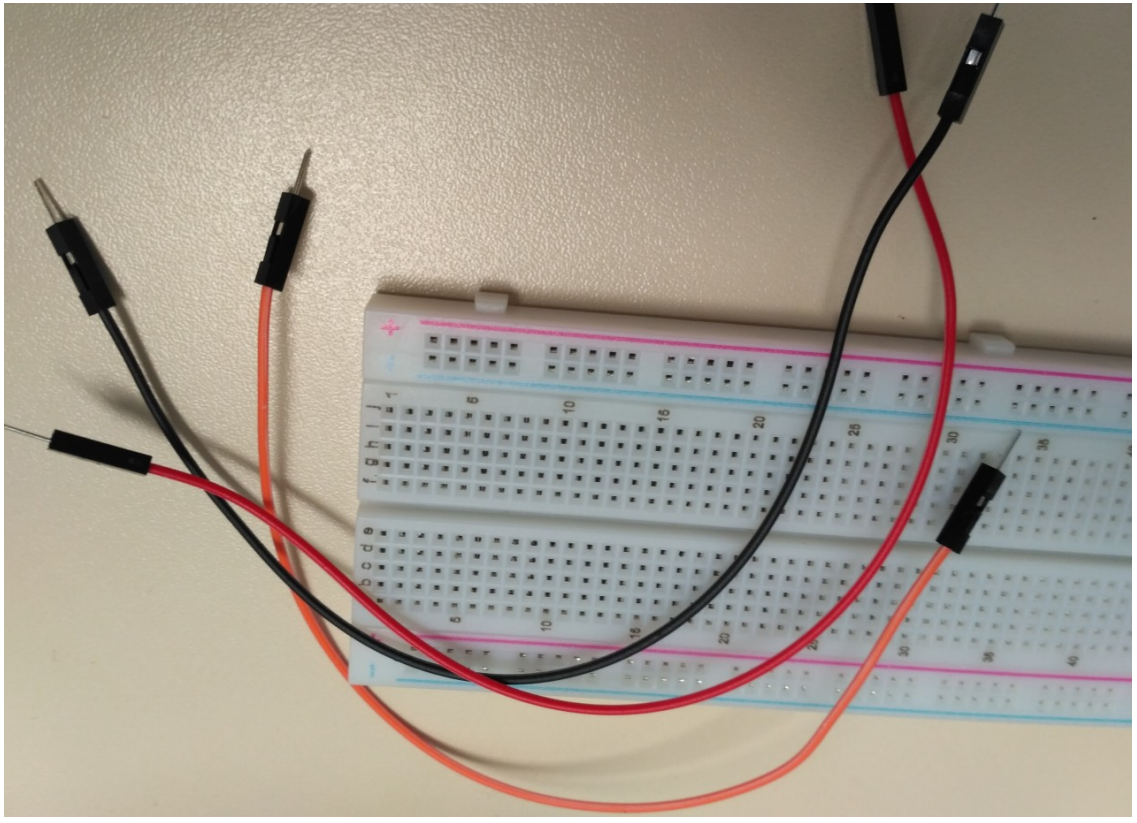


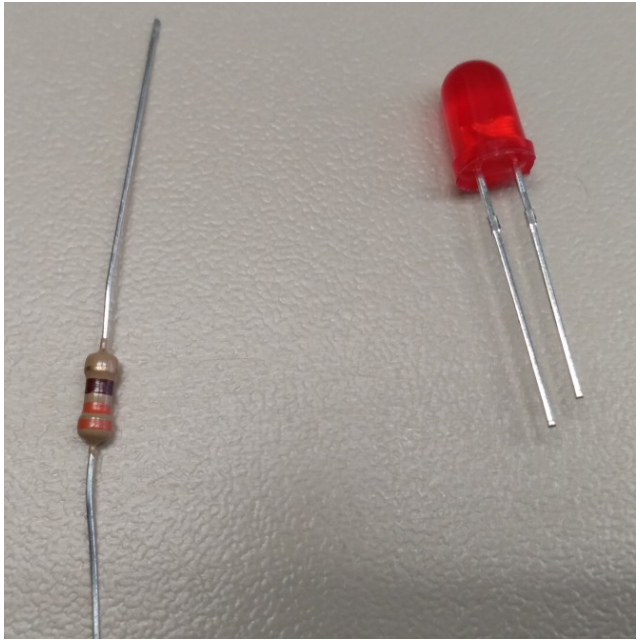
* polo positivo, polo negativo, não pode inverter

Imagem: do autor

simao@ufpr.br - 2025

Componentes necessários





Imagens: foto do autor

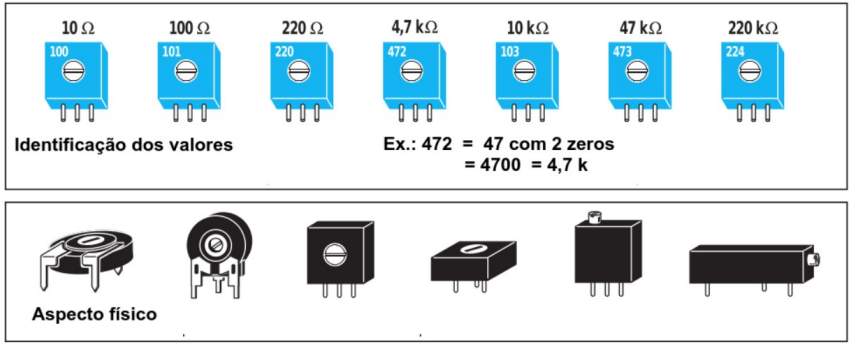
simao@ufpr.br - 2025

Outros componentes

https://tiaplicada.ufpr.br/wp-content/uploads/2025/06/componenteslogicofisico.pdf

Adaptado da revista Electronique et loisires nº 2, 1999.

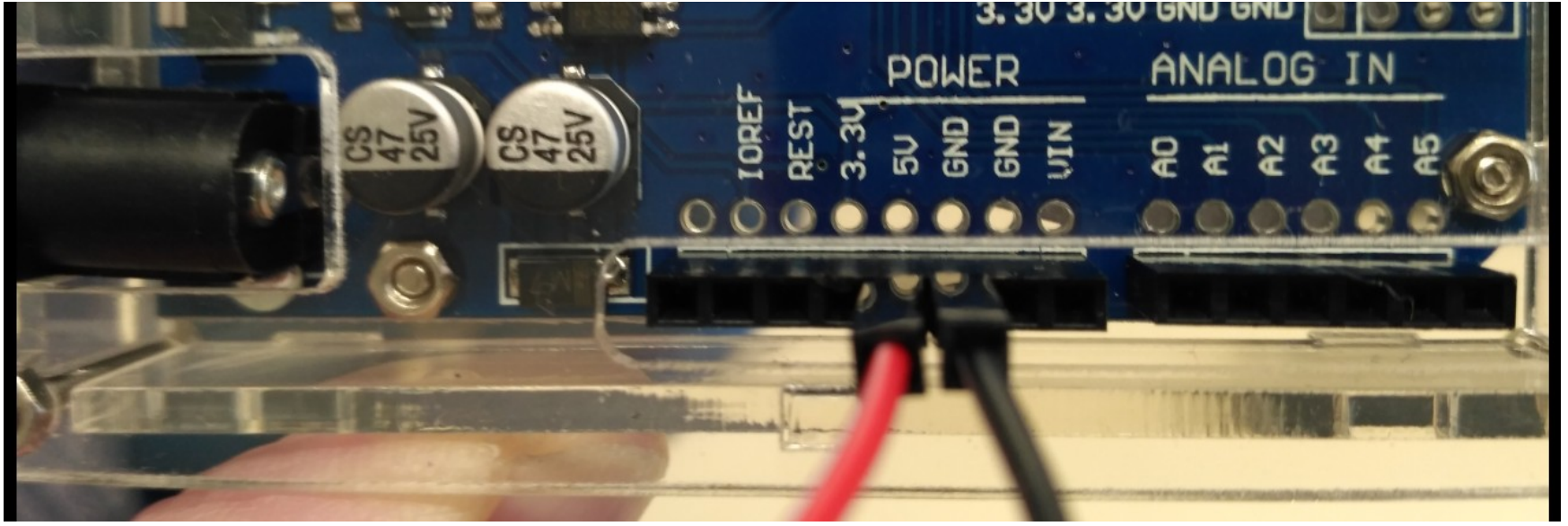
Símbolo	Abr.	Componente	Aspecto físico
	R	Resistência, resistor	
	R ou P	Trimmer ou Resistor ajustável ou Trimpot	
	P ou POT.	Potenciômetro	
	PR LDR	LDR, fotoresistor	
	C	Capacitor/ Condensador (cerâmico ou de poliester)	
	CV	Capacitor variável	



https://tiaplicada.ufpr.br/wp-content/uploads/2025/06/ajustaveis.png

simao@ufpr.br - 2025

1. Ligação de energia

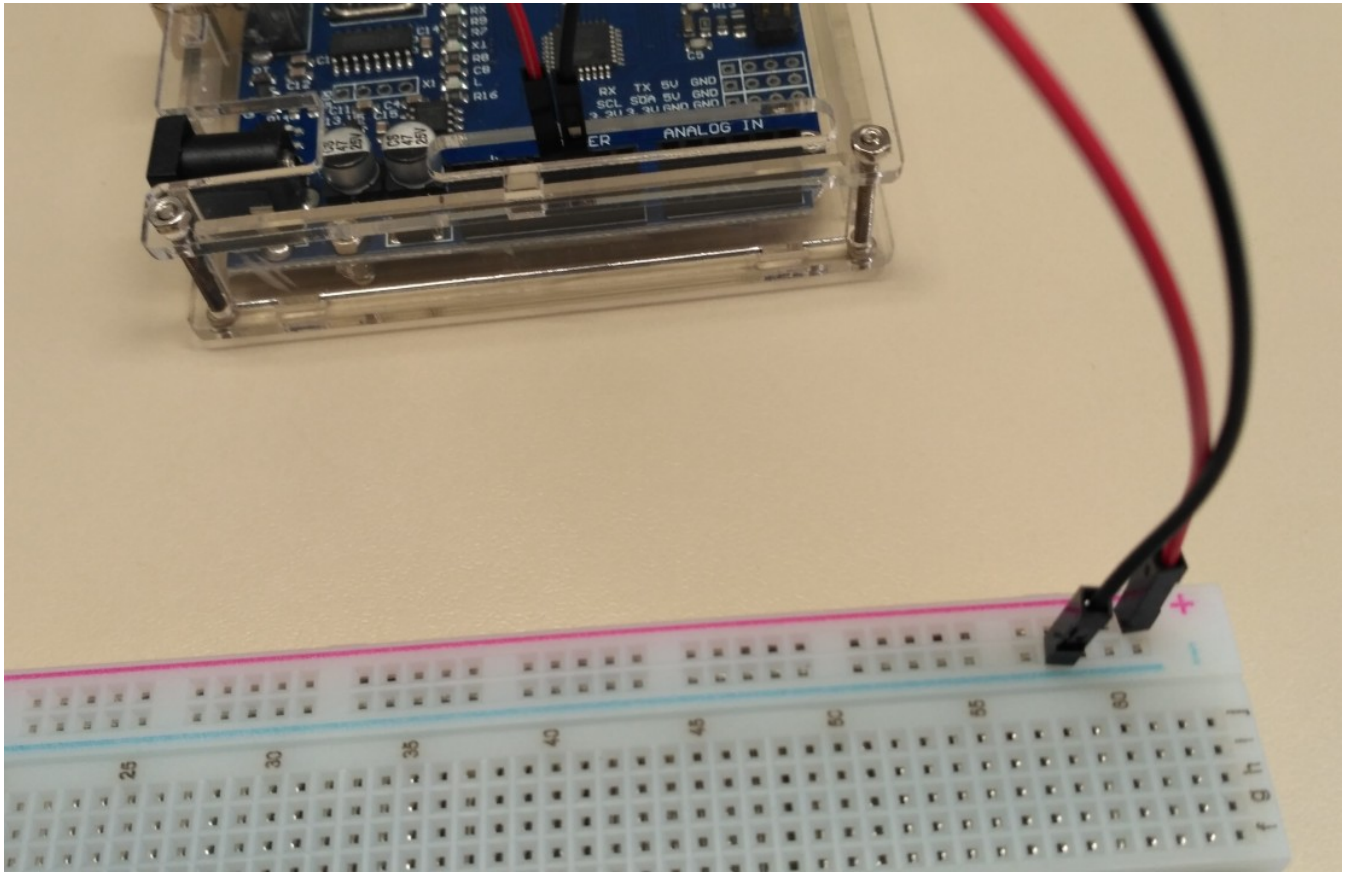


Separe dois condutores (no exemplo **vermelho** e **preto**) e ligue-os aos pontos **5V** (ou VCC) e **GND** (*ground*, 0V) de sua placa Arduino. Em geral, o condutor **vermelho é o positivo** e o **preto é o negativo**.

