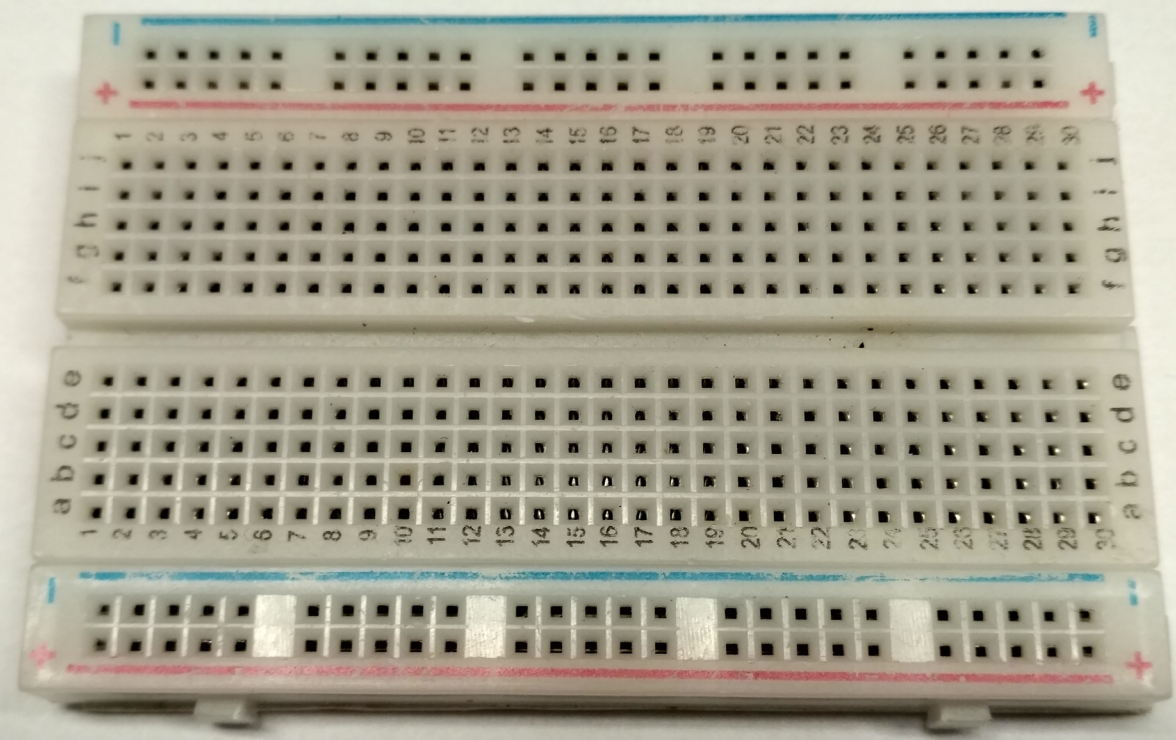


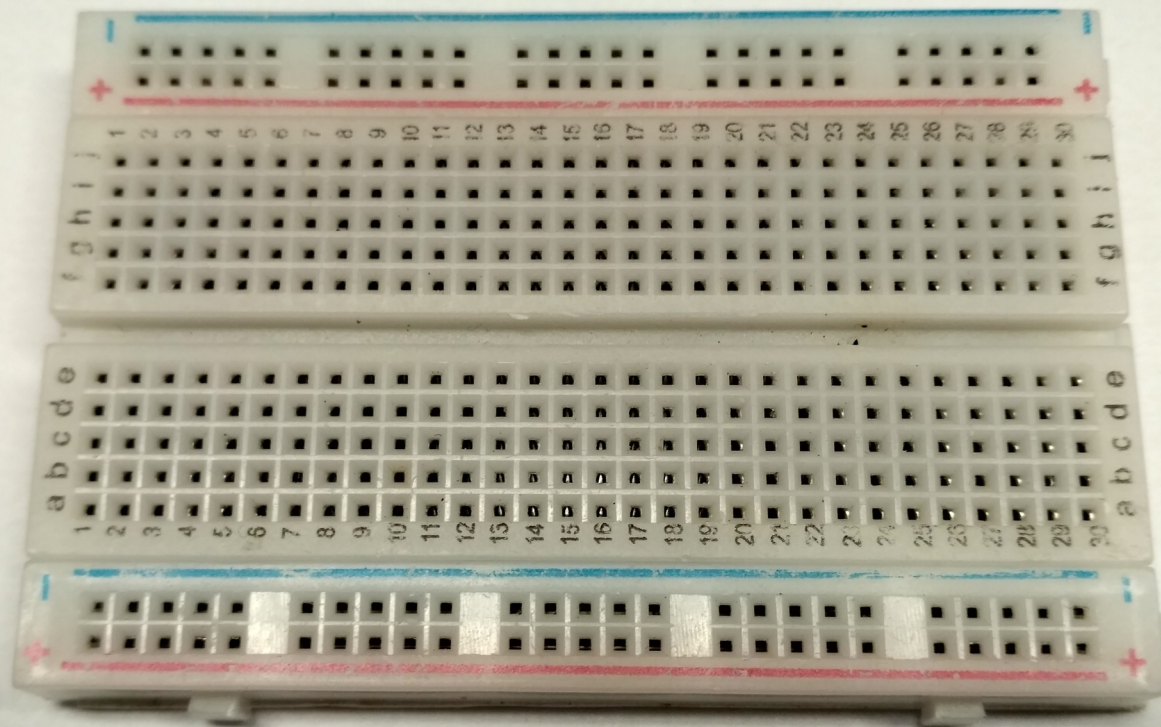
# Arduino



# Breadboard, protoboard



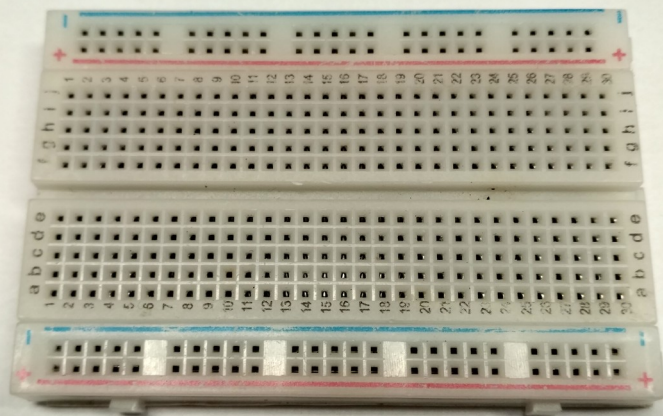
Organização matricial, identificável por linhas alfabéticas e colunas numéricas para facilitar a disposição de componentes.



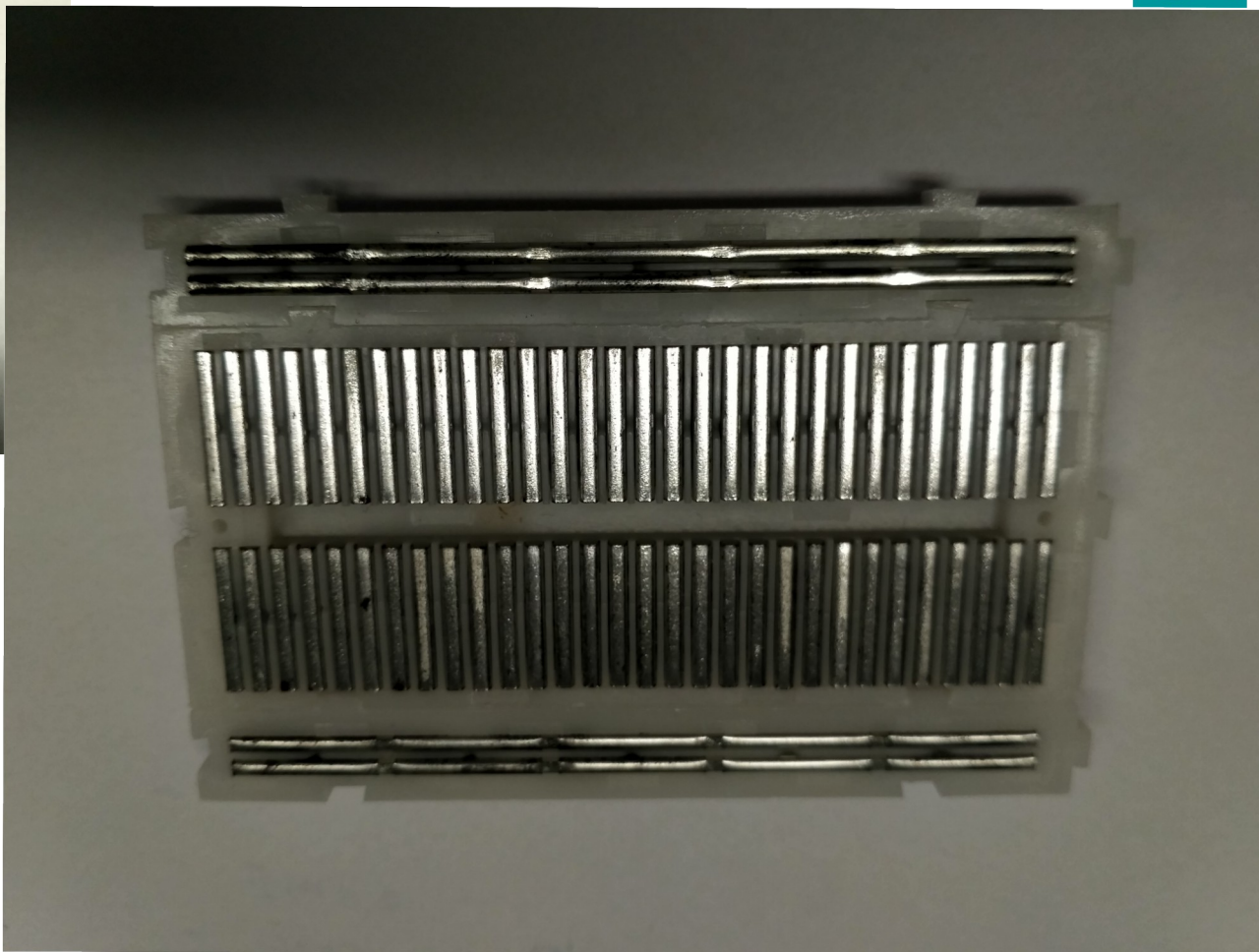
As barras horizontais identificando terminais negativo (-) e positivo (+) são sugestões. Não são polarizadas, servem para organização