Imagens: foto do autor

simao@ufpr.br - 2025

1. Ligação de energia



Leve os pontos de energia para placa de experimentação.

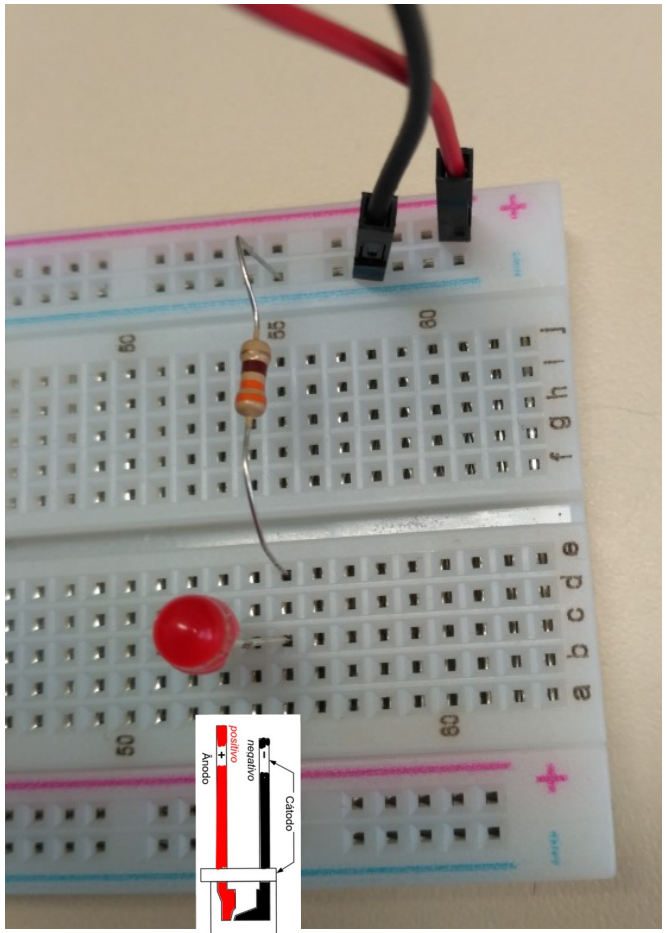
As linhas coloridas no exemplo indicam que os pontos horizontais estão interconectados em uma barra.

Teremos uma linha de alimentação **positiva** (+5V) e outra com o **0V** (que é o lado negativo).

Imagens: foto do autor

simao@ufpr.br - 2025

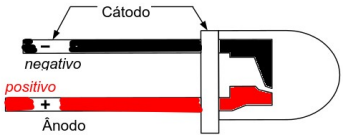
1. Ligação de energia



O resistor não é polarizado, o led é.

Ligue um terminal do resistor na barra de 0V e o outro terminal em uma coluna qualquer (no exemplo, na posição da coluna 55).

Ligue o lado negativo do led (o cátodo, o terminal menor – lado chanfrado) na mesma coluna em que foi ligado o resistor (55 no exemplo ao lado).

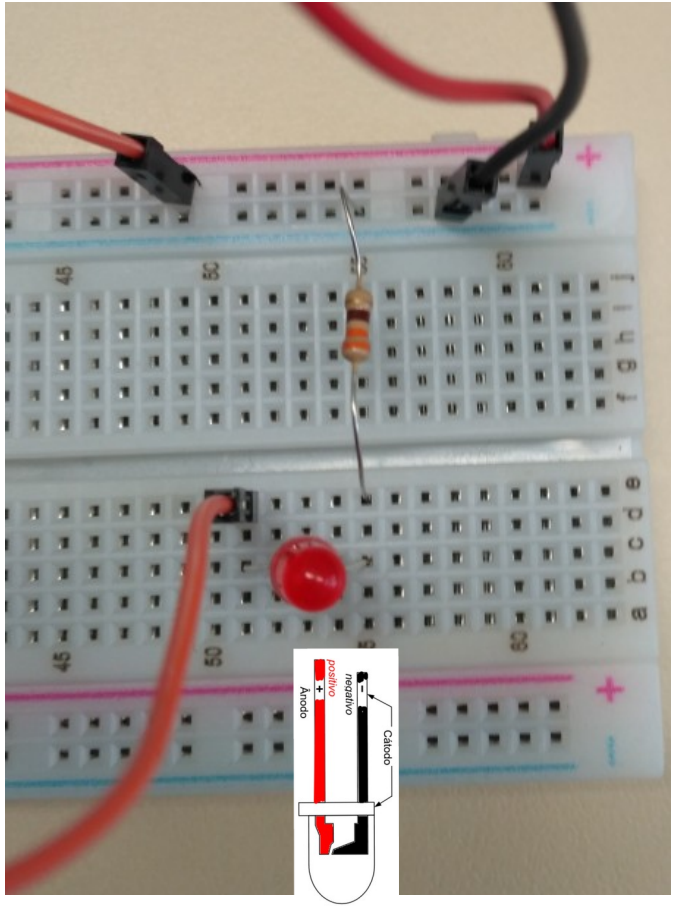


Se os terminais não tiverem sido cortados, o lado maior é o positivo (use uma bateria ou um multímetro para testar se tiver dúvida).

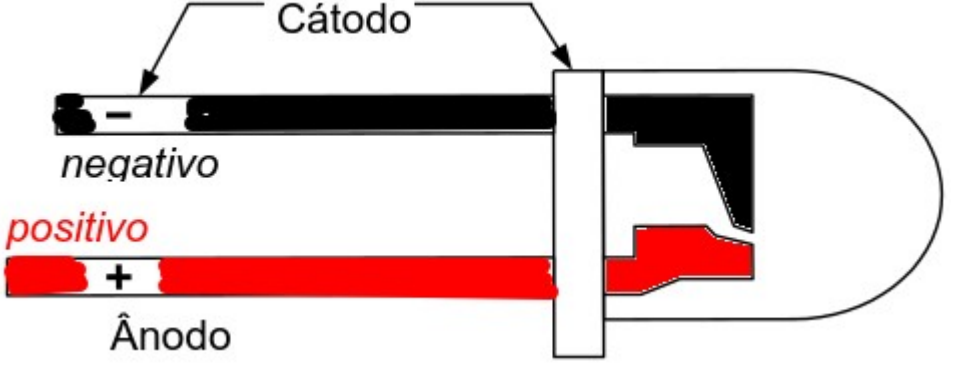
Imagens: foto do autor

simao@ufpr.br - 2025

1. Ligação de energia



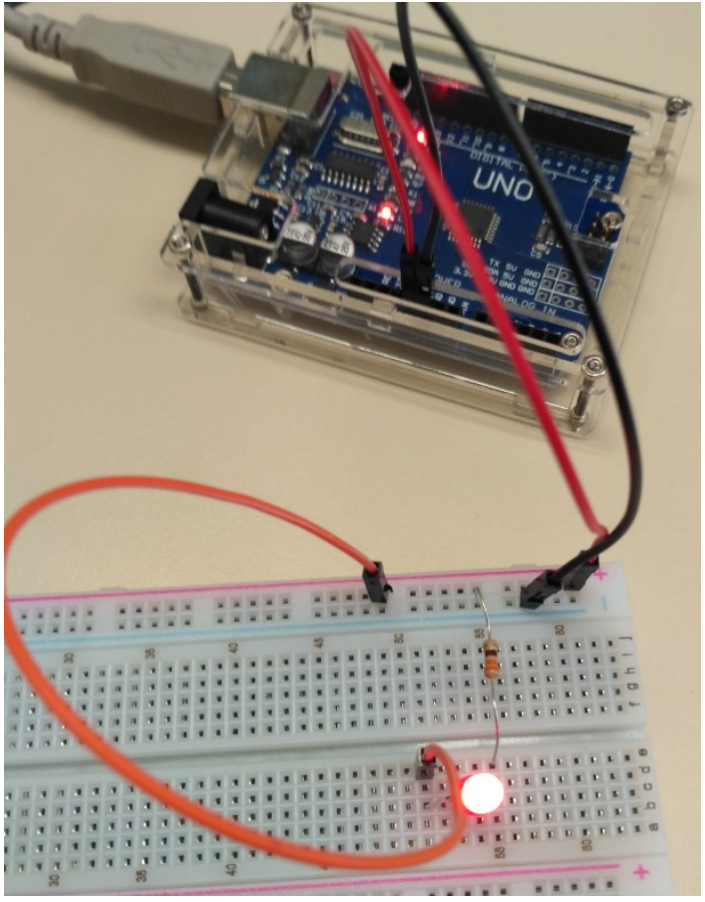
Utilize um fio de ligação para conectar o outro terminal do led (o positivo ou ânodo, terminal mais longo – na figura ao lado ficou na coluna nº 49) até a barra de alimentação positiva (a linha na qual foi conectado o 5V vindo da placa do Arduino).



Imagens: foto do autor

simao@ufpr.br - 2025

1. Ligação de energia



Conecte o cabo USB ao computador.

Não precisa entrar no Ide, somente estamos testando as ligações (e praticando…).

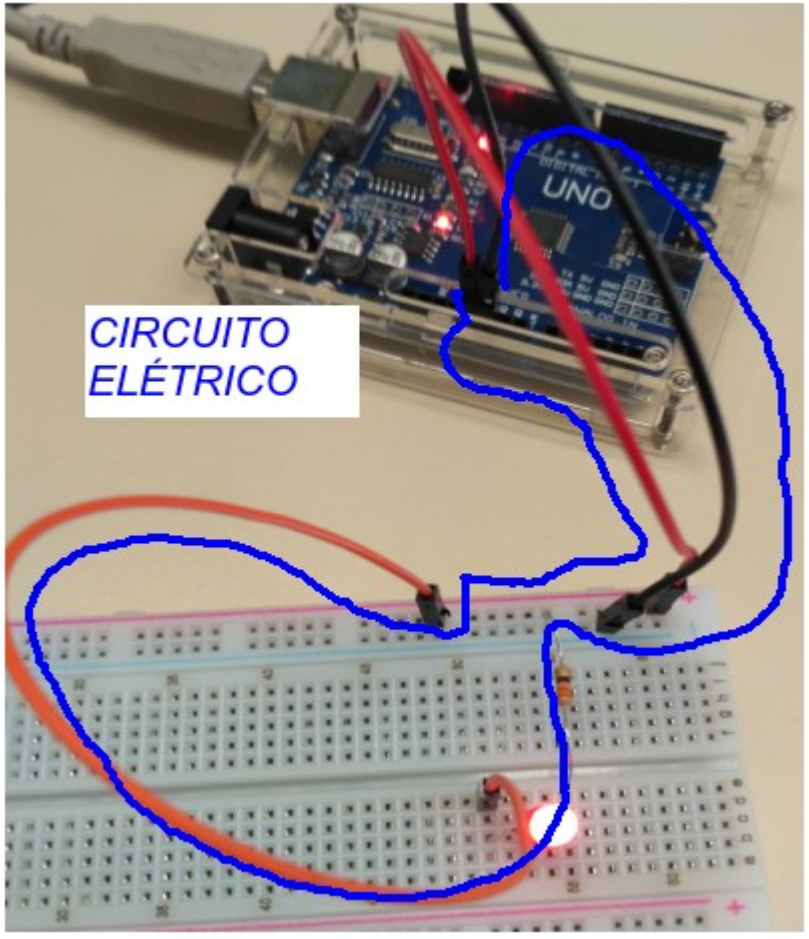
Se não deu certo, desligue o cabo USB e revise as ligações, em especial a do led; troque os fios, eles costumam dar problemas.