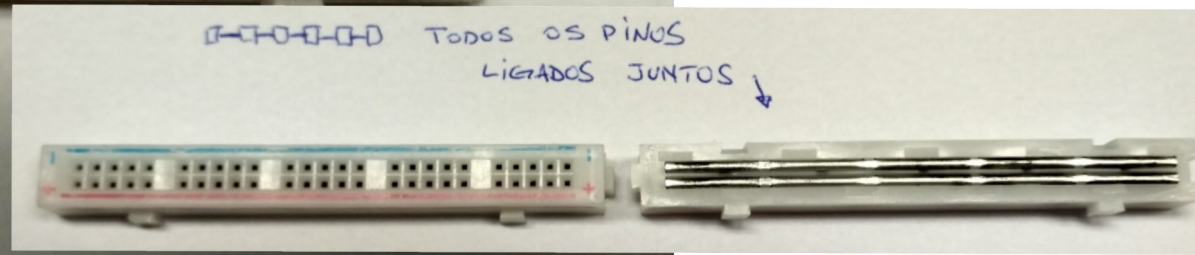
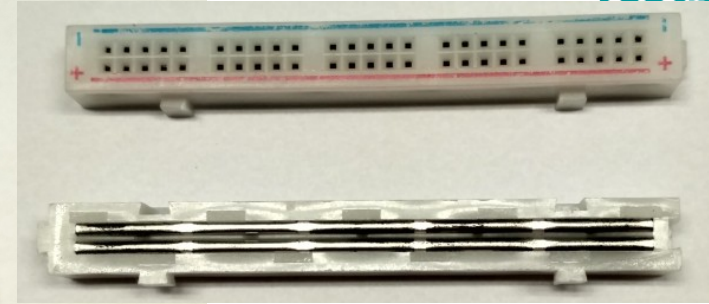
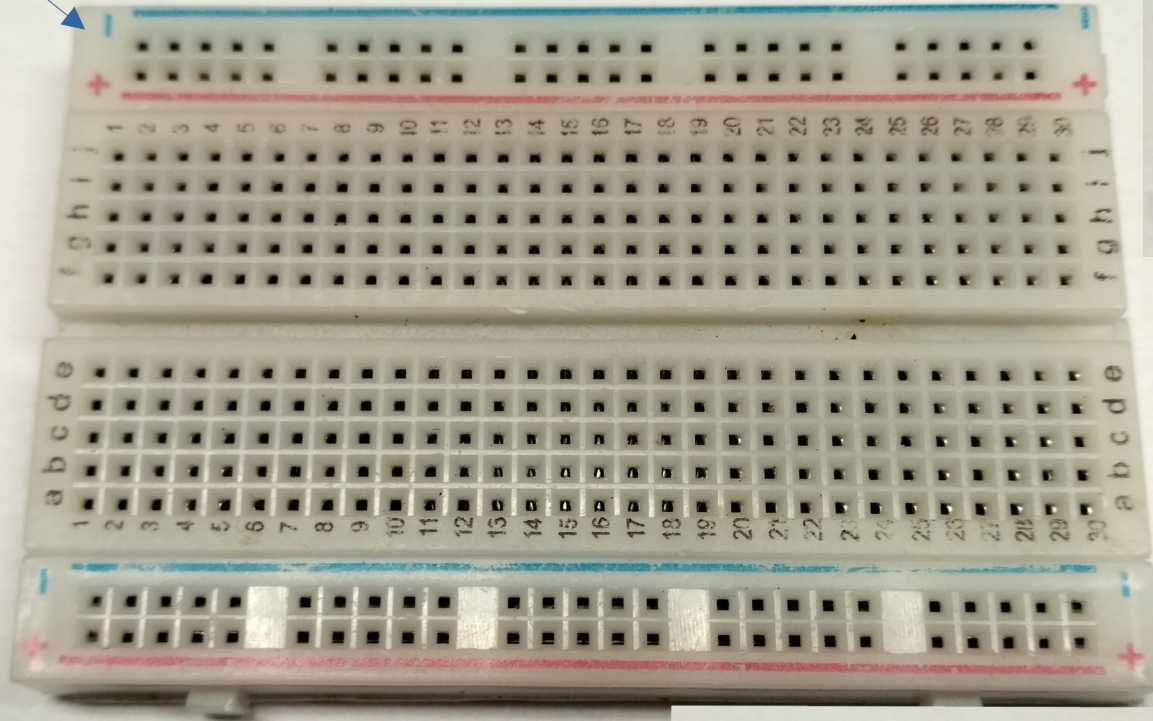
# Visão externa



# Visão interna – conexões elétricas

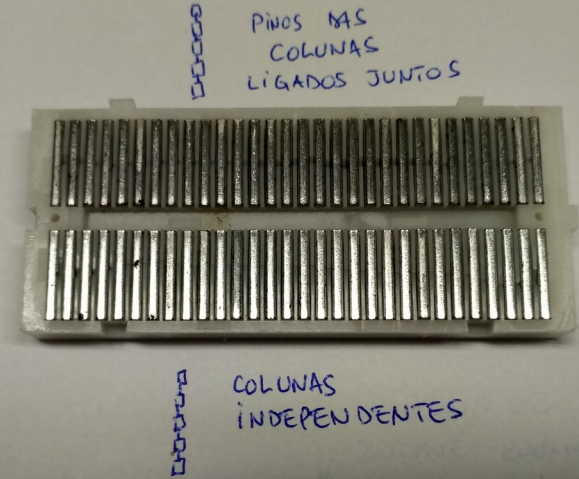
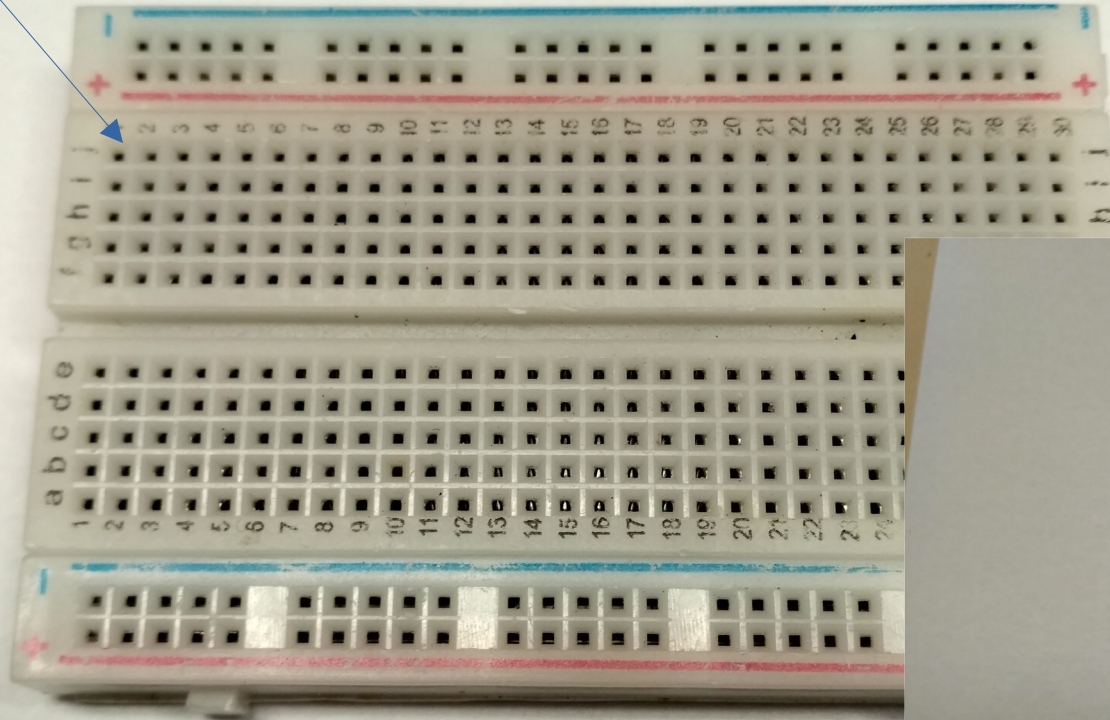


Barras de conexão horizontal de distribuição de energia:  
 todos os pinos da mesma barra estão interconectados





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

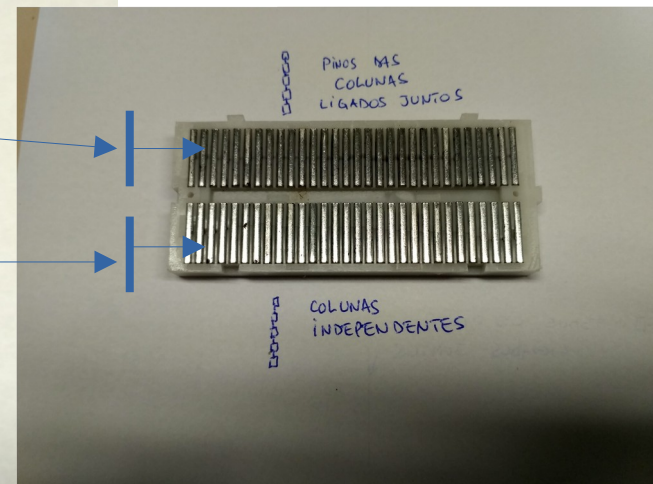
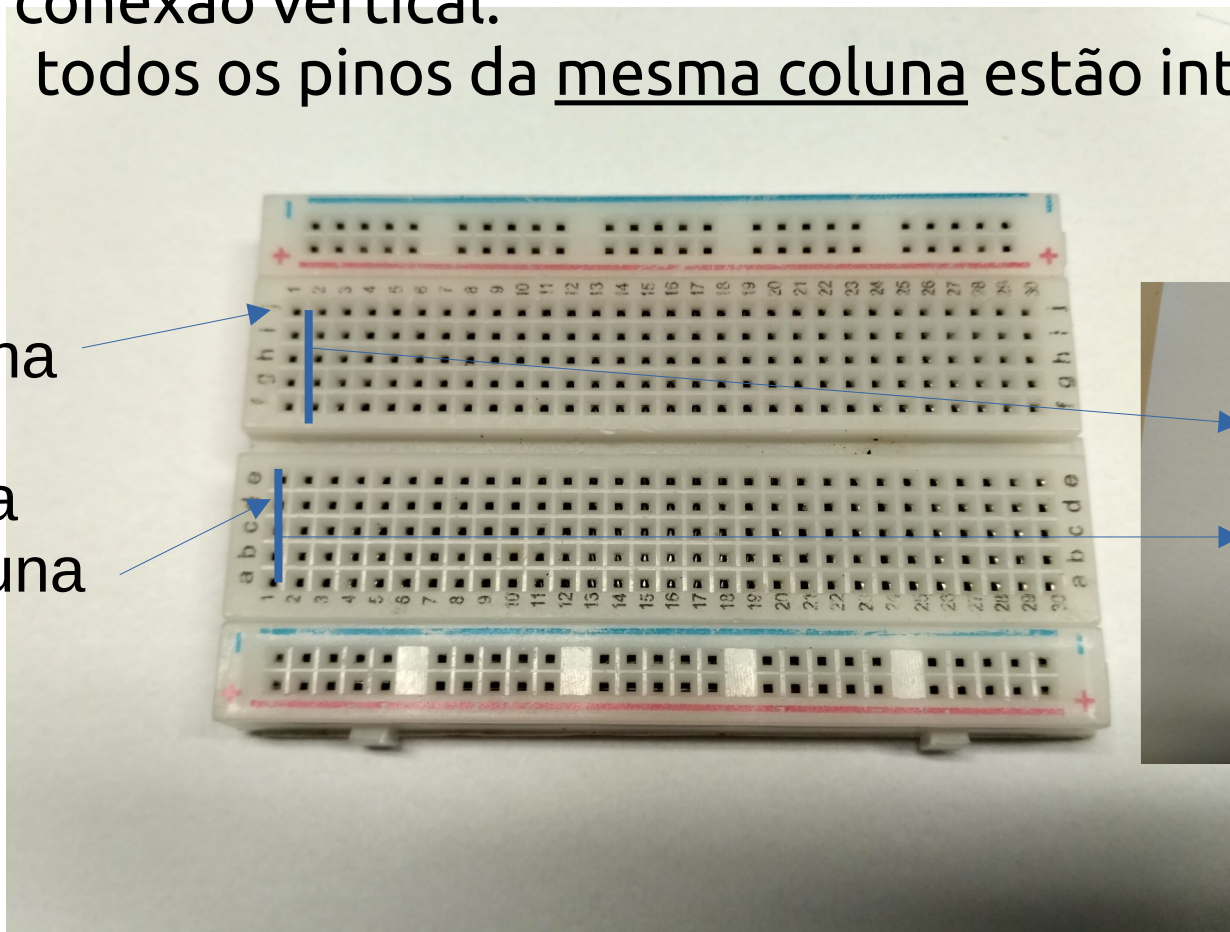