SE tudo deu certo, desligue o cabo USB e vá para próxima etapa.

Imagens: foto do autor

simao@ufpr.br - 2025

Circuito elétrico



A energia circula em um circuito elétrico fechado, que vai do pólo GND pelo fio preto (no exemplo) até a protoboard, na barra (‘-’). Da barra menos a energia circula por meio do resistor, que está ligado ao cátodo do LED (lado menor, negativo). A energia passa pelo LED e sai pelo lado do ânodo (lado maior, positivo) e daí é levada até a barra (‘+’) por meio do fio laranja. Da barra (‘+’) sai um fio vermelho que vai até o +5V do Arduino, fechando o circuito e fornecendo alimentação (energia) ao LED.

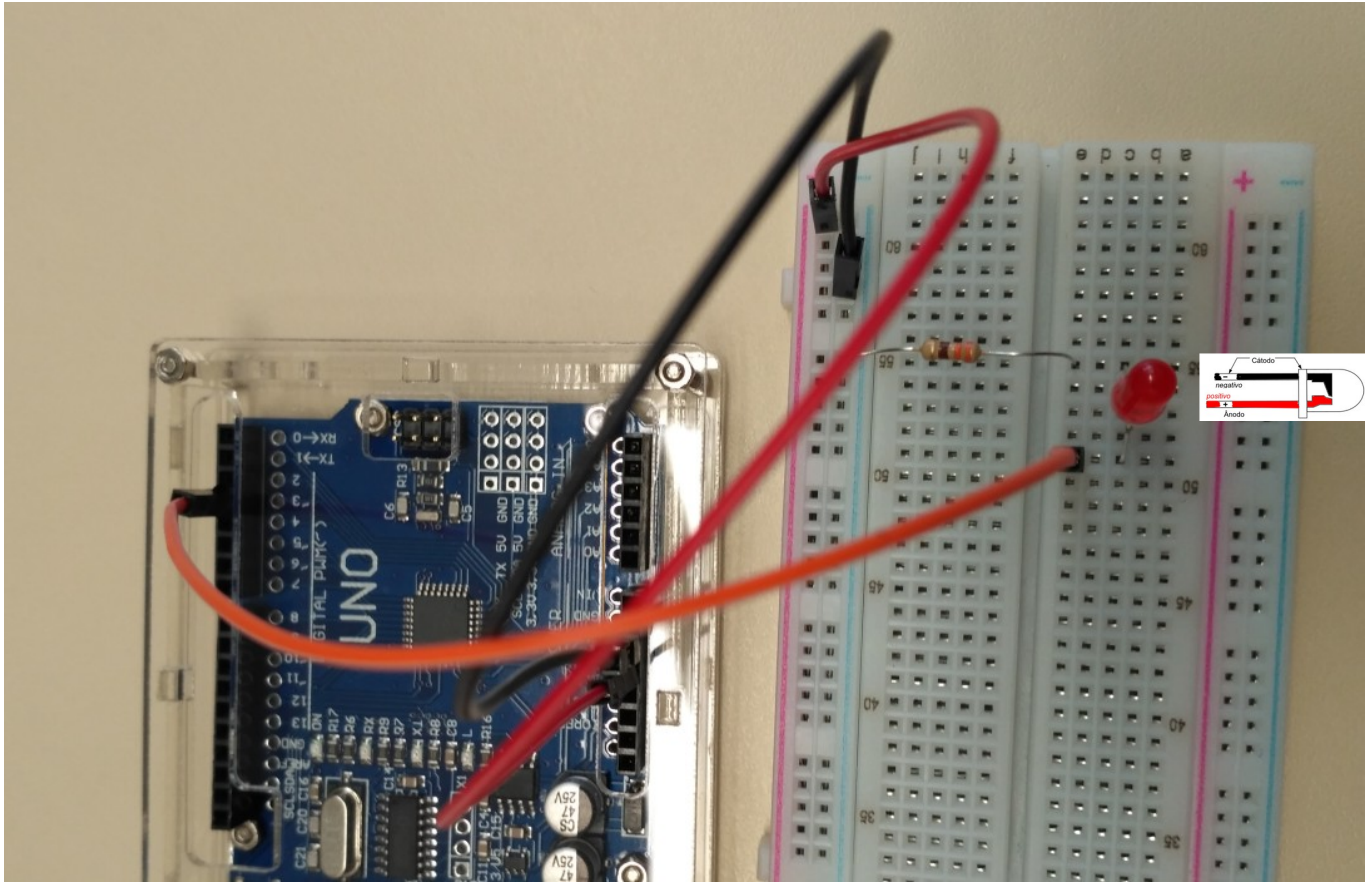
Imagens: foto do autor

simao@ufpr.br - 2025

https://tiaplicada.ufpr.br/iot/introducao-a-eletricidade-com-leds/

Imagens: foto do autor

2. Ligação de controle



Desconecte o cabo USB.

Primeiro, ligue o led no pino 13 e conecte a USB – deverá repetir o led da placa! (lembre-se que que o Arduino repete a última programação).

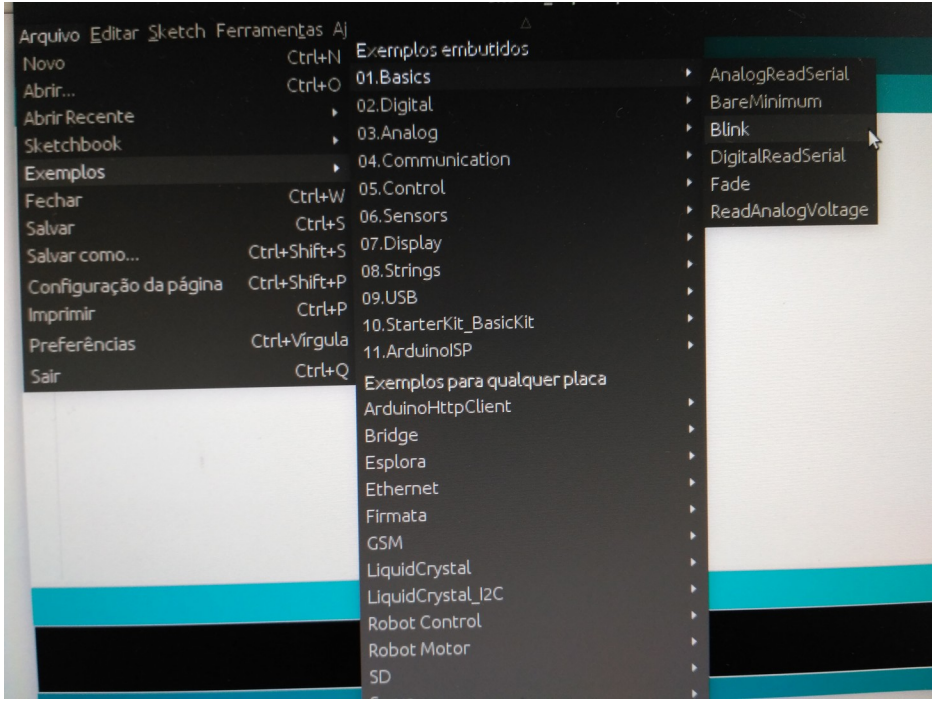
Depois, ligue a conexão led que estava na barra de 5V ao pino de saída digital 3 de seu Arduino. Não vai ligar, pois temos que alterar o código.

Imagens: foto do autor

simao@ufpr.br - 2025

3. Codificação

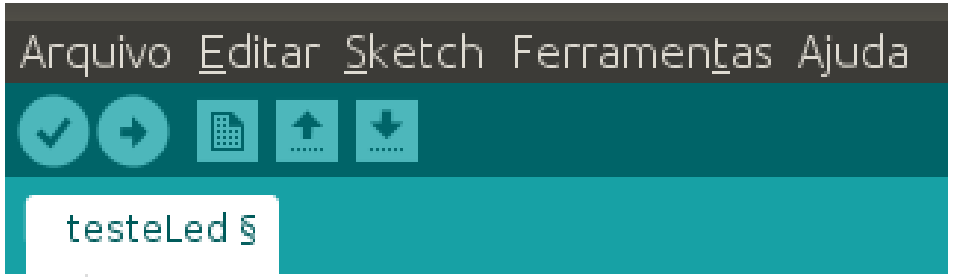
Inicie o IDE e carregue o seu primeiro exemplo, de piscar o led da placa (Arquivo/ Exemplos/ 01.Basics/ Blink)



Imagens: foto do autor

simao@ufpr.br - 2025

3. Codificação




```
1
2  void setup() {
3
4    pinMode(LED_BUILTIN, OUTPUT);
5  }
6
7
8  void loop() {
9    digitalWrite(LED_BUILTIN, HIGH);
10   delay(1000);
11   digitalWrite(LED_BUILTIN, LOW);
12   delay(1000);
13 }
```

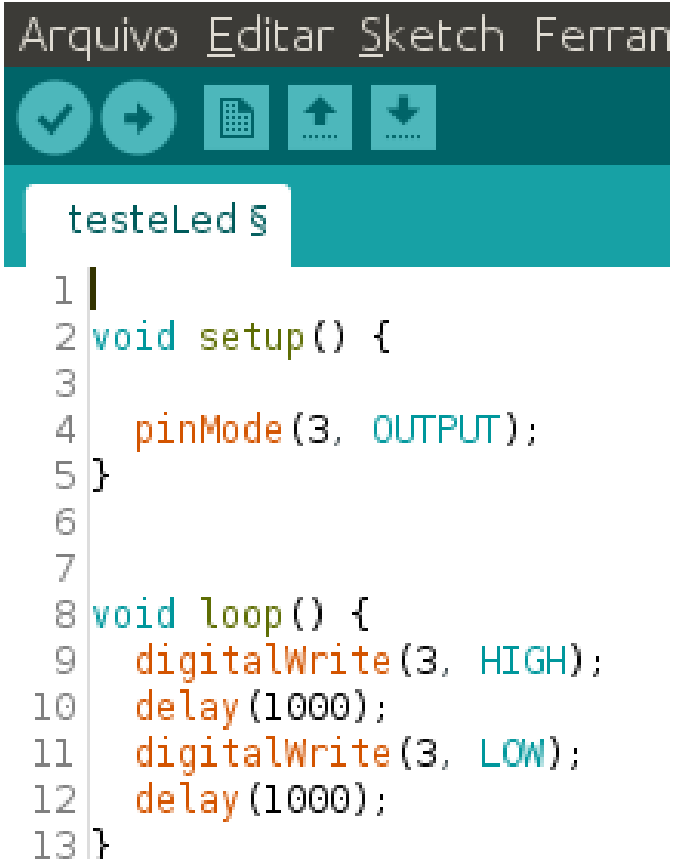
Salve o arquivo com outro nome (Arquivo/ Salvar Como…).

No exemplo ao lado todas as linhas de comentários forma removidas para podermos nos concentrar no código.

Este exemplo utiliza o mesmo conteúdo do exemplo ‘*blink*’, já visto – somente os núemros das linhas foram trocados.

Imagens: foto do autor

3. Codificação



Defina o pino 3 como saída, compile e envie para o Arduino, com a ligação da placa experimental já efetuada e conferida.

Imagens: foto do autor

simao@ufpr.br - 2025

Parabéns!

Bem vindo ao mundo *maker* do Arduino.

https://wokwi.com/projects/new/arduino-uno

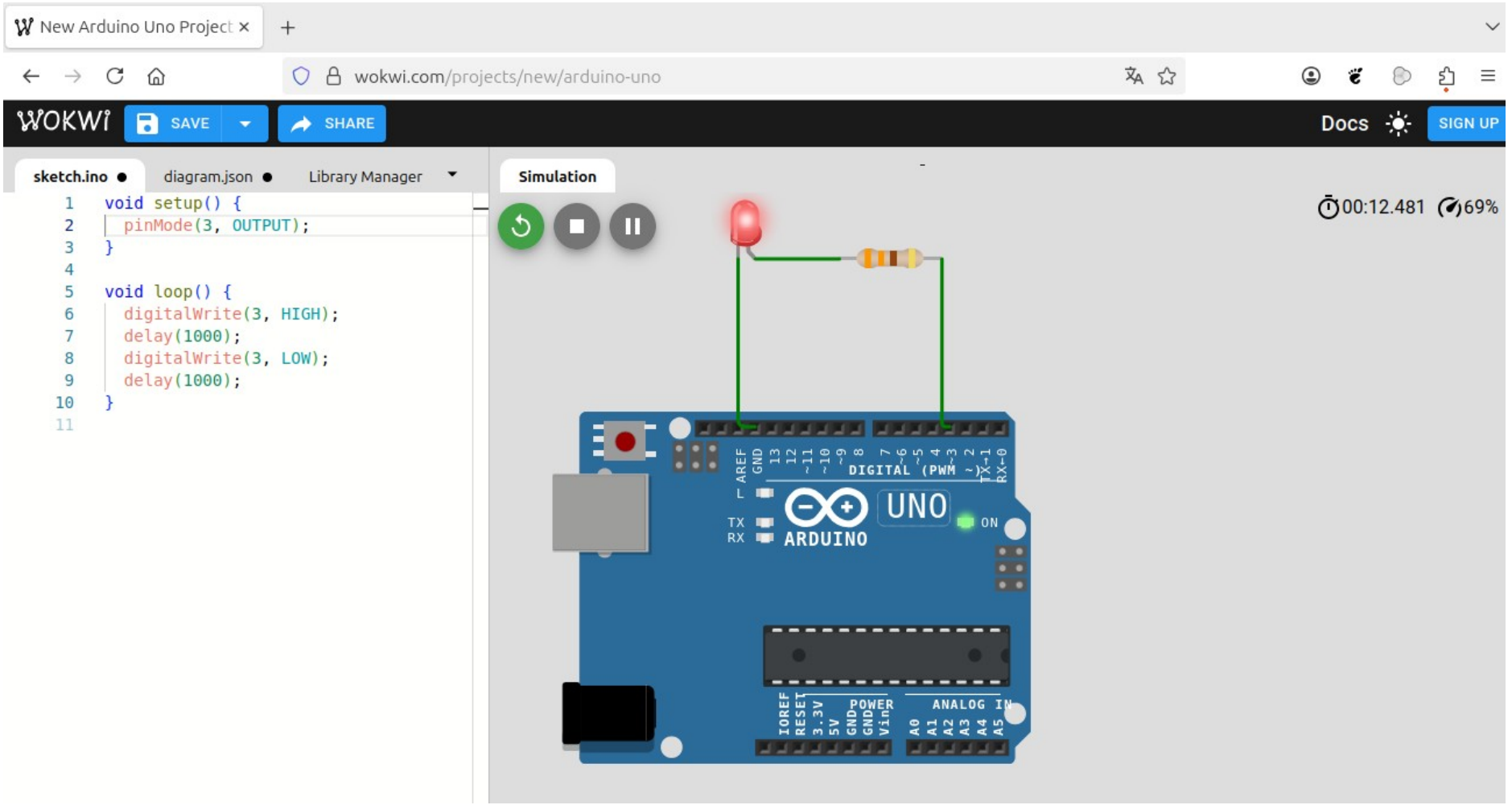
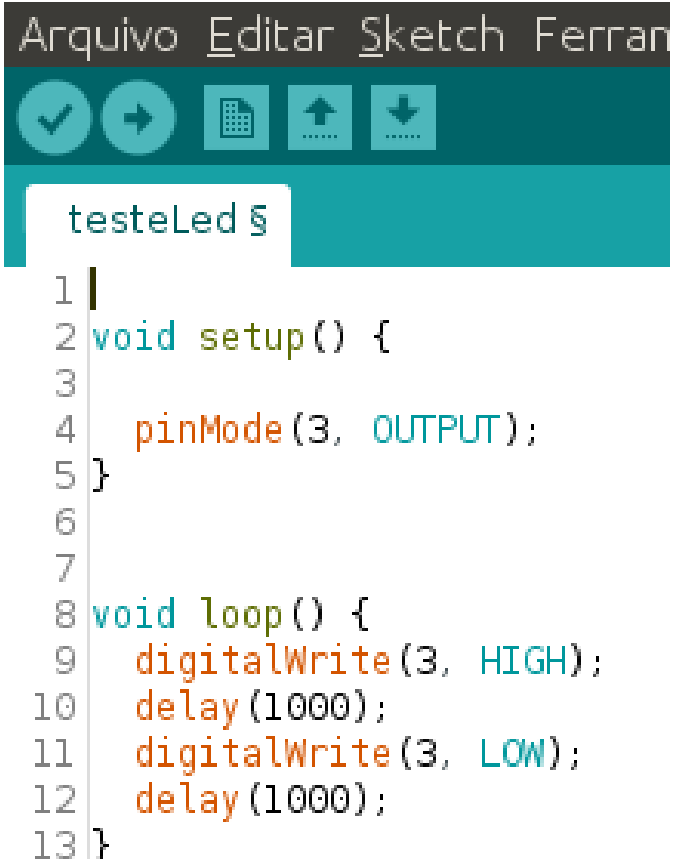


Imagem: cópia de tela de execução de simulação no Wokwi efetuada pelo autor

Melhorando o código



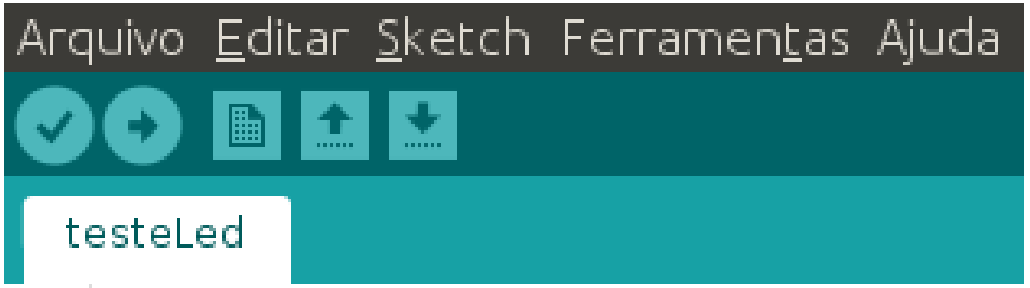
No código original havia uma palavra reservada, `LED_BUILTIN`, a qual corresponde à porta 13 (um led na placa).

Nós, explicitamente, utilizamos o número da porta que escolhemos, 3.

Mas, podemos também dar um ‘nome’ para nossa porta.

Imagens: foto do autor

Melhorando o código




```
1  #define nossoled 3
2  void setup() {
3
4    pinMode(nossoled, OUTPUT);
5  }
6
7
8  void loop() {
9    digitalWrite(nossoled, HIGH);
10   delay(1000);
11   digitalWrite(nossoled, LOW);
12   delay(1000);
13 }
```

Foi criada uma constante nosso led, e atribuída para ela o valor 3.

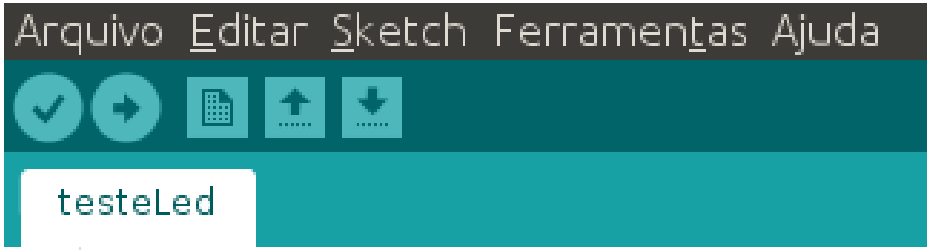
Isto foi feito com a declaração *`#define`*.

Veja ao lado como ficou.

Teste!

Imagens: foto do autor

Agora, com dois leds




```
1  #define nossoled 3
2  void setup() {
3
4    pinMode(nossoled, OUTPUT);
5  }
6
7
8  void loop() {
9    digitalWrite(nossoled, HIGH);
10   digitalWrite(LED_BUILTIN, LOW);
11   delay(1000);
12   digitalWrite(nossoled, LOW);
13   digitalWrite(LED_BUILTIN, HIGH);
14   delay(1000);
15 }
```

Com o código ao lado, queremos um led apagando (`LOW`) e outro acendendo (`HIGH`), alternadamente.

Teste!

Imagens: foto do autor

simao@ufpr.br - 2025

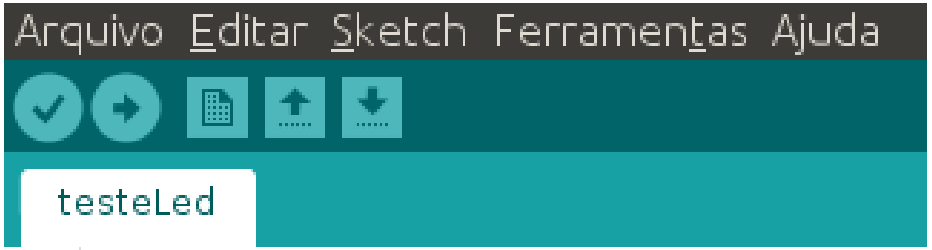
???

E se não funcionou

???

simao@ufpr.br - 2025

Agora com dois leds




```
1  #define nossoled 3
2  void setup() {
3
4    pinMode(nossoled, OUTPUT);
5  }
6
7
8  void loop() {
9    digitalWrite(nossoled, HIGH);
10   digitalWrite(LED_BUILTIN, LOW);
11   delay(1000);
12   digitalWrite(nossoled, LOW);
13   digitalWrite(LED_BUILTIN, HIGH);
14   delay(1000);
15 }
```

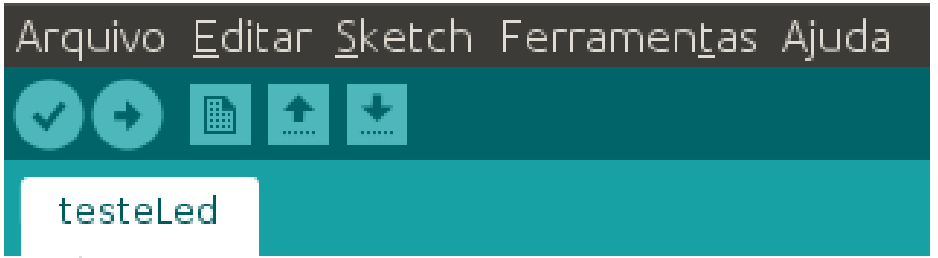
Definimos que haverá um comando de escrita na saída, na função *loop()*.

Porém, não definimos na função *setup()* que a saída da placa estaria ligada…

As duas funções funcionam em conjunto...

Imagens: foto do autor

Agora com dois leds




```
1  #define nossoled 3
2  void setup() {
3    pinMode(LED_BUILTIN, OUTPUT);
4    pinMode(nossoled, OUTPUT);
5  }
6
7
8  void loop() {
9    digitalWrite(nossoled, HIGH);
10   digitalWrite(LED_BUILTIN, LOW);
11   delay(1000);
12   digitalWrite(nossoled, LOW);
13   digitalWrite(LED_BUILTIN, HIGH);
14   delay(1000);
15 }
```

Acerte o código conforme ao lado.

Teste!

Imagens: foto do autor

HIGH ou LOW ?

- SE você ligar um lado do LED no GND, que é equivalente ao 0V, para que o LED acenda o outro lado deve estar positivo.
  - Portanto, acionamos o LED com o comando `HIGH` (que corresponde a colocar +5V na saída).
  - … mas, ...

simao@ufpr.br - 2025

HIGH ou LOW?