# Colunas de conexão vertical:

todos os pinos da mesma coluna estão interconectados



Esta coluna  
não está  
conectada  
nesta coluna

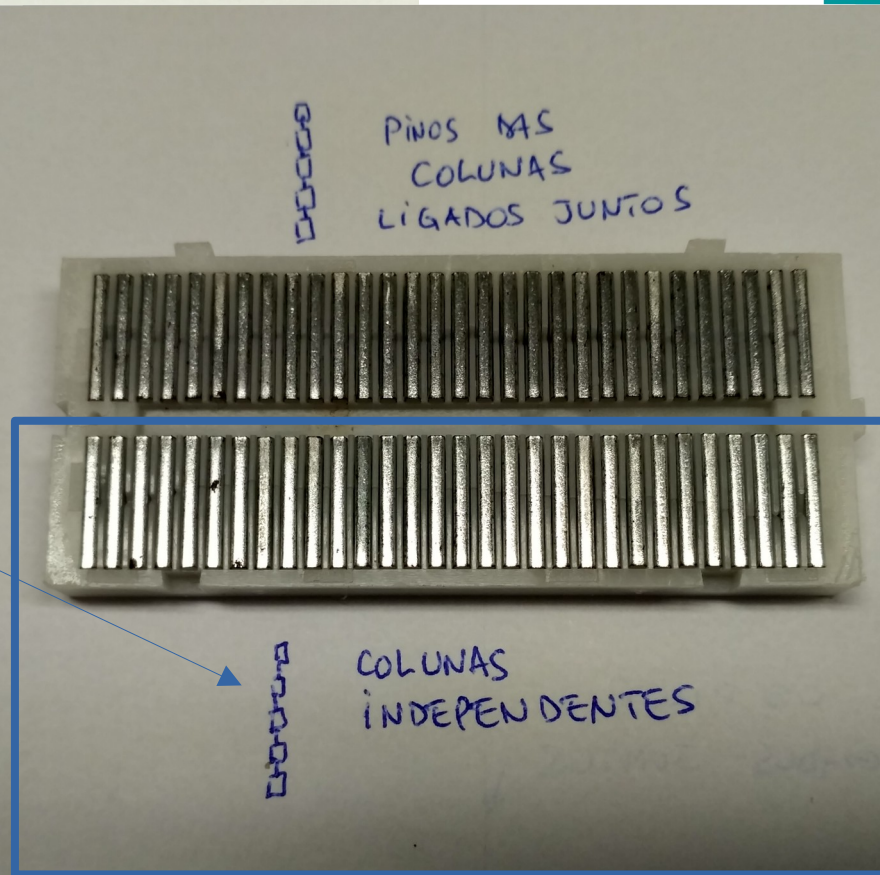
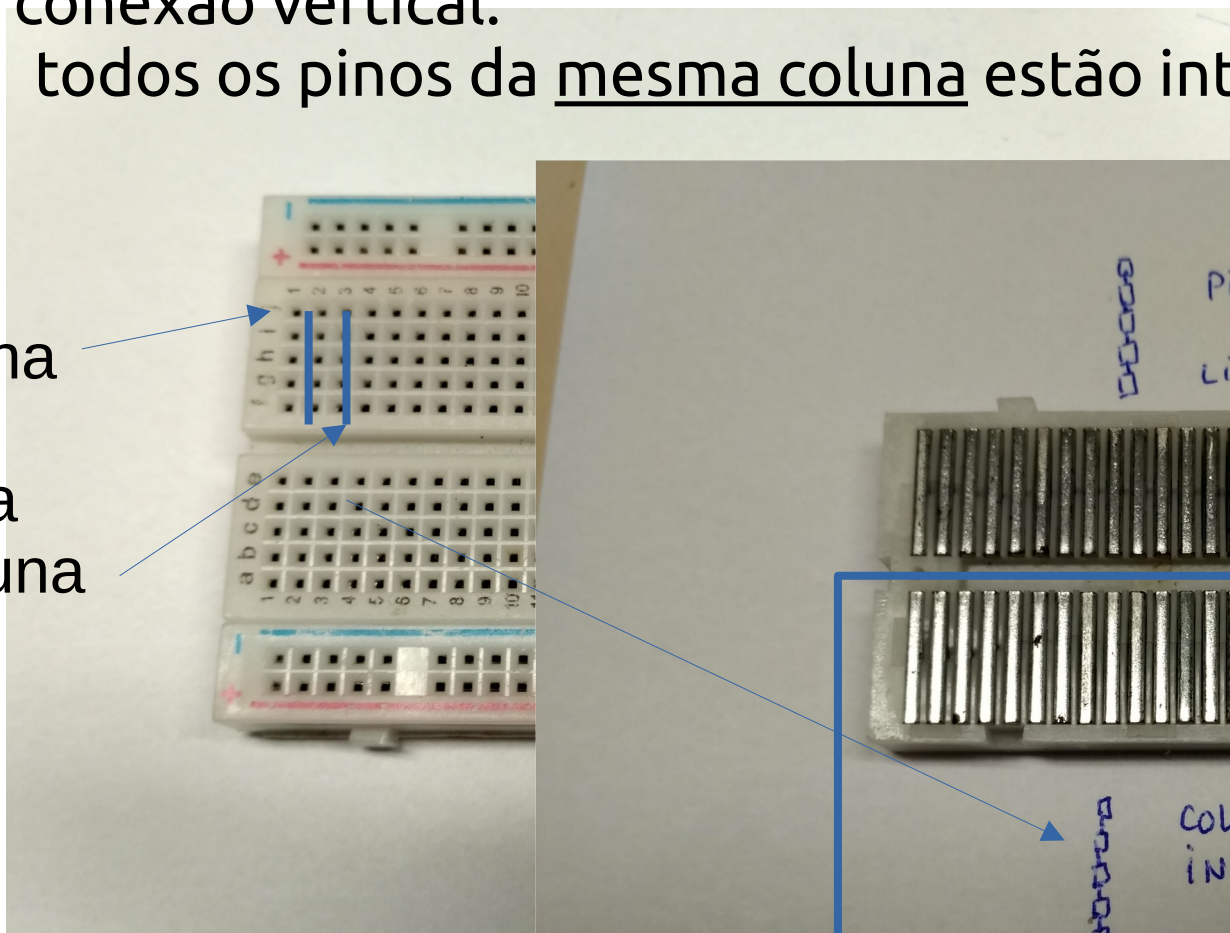




# Colunas de conexão vertical:

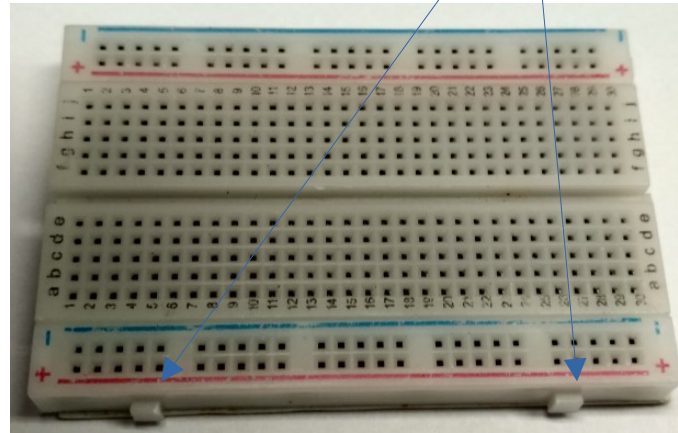
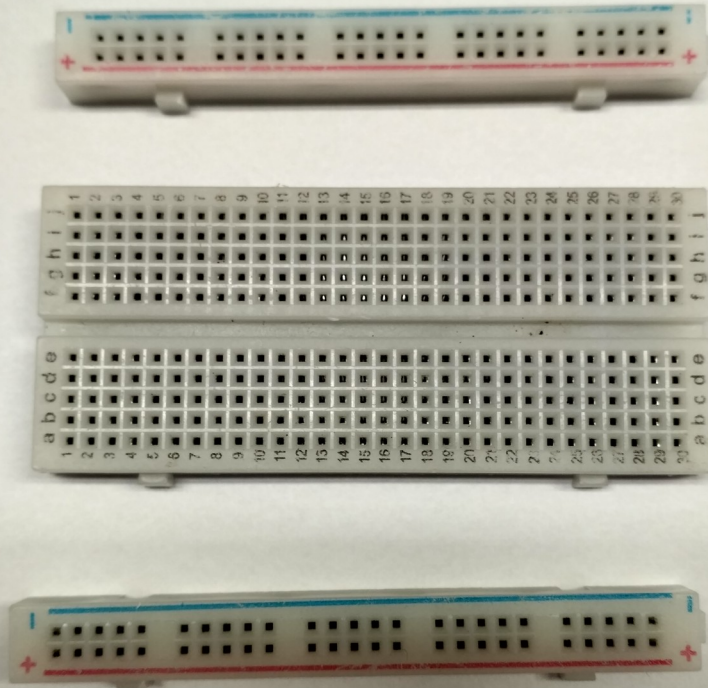
todos os pinos da mesma coluna estão interconectados

Esta coluna  
não está  
conectada  
nesta coluna

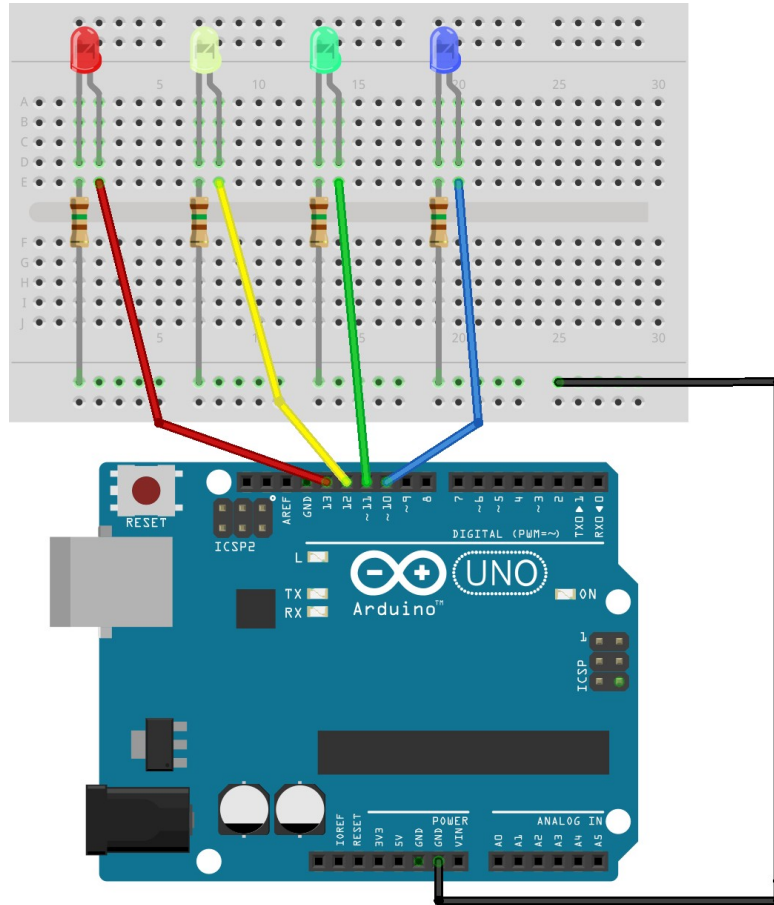




Algumas placas podem ser desmontadas quanto às barras e colunas e todas podem ser associadas para criar placas maiores por meio de encaixes



# 4 leds, 4 resistores



fritzing

```

1 #define ledAmarelo 12
2 #define ledVermelho 13
3 #define ledVerde 11
4 #define ledAzul 10
5
6 void setup(){
7   Serial.begin(9600);
8   pinMode(ledAmarelo, OUTPUT);
9   pinMode(ledVermelho, OUTPUT);
10  pinMode(ledVerde, OUTPUT);
11  pinMode(ledAzul, OUTPUT);
12 }
13
14
15 void loop(){
16   if (Serial.available())
17   {
18     switch(Serial.read())
19     {
20       case 't':
21         digitalWrite(ledAmarelo, 0);
22         digitalWrite(ledVermelho, 0);
23         digitalWrite(ledVerde, 0);
24         digitalWrite(ledAzul, 0);
25         Serial.println("Todos apagados");
26         break;
27       case 'T':
28         digitalWrite(ledAmarelo, 1);
29         digitalWrite(ledVermelho, 1);
30         digitalWrite(ledVerde, 1);
31         digitalWrite(ledAzul, 1);
32         Serial.println("Todos acesos");
33         break;
34     }
35   }
36 }
    
```

```
void loop(){  
  if (Serial.available())  
  {  
    switch(Serial.read())  
    {  
      case 't':  
        digitalWrite(ledAmarelo, 0);  
        digitalWrite(ledVermelho, 0);  
        digitalWrite(ledVerde, 0);  
        digitalWrite(ledAzul, 0);  
        Serial.println("Todos apagados");  
        break;  
      case 'T':  
        digitalWrite(ledAmarelo, 1);  
        digitalWrite(ledVermelho, 1);  
        digitalWrite(ledVerde, 1);  
        digitalWrite(ledAzul, 1);  
        Serial.println("Todos acesos");  
        break;  
    }  
  }  
}
```

Use o monitor serial para interagir com seu circuito (os comandos poderiam também vir via *Bluetooth* ou outra conexão).



```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
      case 't':
        digitalWrite(ledAmarelo, 0);
        digitalWrite(ledVermelho, 0);
        digitalWrite(ledVerde, 0);
        digitalWrite(ledAzul, 0);
        Serial.println("Todos apagados");
        break;
      case 'T':
        digitalWrite(ledAmarelo, 1);
        digitalWrite(ledVermelho, 1);
        digitalWrite(ledVerde, 1);
        digitalWrite(ledAzul, 1);
        Serial.println("Todos acesos");
        break;
    }
  }
}
```

Se houver dados na interface serial

```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
      case 't':
        digitalWrite(ledAmarelo, 0);
        digitalWrite(ledVermelho, 0);
        digitalWrite(ledVerde, 0);
        digitalWrite(ledAzul, 0);
        Serial.println("Todos apagados");
        break;
      case 'T':
        digitalWrite(ledAmarelo, 1);
        digitalWrite(ledVermelho, 1);
        digitalWrite(ledVerde, 1);
        digitalWrite(ledAzul, 1);
        Serial.println("Todos acesos");
        break;
    }
  }
}
```

Leia os dados da interface serial

```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
      case 't':
        digitalWrite(ledAmarelo, 0);
        digitalWrite(ledVermelho, 0);
        digitalWrite(ledVerde, 0);
        digitalWrite(ledAzul, 0);
        Serial.println("Todos apagados");
        break;
      case 'T':
        digitalWrite(ledAmarelo, 1);
        digitalWrite(ledVermelho, 1);
        digitalWrite(ledVerde, 1);
        digitalWrite(ledAzul, 1);
        Serial.println("Todos acesos");
        break;
    }
  }
}
```

Em função dos dados lidos, escolha



```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
      case 't':
        digitalWrite(ledAmarelo, 0);
        digitalWrite(ledVermelho, 0);
        digitalWrite(ledVerde, 0);
        digitalWrite(ledAzul, 0);
        Serial.println("Todos apagados");
        break;
      case 'T':
        digitalWrite(ledAmarelo, 1);
        digitalWrite(ledVermelho, 1);
        digitalWrite(ledVerde, 1);
        digitalWrite(ledAzul, 1);
        Serial.println("Todos acesos");
        break;
    }
  }
}
```

SE for 't' (tê minúsculo)

```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
      case 't':
        digitalWrite(ledAmarelo, 0);
        digitalWrite(ledVermelho, 0);
        digitalWrite(ledVerde, 0);
        digitalWrite(ledAzul, 0);
        Serial.println("Todos apagados");
        break;
      case 'T':
        digitalWrite(ledAmarelo, 1);
        digitalWrite(ledVermelho, 1);
        digitalWrite(ledVerde, 1);
        digitalWrite(ledAzul, 1);
        Serial.println("Todos acesos");
        break;
    }
  }
}
```

Escreva '0' (que equivale a um LOW) nas portas em que estão os LEDs (e, com isso, devido à ligação efetuada, desligue-os/ apague-os).

Escreva no monitor serial que os leds estão todos apagados.

```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
      case 't':
        digitalWrite(ledAmarelo, 0);
        digitalWrite(ledVermelho, 0);
        digitalWrite(ledVerde, 0);
        digitalWrite(ledAzul, 0);
        Serial.println("Todos apagados");
        break;
      case 'T':
        digitalWrite(ledAmarelo, 1);
        digitalWrite(ledVermelho, 1);
        digitalWrite(ledVerde, 1);
        digitalWrite(ledAzul, 1);
        Serial.println("Todos acesos");
        break;
    }
  }
}
```

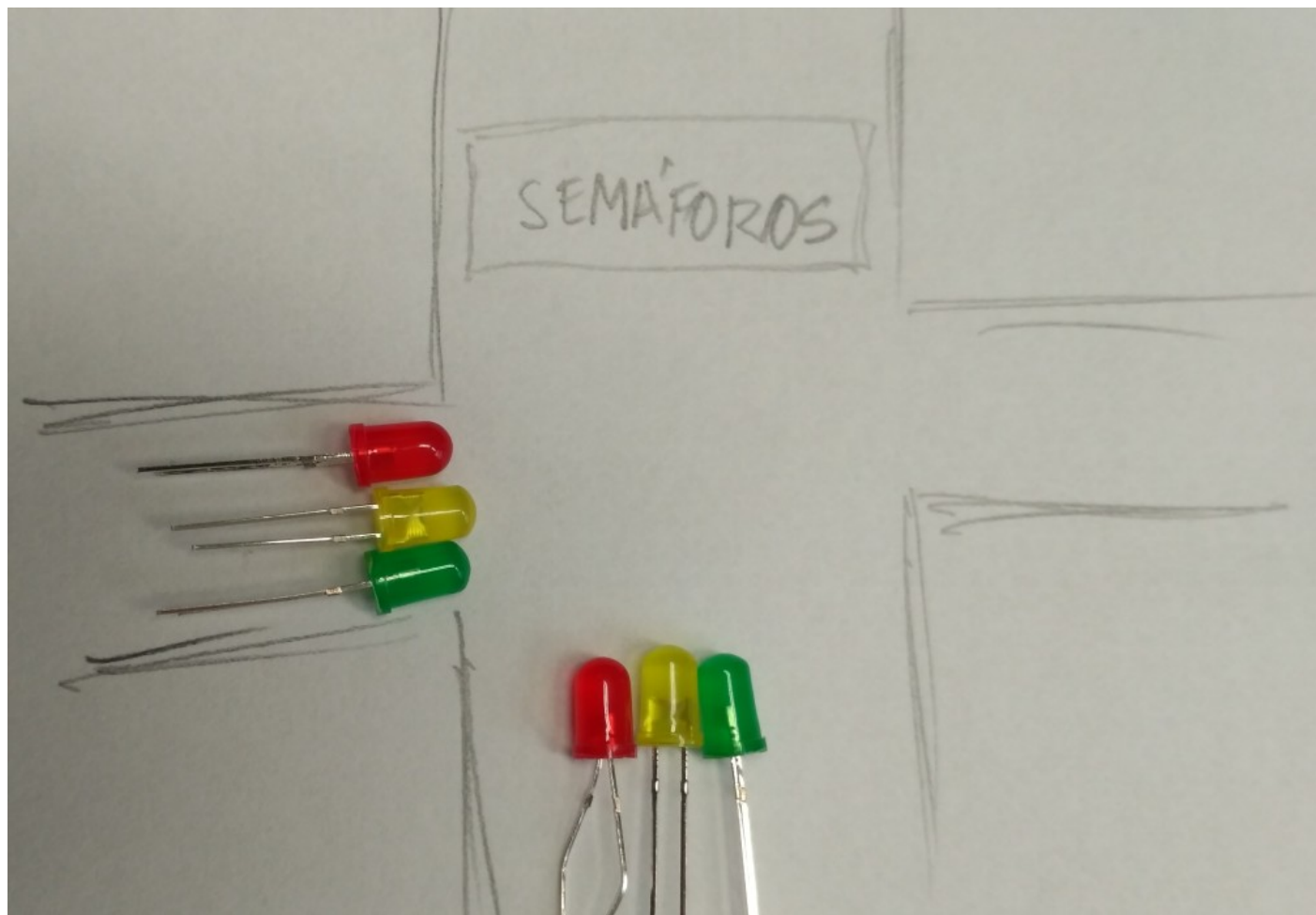
SE for 'T' (tê maiúsculo)



```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
      case 't':
        digitalWrite(ledAmarelo, 0);
        digitalWrite(ledVermelho, 0);
        digitalWrite(ledVerde, 0);
        digitalWrite(ledAzul, 0);
        Serial.println("Todos apagados");
        break;
      case 'T':
        digitalWrite(ledAmarelo, 1);
        digitalWrite(ledVermelho, 1);
        digitalWrite(ledVerde, 1);
        digitalWrite(ledAzul, 1);
        Serial.println("Todos acesos");
        break;
    }
  }
}
```

Escreva '1' (que equivale a um HIGH) nas portas em que estão os LEDs (e, com isso, devido à ligação efetuada, ligue-os/ acenda-os).

Escreva no monitor serial que os leds  
estão todos acesos.