- Também podemos deixar um lado do LED ligado no +5V e ligar o outro lado no Arduino, usando uma lógica negativa:
  - Quando colocarmos um sinal `LOW` no pino em que foi ligado o LED ele acenderá.
- Usar a lógica positiva ou a negativa é indiferente, depende de com qual você se sente mais à vontade.

simao@ufpr.br - 2025

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

Escrita PWM

simao@ufpr.br - 2025

PWM

- PWM significa "*Pulse Width Modulation*"
- A Modulação por Largura de Pulso possibilita que controlemos quanto de energia (0 a 100%) será fornecida dentro de um pulso (qual será sua largura).
- Vamos aproveitar nosso protótipo para testar esta funcionalidade.

simao@ufpr.br - 2025

Ciclo de trabalho – *duty cicle*

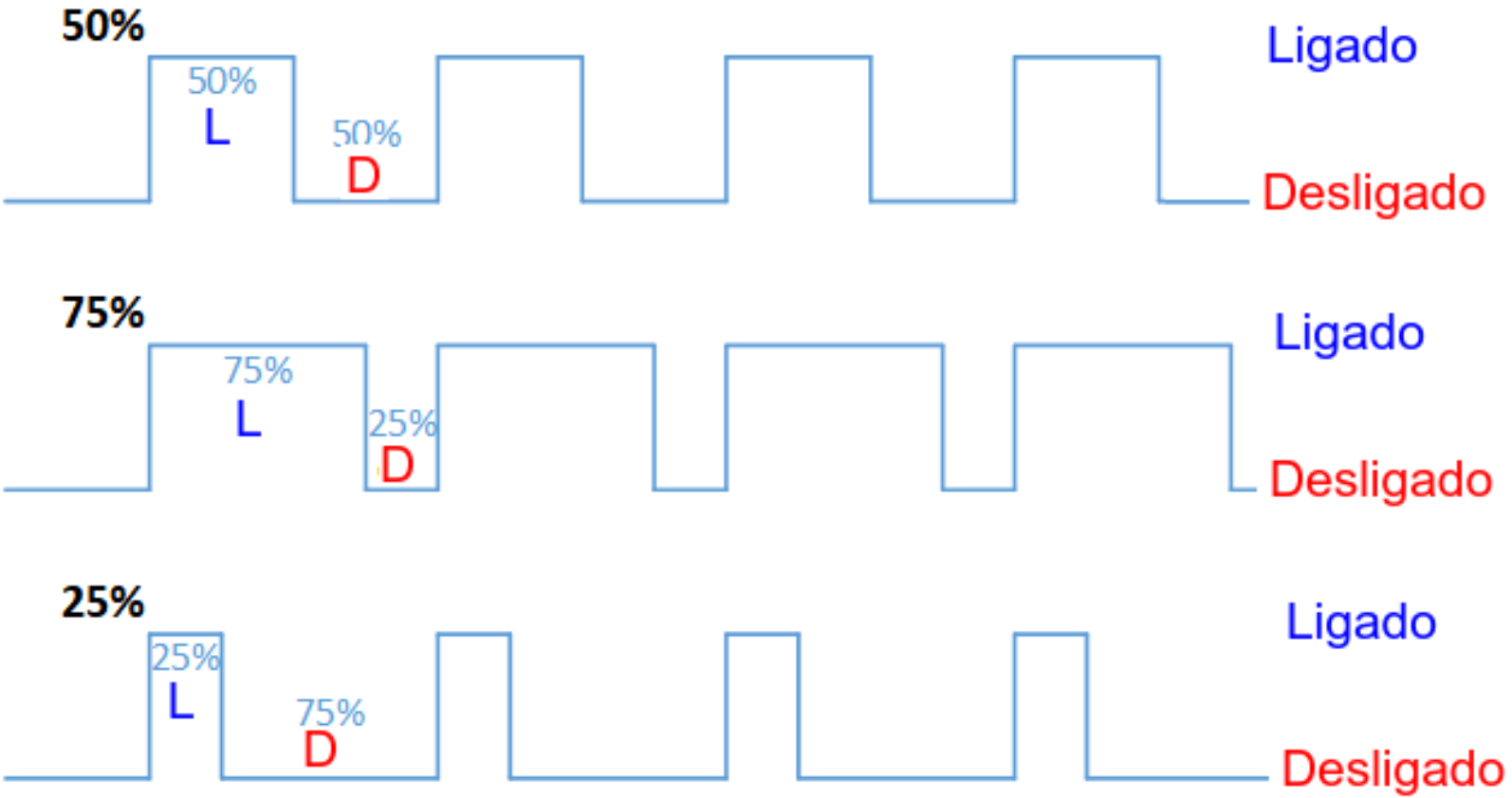
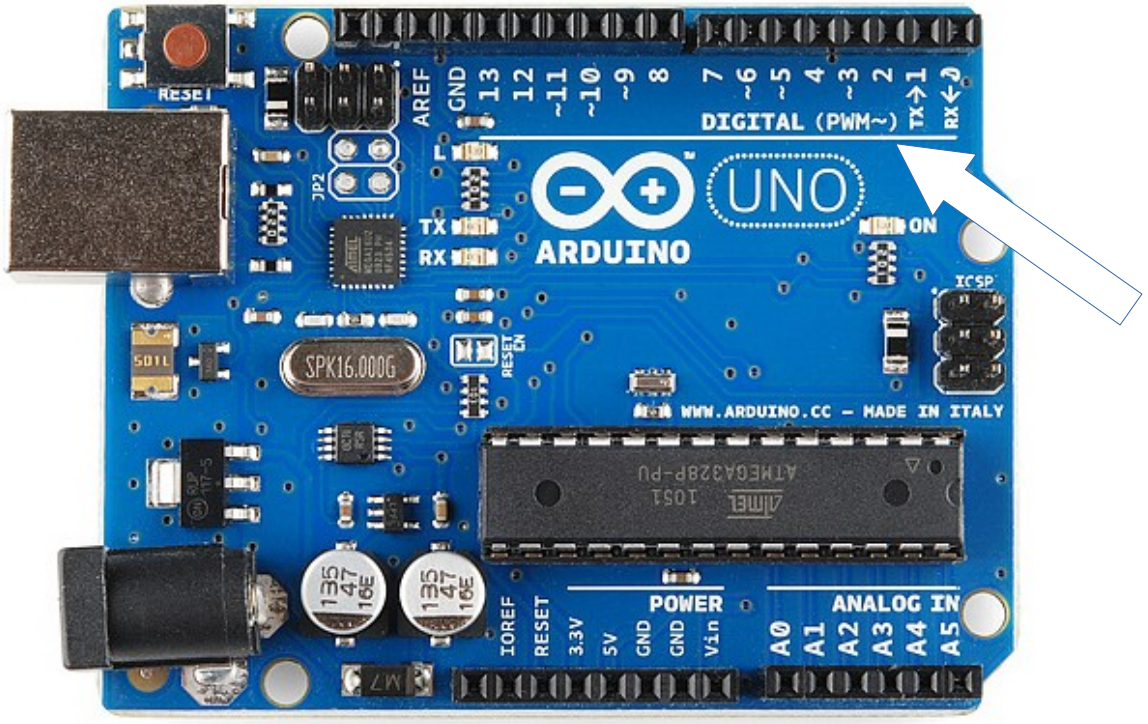


Imagem: A posteriori

simao@ufpr.br - 2025

Portas PWM (indicadas com ‘~’)

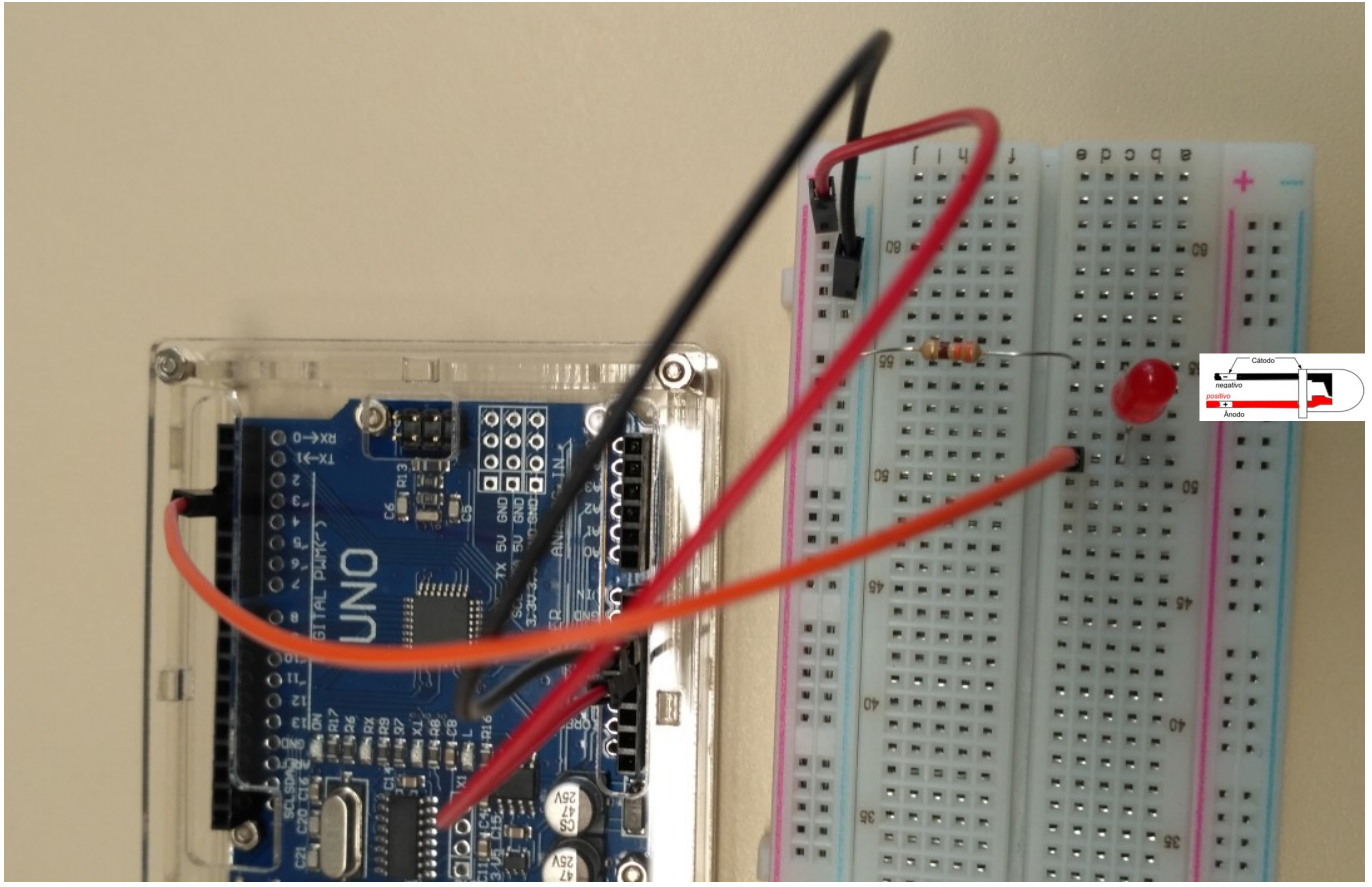


A indicação usual das portas que possuem a capacidade PWM é por meio do sinal ‘til’ antes do número da porta (‘~’)

No Arduino Uno são as portas: 3,5,6,9,10 e 11

Imagem: A posteriori

PWM



Usaremos a mesma conexão (perceba que o pino ligado ao 5V não está sendo usado...).