# Semáforo

- Baseado em seus experimentos anteriores, crie um semáforo contendo 6 leds, nas cores usuais de um semáforo.
  - Quando o led vermelho de um estiver aceso o led verde o outro estará aceso
  - Uma variação é colocar um temporizador para que os pedestres atravessem, antes que o segundo sinal abra, após o primeiro fechar

# O que você precisará?

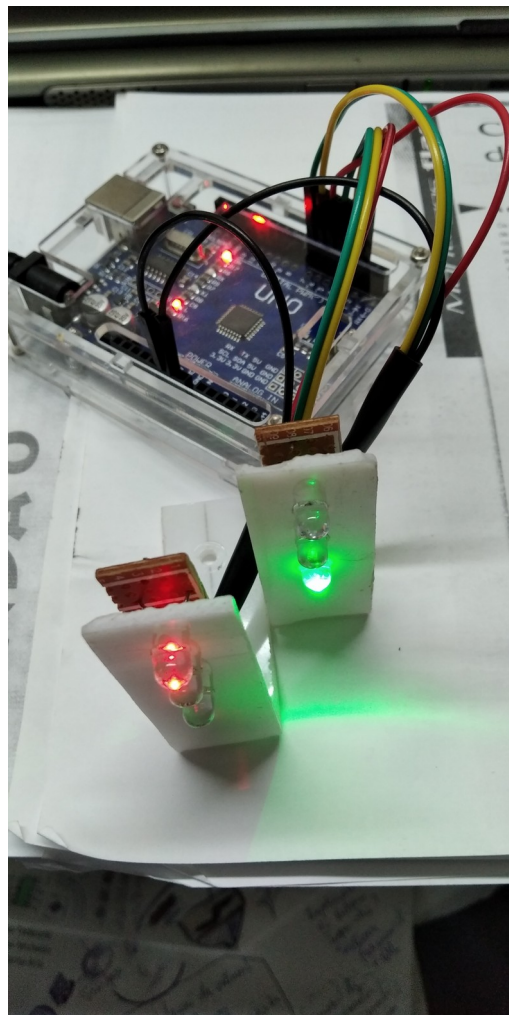
- Arduino configurado na IDE
- Uma placa de experimentação e fios
- 6 leds, sendo 2 vermelhos, 2 verdes e 2 amarelos
- 6 resistores para limitar corrente/ tensão, conforme já visto



# Sequência

- Desligue o cabo USB
- Ligue os componentes na placa
- Conecte os fios à placa do Arduino; anote as portas digitais utilizadas para cada led
- Elabore o código, lembrando de declarar as portas como saída no **setup()** e depois escrevendo níveis LOW ou HIGH em cada porta dentro do **loop()**

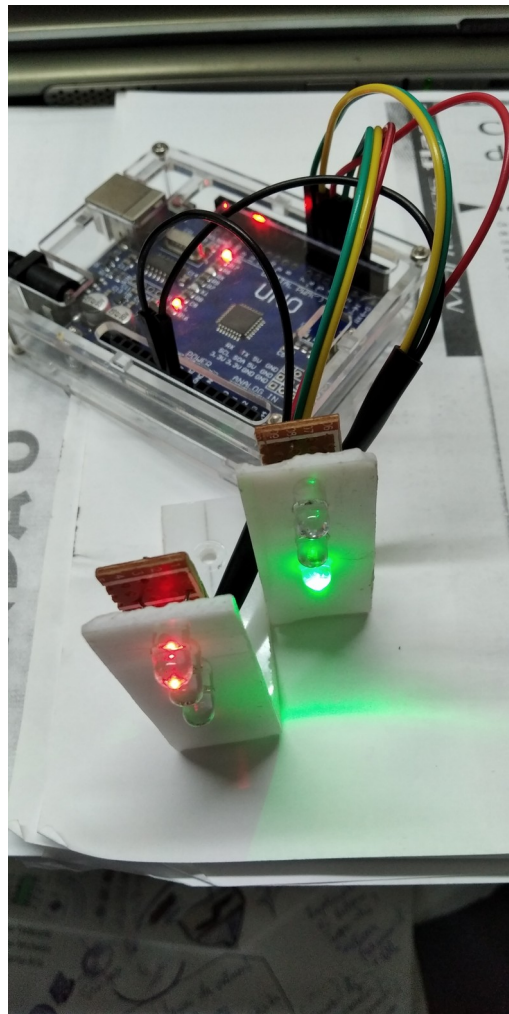
Exemplo – você pode mudar as portas e os nomes das constantes



```

1 //semáforo 1
2 #define vm1 5
3 #define am1 6
4 #define vd1 7
5
6 //semáforo 2
7
8 #define vm2 8
9 #define am2 9
10 #define vd2 10
11
12 void setup() {
13     //liga como saída os pinos para o semáforo 1
14     pinMode(vm1, OUTPUT);
15     pinMode(am1, OUTPUT);
16     pinMode(vd1, OUTPUT);
17     //liga como saída os pinos para o semáforo 2
18     pinMode(vm2, OUTPUT);
19     pinMode(am2, OUTPUT);
20     pinMode(vd2, OUTPUT);
21 }

```



```

23 void loop() {
24     // semáforos 1 e 2 vermelhos, pedestre atravessando
25     digitalWrite(vm1, HIGH);
26     digitalWrite(vm2, HIGH);
27     delay(1000);
28     //abre o semáfor 1, semáforo 2 fechado
29     digitalWrite(vm1, LOW);
30     digitalWrite(vd1, HIGH);
31     delay(1000);
32     //prepara para fechar o semáforo 1
33     digitalWrite(vd1, LOW);
34     digitalWrite(am1, HIGH);
35     delay(1000);
36     digitalWrite(am1, LOW);
37     //fecha o semáforo 1 e abre o 2
38     digitalWrite(vm1, HIGH);
39     digitalWrite(vm2, LOW);
40     digitalWrite(vd2, HIGH);
41     delay(1000);
42     //prepara para fechar o semáforo 2
43     digitalWrite(vd2, LOW);
44     digitalWrite(am2, HIGH);
45     delay(1000);
46     digitalWrite(am2, LOW);
47 }

```

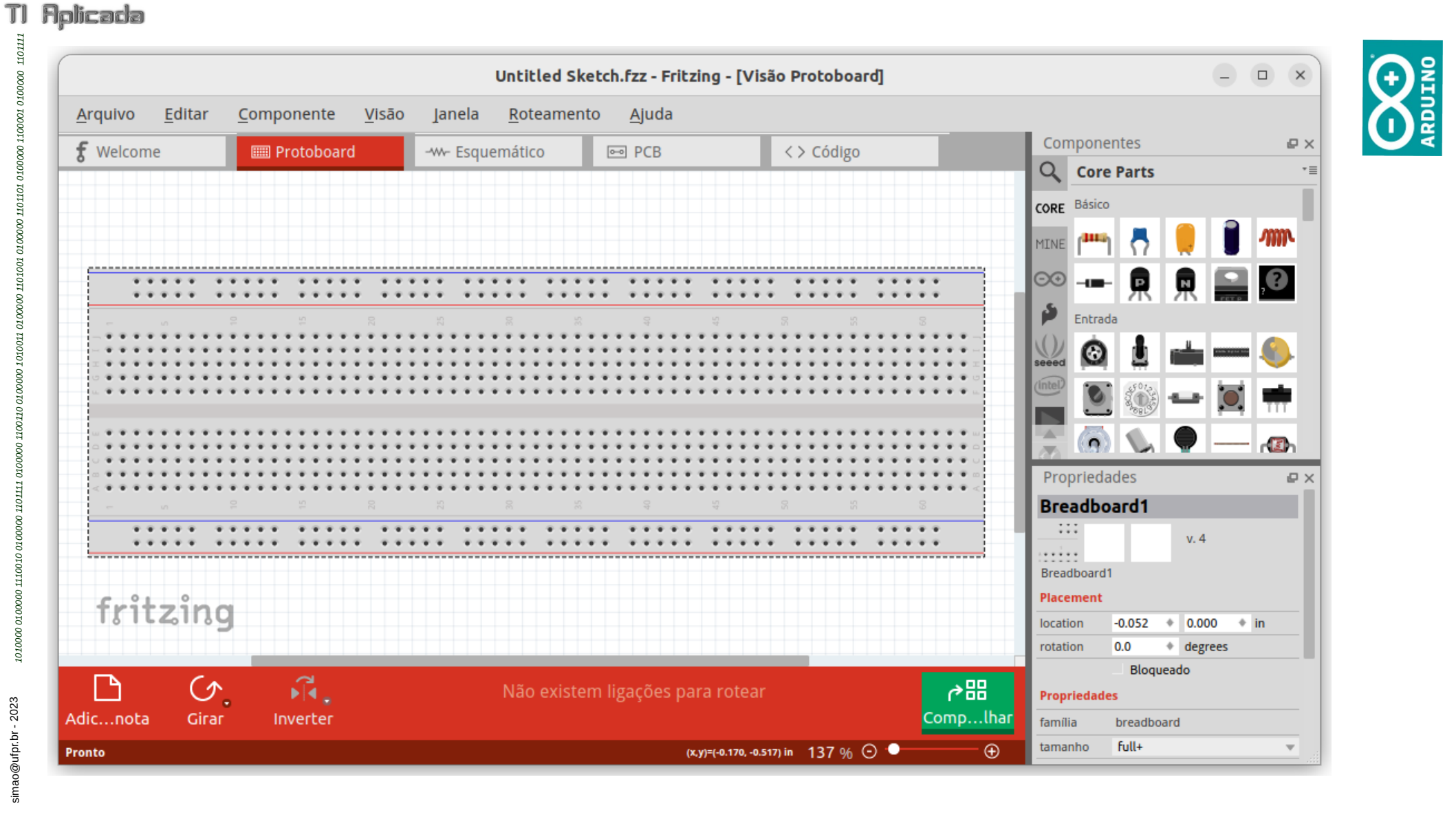


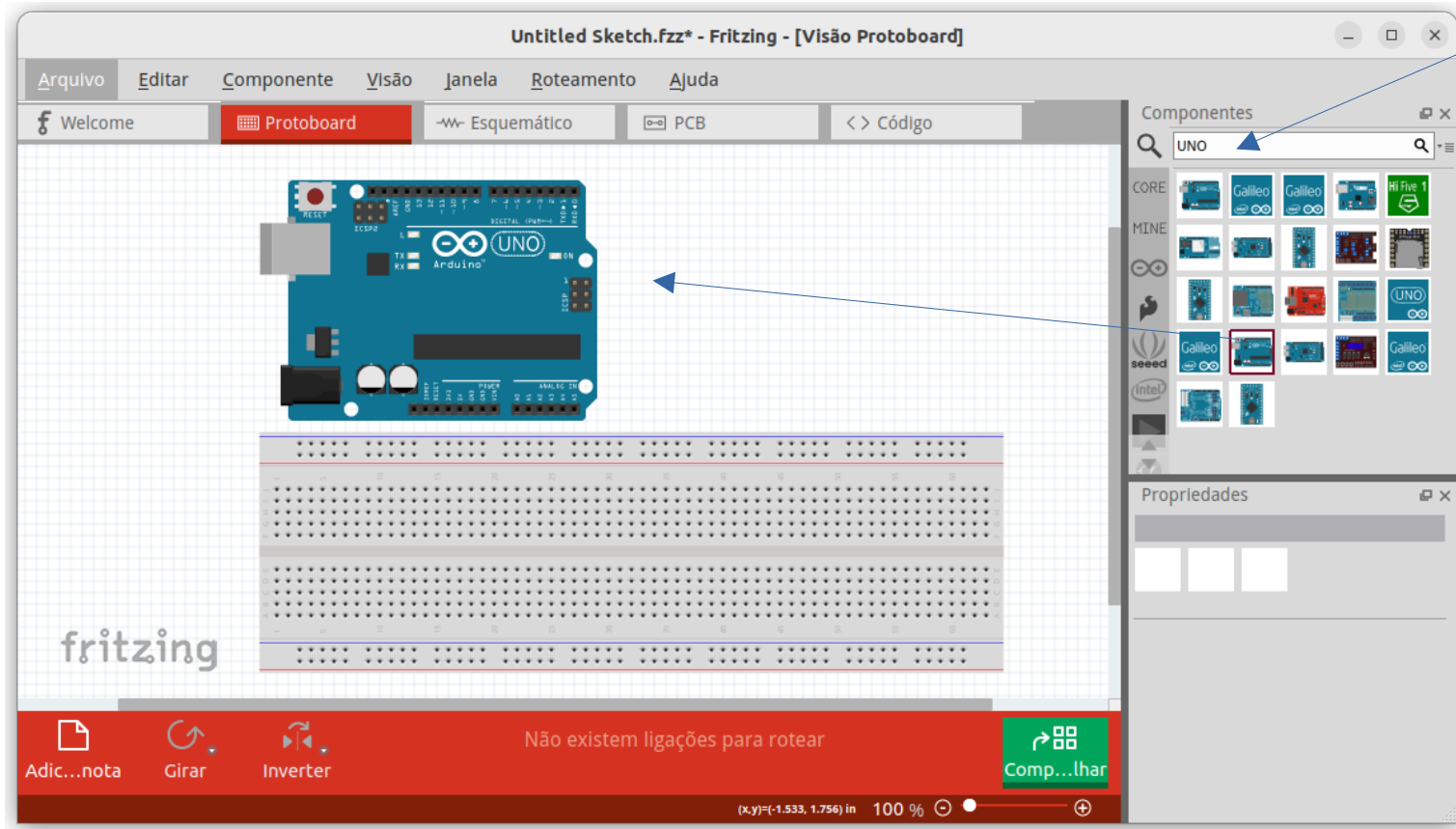
Como ficou sua ligação?

Documente!

Use o Fritzing!

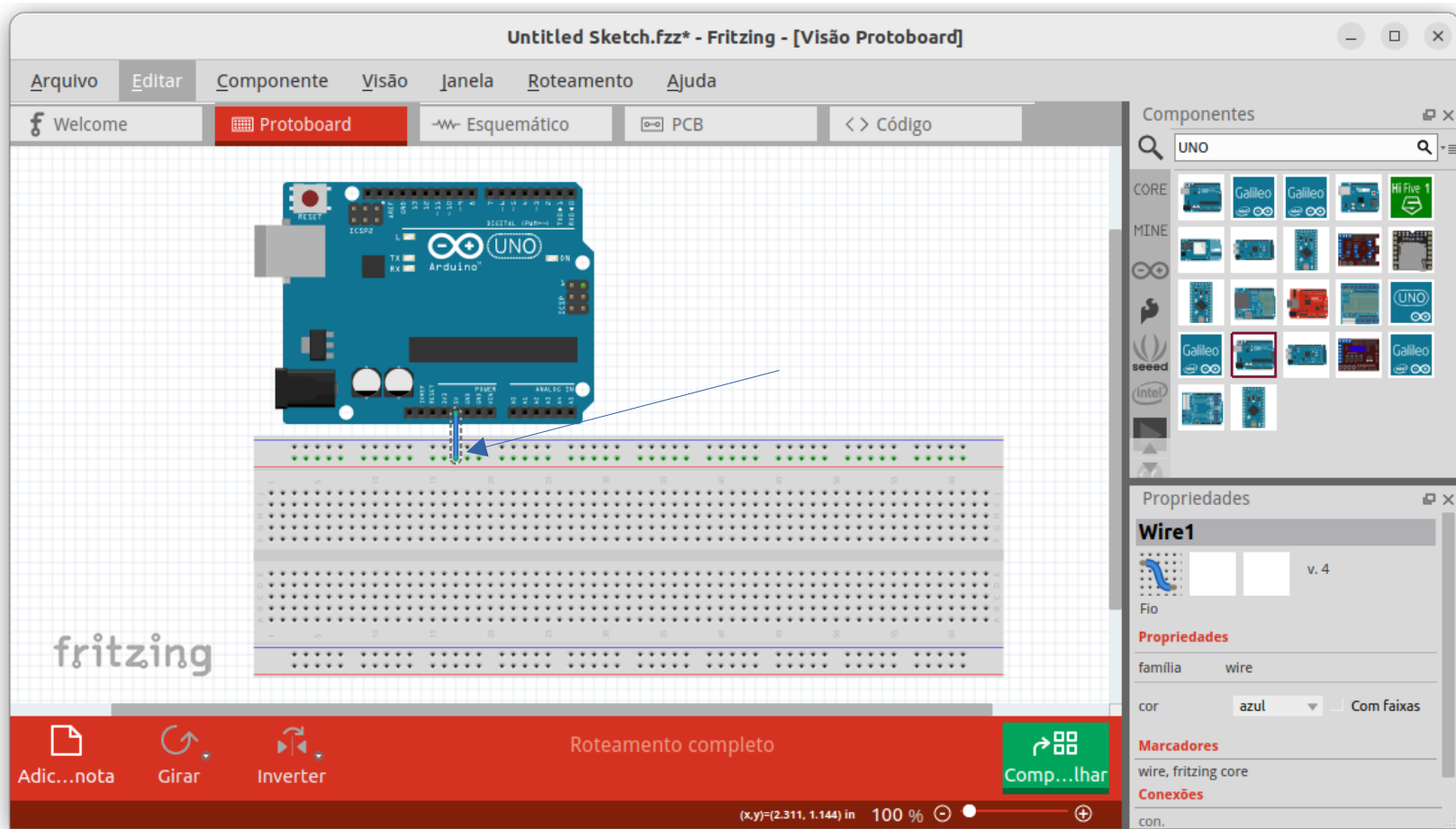






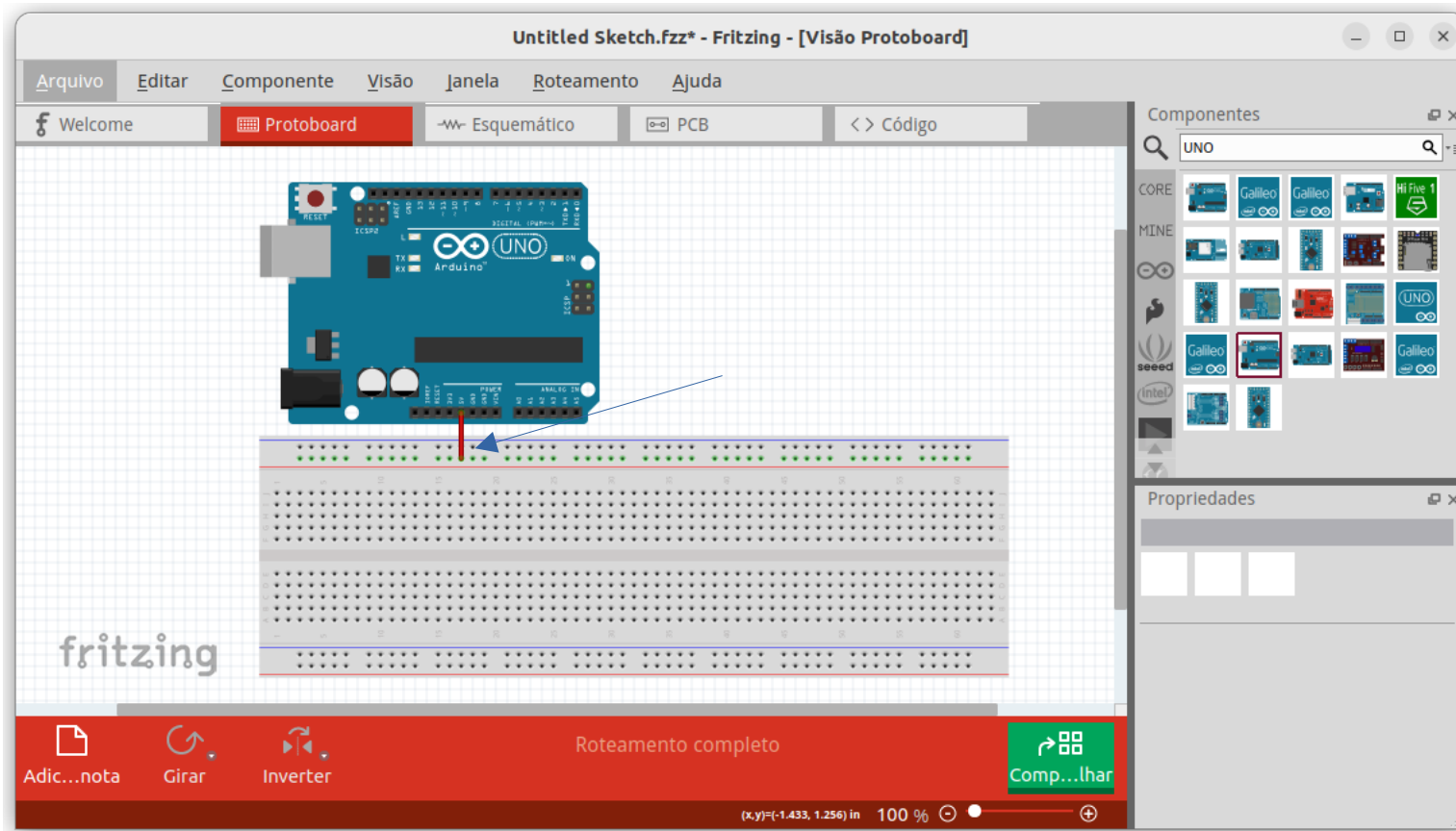


Clique sobre o pino 5V na placa do Arduino e arraste até a barra positiva. O Fritzing vai indicar a ligação com um fio.