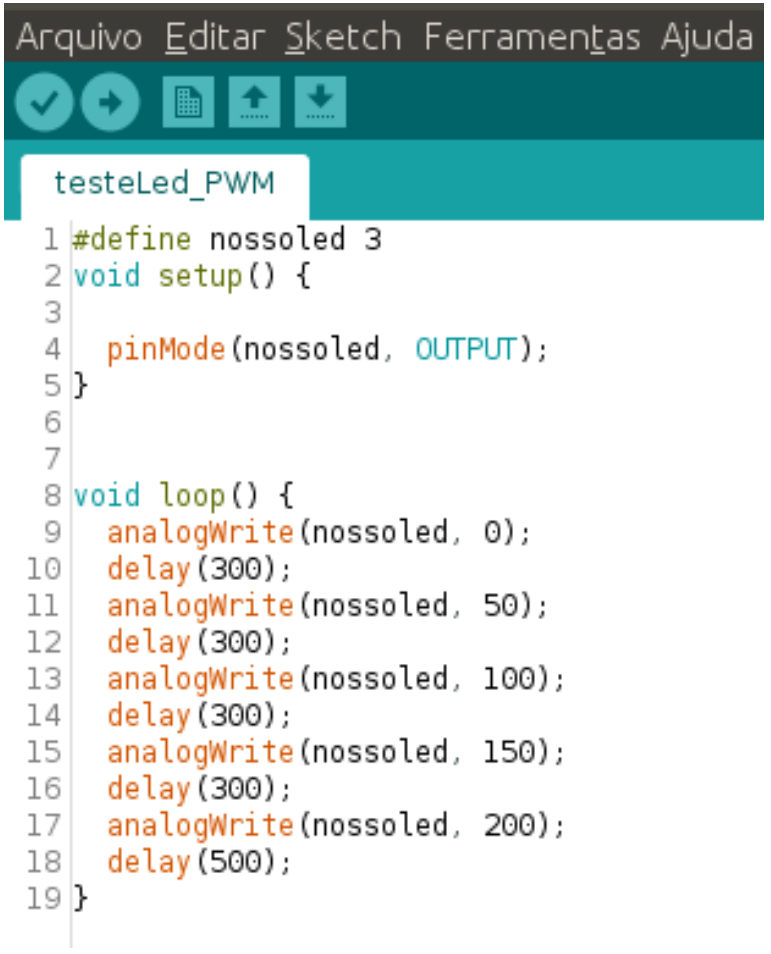
Não precisa desconectar.

Salve seu programa com outro nome.

Imagens: foto do autor

simao@ufpr.br - 2025

PWM



A função de escrita agora é a *`analogWrite(porta, valor)`*.

O valor é a quantidade do ciclo de trabalho PWM que ficará ativo (em nível `HIGH`).

Teste!

Imagens: foto do autor

PWM

Arquivo Editar Sketch Ferramentas Ajuda

testeLed_PWM_continuo

```
1  #define nossoled 3
2  void setup() {
3
4    pinMode(nossoled, OUTPUT);
5  }
6
7
8  void loop() {
9    int qtdade = 0;
10   for (qtdade = 0; qtdade < 256; qtdade++)
11   {
12         analogWrite(nossoled, qtdade);
13         delay(50);
14   }
15 }
```

Agora transformamos o valor PWM em uma variável inteira, chamada ‘qtdade’.

E colocamos um *loop* controlado, que varia o valor armazenado em ‘qtdade’ de 0 a 255, continuamente – na linha 10, por meio do comando `for` (valor inicial; teste lógico/ limite; valor do incremento).

Teste!

(obs.: `qtdade++` significa ‘some um ao valor atual da variável qtdade’).

Imagens: foto do autor

for

```
10  for (qtdade = 0; qtdade < 256; qtdade++)
11  {
12        analogWrite(nossoled, qtdade);
13        delay(50);
14  }
```

`for` (valor inicial; teste lógico/ limite; valor do incremento)

No exemplo, a variável qtdade vai controlar o comando for; ela inicia com um valor 0, e, enquanto for menor do que 256, o comando é executado. A cada momento em que o comando for executado a variável será incrementada de 1 (`qtdade++`).

A execução do comando corresponde à execução do bloco de comandos contido entre as linhas 11 e 14 (delimitados pelas chaves), nesta caso a linha 12, que escreve o valor atual da variável quantidade na porta 'nossoled', e a linha 15 que força uma espera de 50 milissegundos.

simao@ufpr.br - 2025

Imagens: foto do autor

De 0 a 255

A porta de escrita analógica trabalha com uma **resolução** de 8 bits.

Isto significa uma combinação de $2^8 = 256$ valores possíveis.

Como o número 0 é a primeira combinação, sobram 255 combinações, indo, então, de 1 a 255 as combinações restantes.

Desta forma, com 8 bits de resolução teremos a codificação de números entre 0 e 255.

simao@ufpr.br - 2025

PWM

Arquivo Editar Sketch Ferramentas Ajuda

testeLed_PWM_continuoSobeDesce

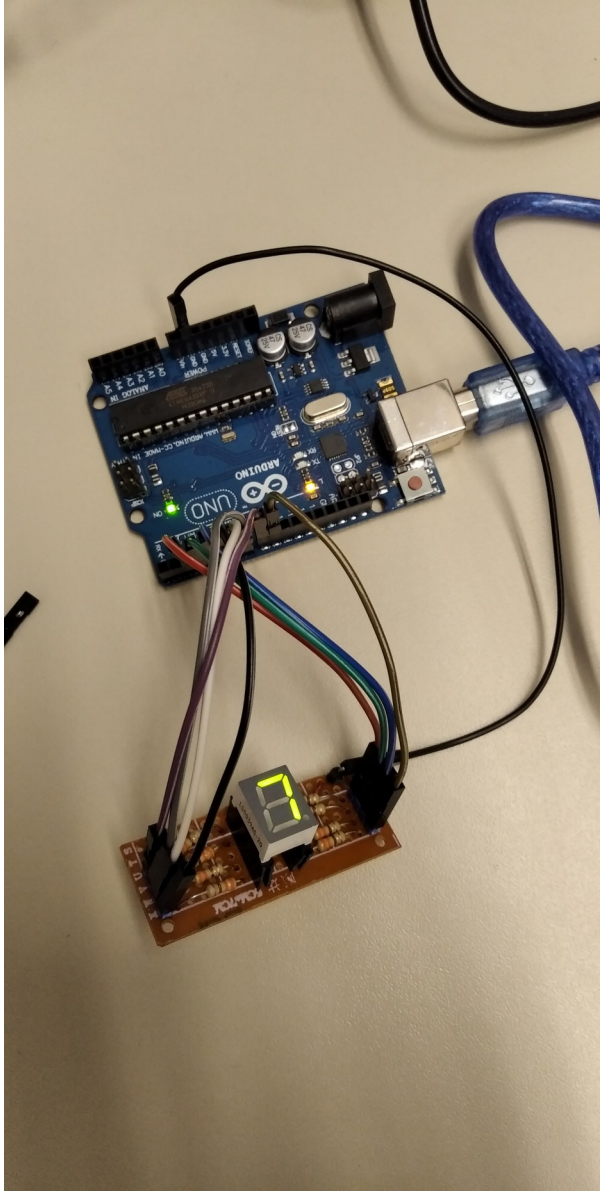
```
1  #define nossoled 3
2  void setup() {
3
4    pinMode(nossoled, OUTPUT);
5  }
6
7
8  void loop() {
9    int qtdade = 0;
10   for (qtdade = 0; qtdade < 256; qtdade++)
11   {
12         analogWrite(nossoled, qtdade);
13         delay(30);
14   }
15   for (qtdade = 255; qtdade>=0 ; qtdade--)
16   {
17         analogWrite(nossoled, qtdade);
18         delay(30);
19   }
20 }
```

Agora o código aumentará o valor até o máximo e depois diminuirá (note o uso de `qtdade++` e `qtdade--`, bem como o teste utilizado no `for` das linhas 10 e 15).

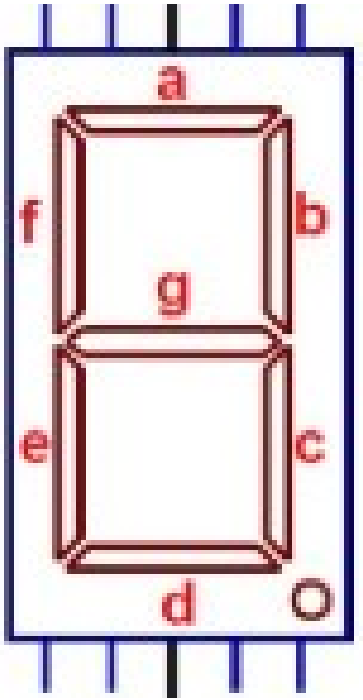
Teste!

Experimente comentar as linhas nas quais ocorre o `delay(30)` e/ ou alterar seu valor e veja o resultado. Teste sem o `delay` transformando-o em comentário
(para comentar, coloque um '//' antes do comando, ele ficará cinzento e passará a ser um comentário – ou seja, não será funcional).
Ajuste da forma que ficar melhor para você.

Imagens: foto do autor



Display de 7 segmentos

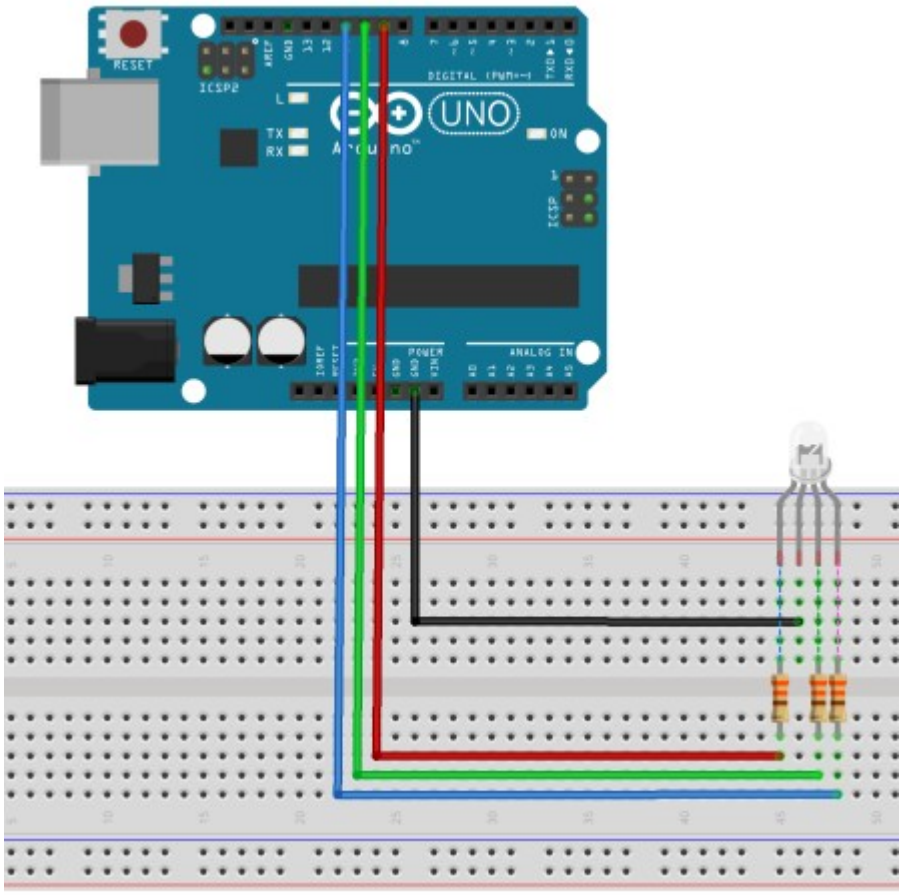


Para formar um número basta acender os leds correspondentes aos segmentos (mais o ponto decimal)

Imagem: do autor

simao@ufpr.br - 2025

LED RGB



```
#define vermelho 9
#define verde 10
#define azul 11

int aleatorio;

void setup() {
  Serial.begin(9600);
  pinMode(vermelho, OUTPUT);
  pinMode(verde, OUTPUT);
  pinMode(azul, OUTPUT);
  randomSeed(analogRead(0));
}

void loop() {
  aleatorio = random(256);
  analogWrite(vermelho, aleatorio);
  delay(100);
  aleatorio = random(256);
  analogWrite(verde, aleatorio);
  delay(100);
  aleatorio = random(256);
  analogWrite(azul, aleatorio);
  delay(100);
}
```

Imagem: do autror, pelo Fritzing

LED RGB

```
#define vermelho 9
#define verde 10
#define azul 11

int aleatorio;

void setup() {
  Serial.begin(9600);
  pinMode(vermelho, OUTPUT);
  pinMode(verde, OUTPUT);
  pinMode(azul, OUTPUT);
  randomSeed(analogRead(0));
}

void loop() {
  aleatorio = random(256);
  analogWrite(vermelho, aleatorio);
  delay(100);
  aleatorio = random(256);
  analogWrite(verde, aleatorio);
  delay(100);
  aleatorio = random(256);
  analogWrite(azul, aleatorio);
  delay(100);
}
```

O comando `randomSeed` serve para inicializar o gerador de números aleatórios. No exemplo, foi passado como parâmetro uma leitura analógica, o que fará com que seja lido um 'ruído', inicializando o gerador com um número qualquer.