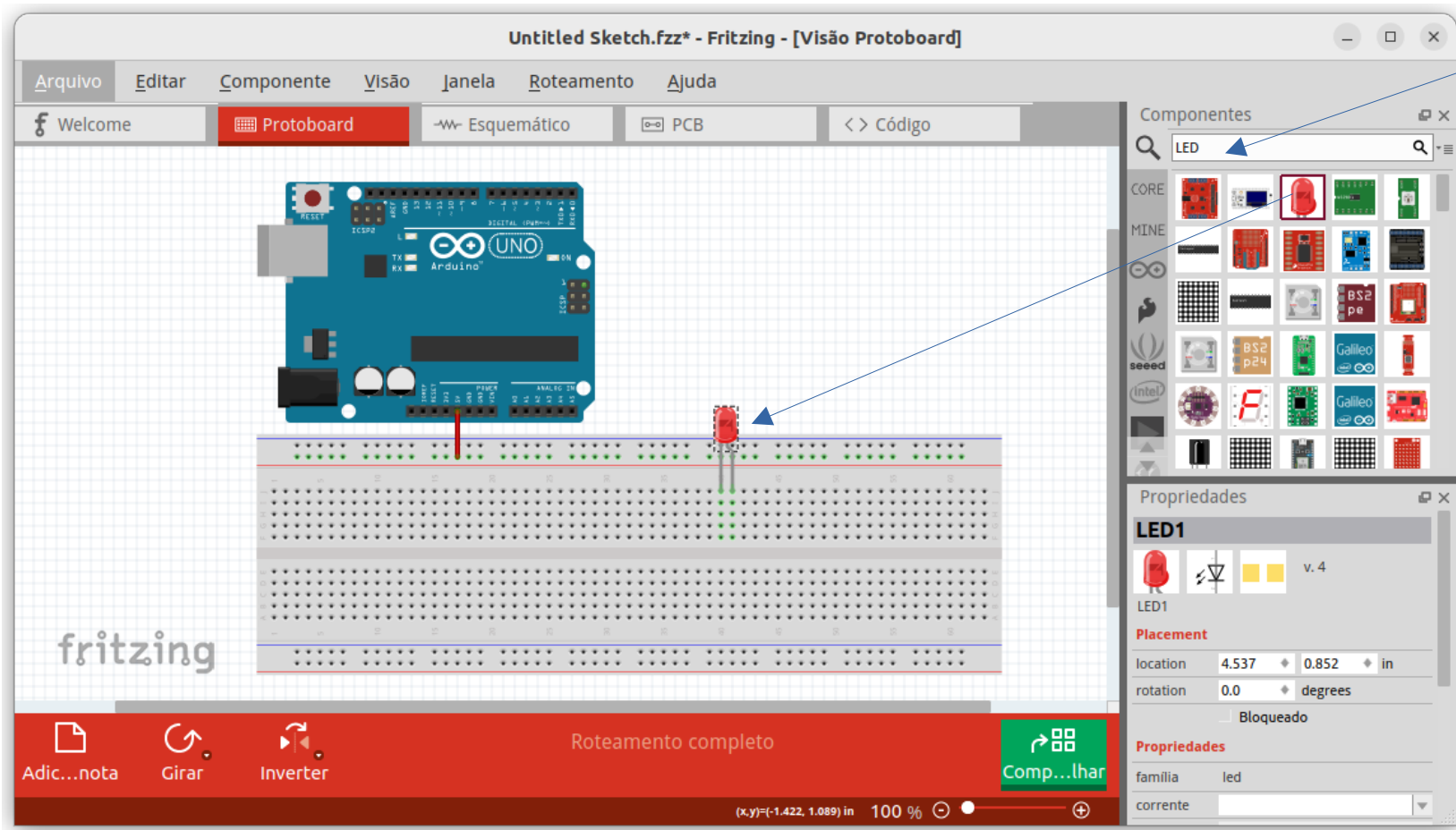
O programa utiliza a última cor utilizada como padrão. Portanto, no seu caso a cor do fio poderá não ser azul como a do exemplo ao lado.

Clique com o botão direito do mouse sobre o fio e selecione 'Cor dos fios'. Troque a cor para vermelho\*



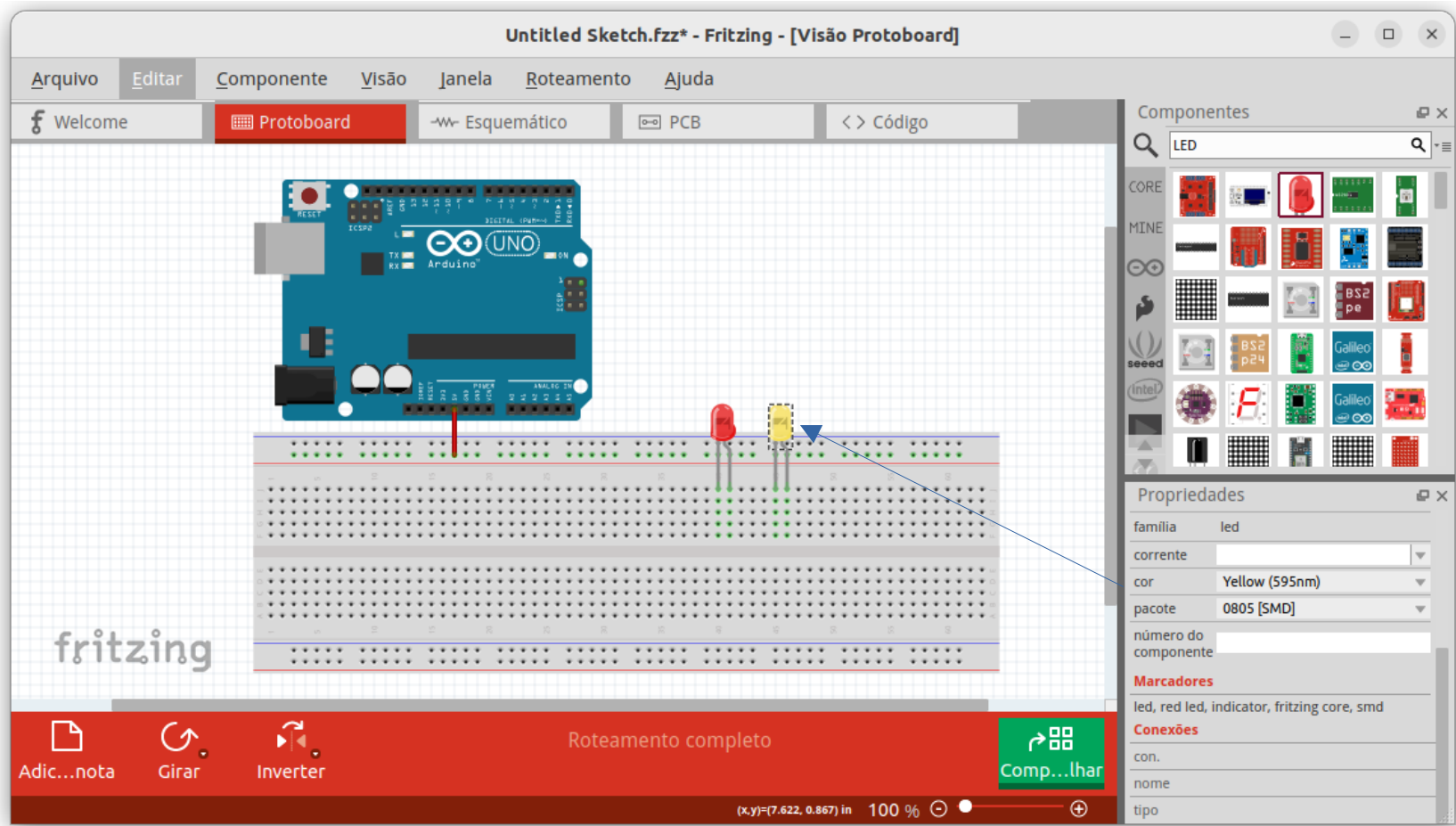
\* no caso de a cor do fio não ser esta

Na aba de componentes, digite LED, selecione-o e arraste-o para o *protoboard*.



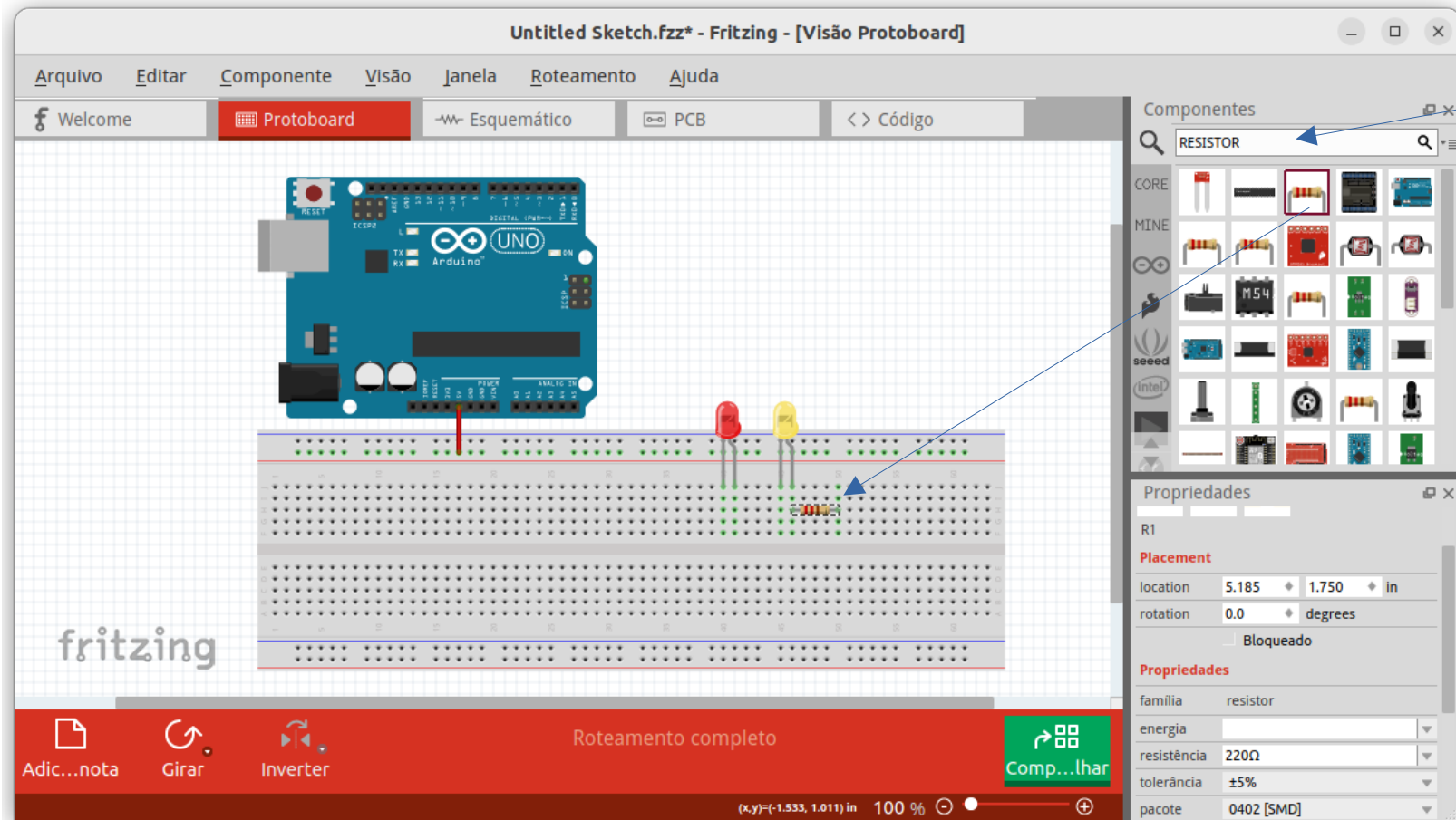
\* não se preocupe com a cor, dá para mudar depois

Coloque mais um LED no *proto board*. Depois, na aba de propriedades, ajuste a cor do LED para amarelo.