A variável inteira 'aleatorio' irá receber um valor gerado pela instrução `random`, que será menor do 256 (entre 0 e 256). Pode-se obter valores entre um número e um limite, como, por exemplo, `random(30,100)`, que vai gerar um número entre 30 e 99.

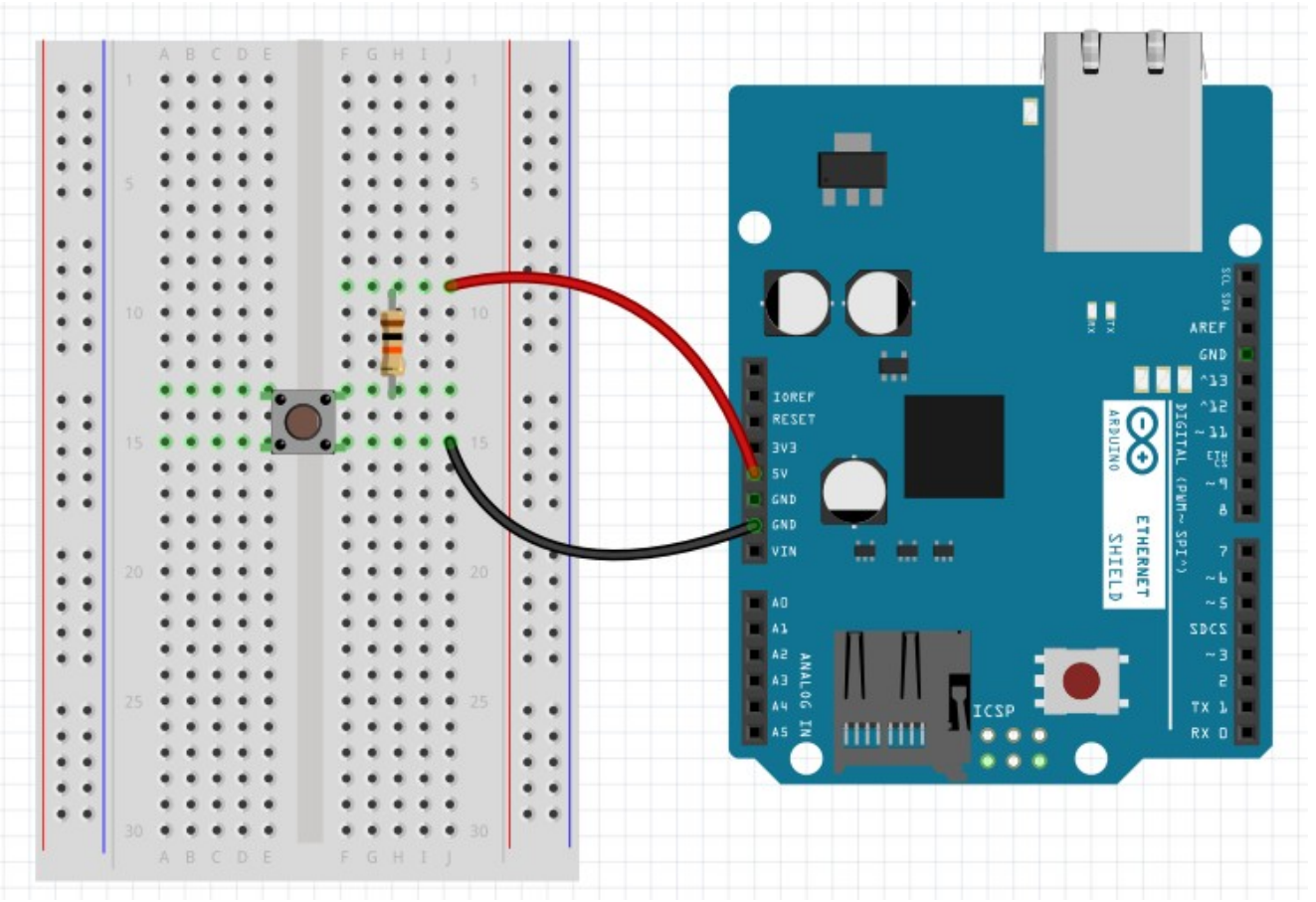
simao@ufpr.br - 2025

Ler um valor digital

simao@ufpr.br - 2025

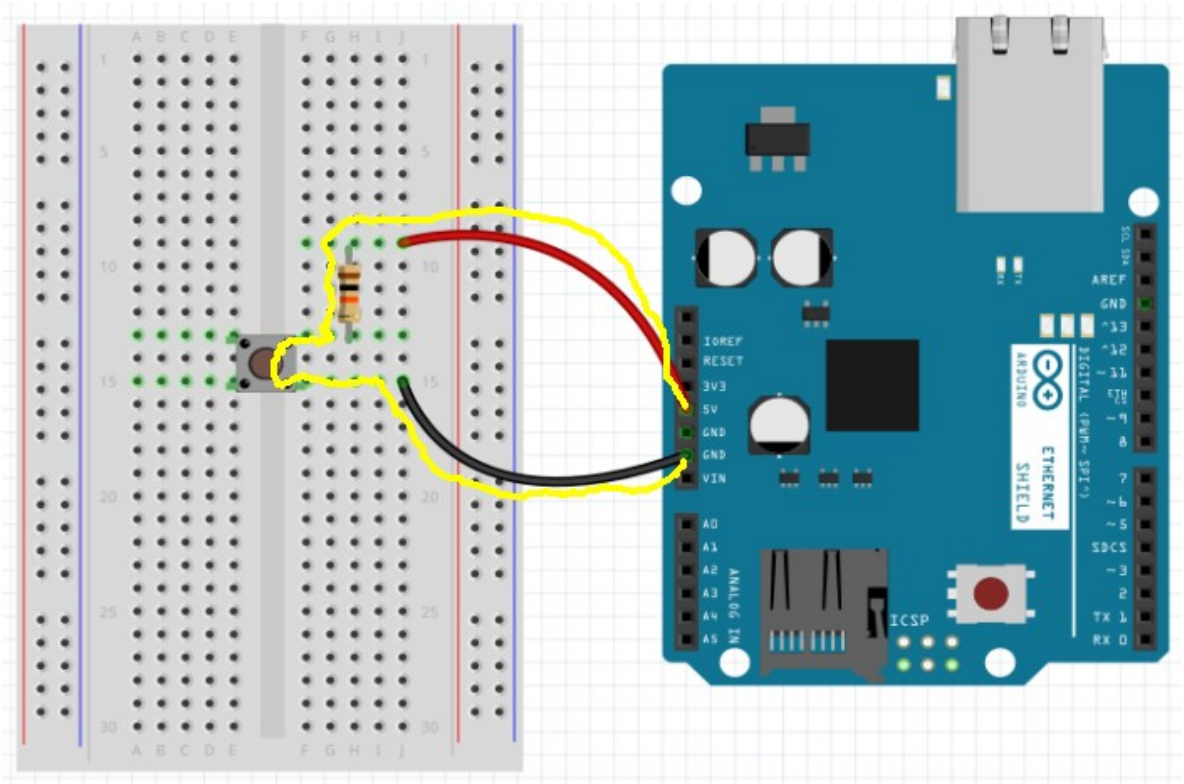
Circuito base

O resistor do exemplo é de 10000 ohms, ou 10k (marrom, preto, laranja).



simao@ufpr.br - 2025

Circuito elétrico



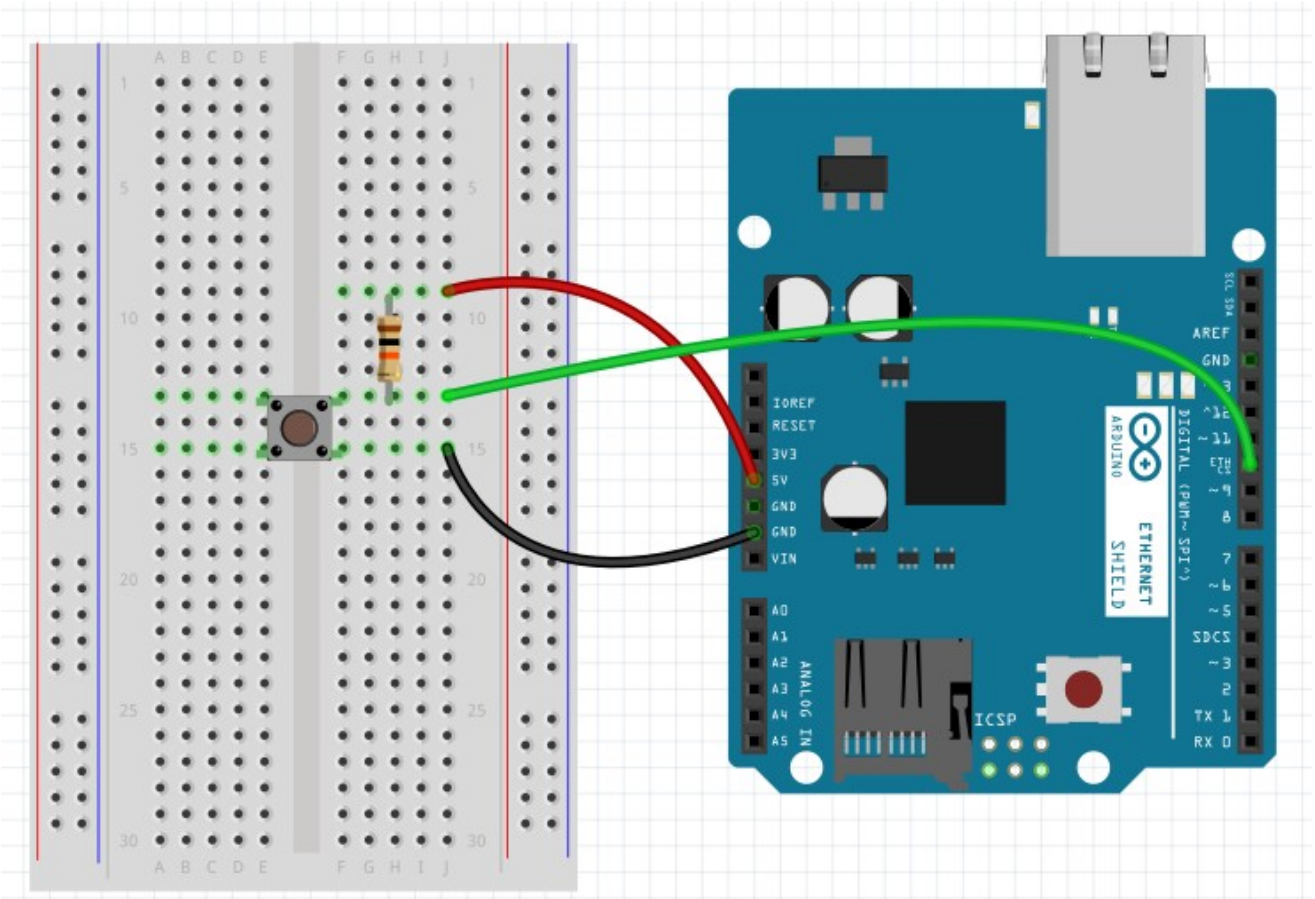
A energia vai do pólo GND pelo fio preto (no exemplo) até a protoboard e até o interruptor. Passa pelo interruptor, pelo resistor, e retorna para o +5V do Arduino, fechando o circuito.

Não haverá curto circuito quando o interruptor for pressionado pois a corrente passará pelo resistor.

Só haverá corrente quando o interruptor for pressionado.

simao@ufpr.br - 2025

Circuito elétrico



O fio verde (no exemplo) está ligado na oirta 10. O nível do sinal será de 5V (por meio da ligação do resistor ao 5V do Arduino) enquanto o interruptor não estiver pressionado.

Quando o interruptor for pressionado o nível de tensão da porta 10 será levado ao nível GND.

simao@ufpr.br - 2025

Código

```
1  void setup() {
2    pinMode(10,INPUT);
3    pinMode(LED_BUILTIN,OUTPUT);
4    digitalWrite(LED_BUILTIN,LOW);
5  }
6
7  void loop() {
8    if (digitalRead(10)==HIGH){
9      digitalWrite(LED_BUILTIN,LOW);
10   }
11   else{
12     digitalWrite(LED_BUILTIN,HIGH);
13   }
14   delay(100);
15 }
```

simao@ufpr.br - 2025

Código

```
1  void setup() {
2    pinMode(10,INPUT);
3    pinMode(LED_BUILTIN,OUTPUT);
4    digitalWrite(LED_BUILTIN,LOW);
5  }
6
7  void loop() {
8    if (digitalRead(10)==HIGH){
9      digitalWrite(LED_BUILTIN,LOW);
10   }
11   else{
12     digitalWrite(LED_BUILTIN,HIGH);
13   }
14   delay(100);
15 }
```

Na linha 2, `pinMode` liga o pino 10 como sendo uma entrada (as opções são `OUTPUT` para saída, `INPUT` para entrada e `INPUT_PULLUP` para entrada com o resistor colocado internamente ao Arduino).

simao@ufpr.br - 2025

Circuito elétrico

```
1  void setup() {
2    pinMode(10,INPUT);
3    pinMode(LED_BUILTIN,OUTPUT);
4    digitalWrite(LED_BUILTIN,LOW);
5  }
6
7  void loop() {
8    if (digitalRead(10)==HIGH){
9      digitalWrite(LED_BUILTIN,LOW);
10   }
11   else{
12     digitalWrite(LED_BUILTIN,HIGH);
13   }
14   delay(100);
15 }
```

Na linha 3 ligamos o LED da placa (identificado como `LED_BUILTIN`, conectado ao pino 13) como sendo uma saída (`OUTPUT`).

Na linha 4 garantimos que o LED da placa seja inicie desligado, por meio da instrução `digitalWrite` enviadno um nível baixo (`LOW`).

Circuito elétrico

```
1  void setup() {
2    pinMode(10,INPUT);
3    pinMode(LED_BUILTIN,OUTPUT);
4    digitalWrite(LED_BUILTIN,LOW);
5  }
6
7  void loop() {
8    if (digitalRead(10)==HIGH){
9      digitalWrite(LED_BUILTIN,LOW);
10   }
11   else{
12     digitalWrite(LED_BUILTIN,HIGH);
13   }
14   delay(100);
15 }
```

Na linha 8 há um teste lógico. O comando `if` executará a comparação que está entre parêntes, e, se for verdadeira, irá executar o bloco de comandos contido entre '`{`' e '`}`' que vem a seguir; se for falsa executa o bloco de comandos contido pelas chaves da instrução `else`.

Note que o comparador 'igual' nesta linguagem é representado por '==' !

simao@ufpr.br - 2025

Circuito elétrico

```
1  void setup() {
2    pinMode(10,INPUT);
3    pinMode(LED_BUILTIN,OUTPUT);
4    digitalWrite(LED_BUILTIN,LOW);
5  }
6
7  void loop() {
8    if (digitalRead(10)==HIGH){
9      digitalWrite(LED_BUILTIN,LOW);
10   }
11   else{
12     digitalWrite(LED_BUILTIN,HIGH);
13   }
14   delay(100);
15 }
```

SE o teste da linha 8 for verdadeiro, será executada a instrução da linha 9. Nesta linha será utilizado o método `digitalWrite` para enviar ao LED da placa um nível baixo (`LOW`, que corresponde ao interruptor não pressionado em nossa montagem). SE o teste for falso será executado o bloco eles, e enviado ao LED um sinal `HIGH` (por meio do comando da linha 12).
Na linha 14 o `delay` apenas acrescenta uma espera de 100ms.

Teste!

simao@ufpr.br - 2025

Outro código

```
1  void setup() {
2    pinMode(10,INPUT);
3    Serial.begin(9600);
4  }
5
6  void loop() {
7    if (digitalRead(10)==HIGH){
8      Serial.println("Interruptor nao pressionado");
9    }
10   else{
11     Serial.println("Interruptor pressionado");
12   }
13   delay(1500);
14 }
```

Outro código

```
1  void setup() {
2    pinMode(10,INPUT);
3    Serial.begin(9600);
4  }
5
6  void loop() {
7    if (digitalRead(10)==HIGH){
8      Serial.println("Interruptor nao pressionado");
9    }
10   else{
11     Serial.println("Interruptor pressionado");
12   }
13   delay(1500);
14 }
```

Na linha 2, `pinMode` liga o pino 10 como sendo uma entrada (as opções são `OUTPUT` para saída, `INPUT` para entrada e `INPUT_PULLUP` para entrada com o resistor colocado internamente ao Arduino).

simao@ufpr.br - 2025

Outro código

```
1  void setup() {
2    pinMode(10,INPUT);
3    Serial.begin(9600);
4  }
5
6  void loop() {
7    if (digitalRead(10)==HIGH){
8      Serial.println("Interruptor nao pressionado");
9    }
10   else{
11     Serial.println("Interruptor pressionado");
12   }
13   delay(1500);
14 }
```

Na linha 3 iniciamos o objeto de comunicação serial com velocidade de comunicação de 9600 bps, por meio da instrução `Serial.begin(9600)`.

simao@ufpr.br - 2025

Outro código

```
1  void setup() {
2    pinMode(10,INPUT);
3    Serial.begin(9600);
4  }
5
6  void loop() {
7    if (digitalRead(10)==HIGH){
8      Serial.println("Interruptor nao pressionado");
9    }
10   else{
11     Serial.println("Interruptor pressionado");
12   }
13   delay(1500);
14 }
```

Na linha 7 há um teste lógico. O comando `if` executará a comparação que está entre parêntes, e, se for verdadeira, irá executar o bloco de comandos contido entre '`{`' e '`}`' que vem a seguir; se for falsa executa o bloco de comandos contido pelas chaves da instrução `else`.

Note que o comparador 'igual' nesta linguagem é representado por '==' !

simao@ufpr.br - 2025

Outro código

```
1  void setup() {
2    pinMode(10,INPUT);
3    Serial.begin(9600);
4  }
5
6  void loop() {
7    if (digitalRead(10)==HIGH){
8      Serial.println("Interruptor nao pressionado");
9    }
10   else{
11     Serial.println("Interruptor pressionado");
12   }
13   delay(1500);
14 }
```

SE o teste da linha 7 for verdadeiro, será executada a instrução da linha 8. Nesta linha será utilizado o método `println` do objeto `Serial`, que efetua a escrita de um valor passado como parâmetro (entre aspas) e pula uma linha (o '`ln`' ao final do '`print`' significa 'line', pule uma linha após realizar a escrita). O `delay` da linha 13 é somente para que as mensagens fiquem menos rápidas.

simao@ufpr.br - 2025

Monitor serial

Ligue o monitor serial e teste seu código.

(para exibir o monitor serial clique em Ferramentas e selecione Monitor Serial – outros ambientes tem interfaces diferentes)

simao@ufpr.br - 2025

Teste !

simao@ufpr.br - 2025

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

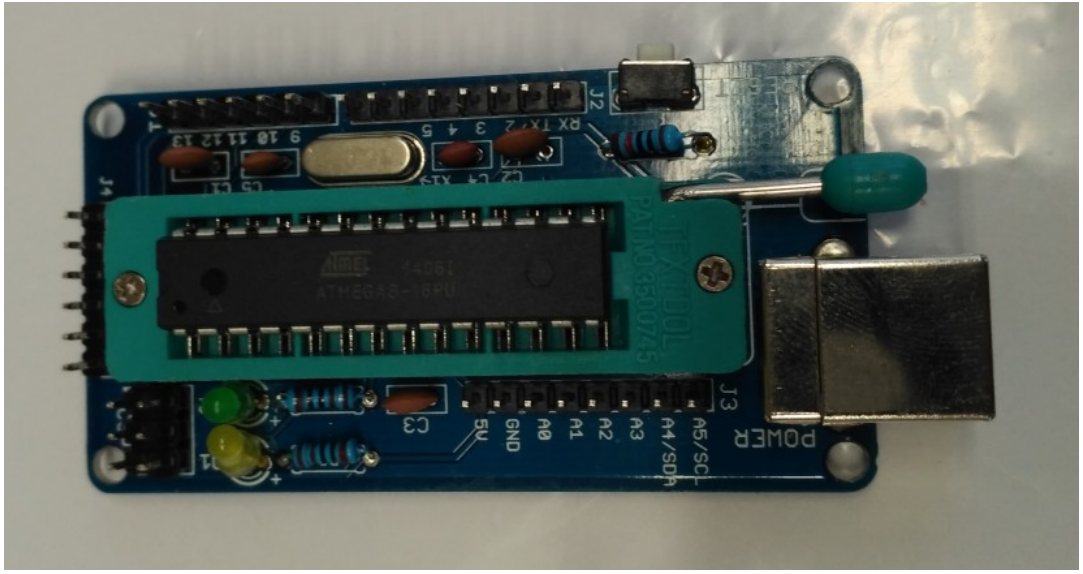
Caso contrário, vamos em frente...

simao@ufpr.br - 2025

Terminou seu protótipo?

simao@ufpr.br - 2025

Gravação do código

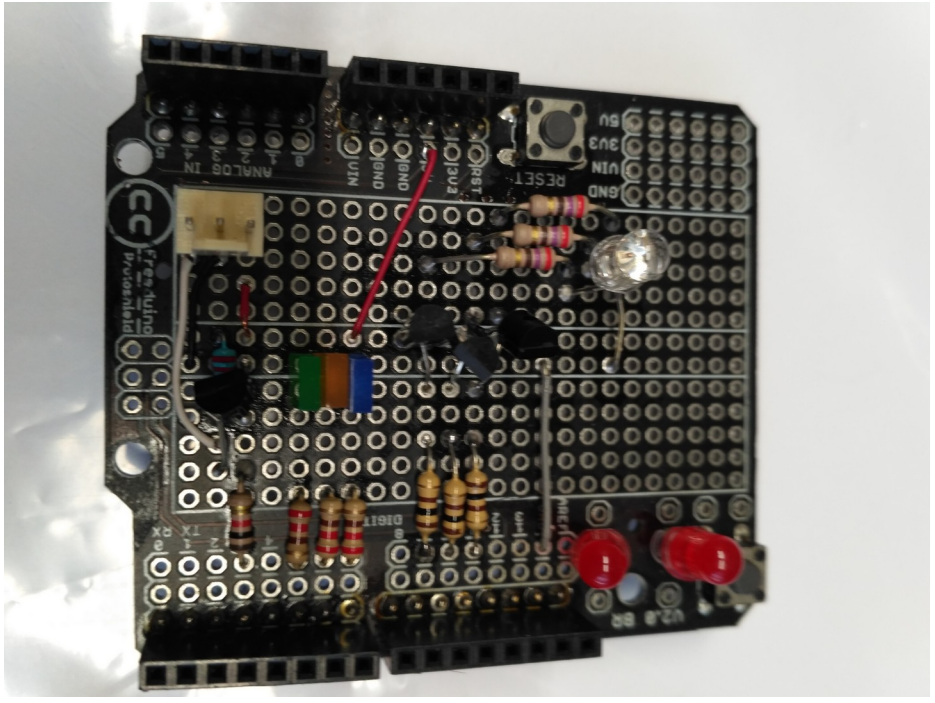


Pode gravá-lo em um microcontrolador compatível, que será levado para sua placa definitiva, ou seu MVP!

Imagem: foto do autor

simao@ufpr.br - 2025

Componentes



Também pode produzir uma versão em placas com solda, de forma que não ficará com fios soltos e será mais prático de demonstrar.

Imagem: foto do autor

simao@ufpr.br - 2025

O que você aprendeu, que será necessário para os próximos passos:

1. Ligar componentes externos à placa do Arduino
2. Utilizar uma saída digital com escrita digital
3. Utilizar uma saída digital com escrita analógica
4. Ler um valor em porta digital
5. Escrever algo no monitor serial

simao@ufpr.br - 2025

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

Parabéns!

simao@ufpr.br - 2025