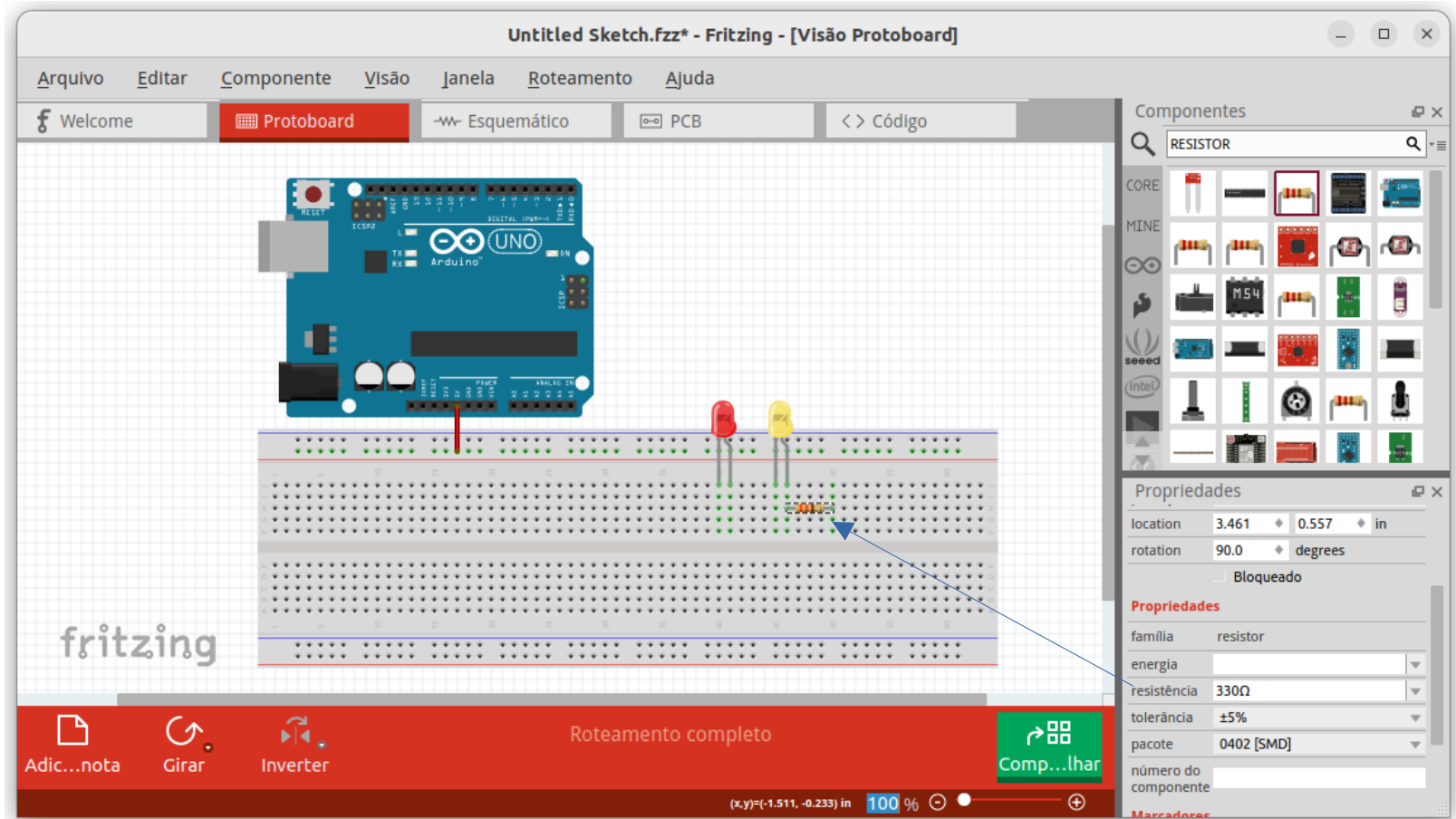




Na aba de componentes, procure resistor e arraste um para o *proto*board. O valor pode ser ajustado depois.



Ajuste o valor do resistor na aba de propriedades (para o valor que você está usando). As cores mudam automaticamente.

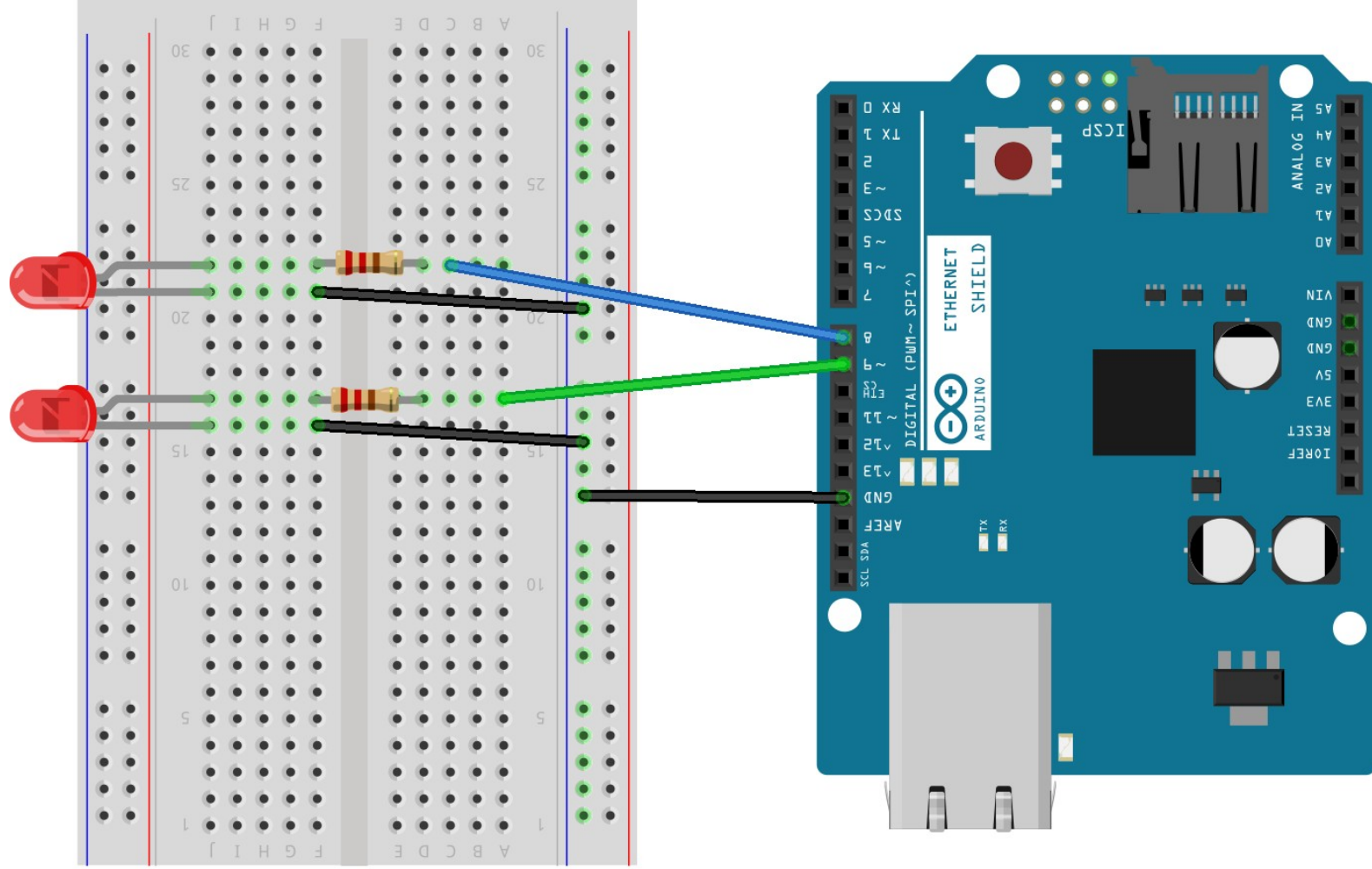


# Lembre-se

- Seu desenho deverá ter 6 leds (2 vermelhos, 2 verdes e 2 amarelos), e seis resistores limitadores de corrente.
- Você terá usado 6 portas do Arduino. Ligue-os de acordo com a numeração que você usou em seu código, para a que a documentação fique correta.

# Pisca Pisca





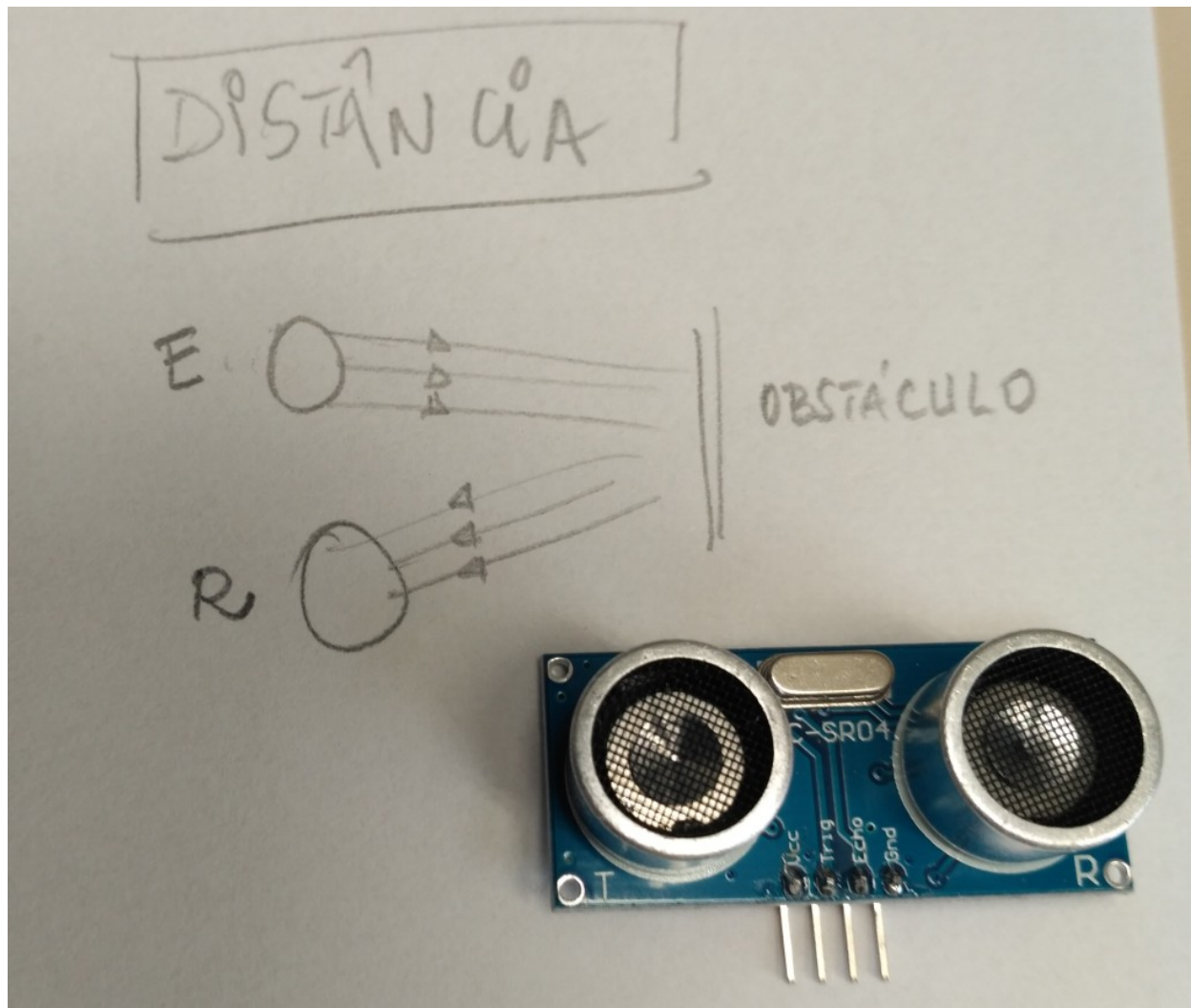
fritzing

```

1 //ligar os leds por meio de resistores para limitar a corrente
2 #define LED1 8
3 #define LED2 9
4
5 class Pisca {                                //Declara a classe Pisca
6     public:                                  //Público - acessível fora da classe
7     int pino;                                //Qual porta será utilizada
8     Pisca (int quem) {                      //Recebe a porta escolhida
9         pinMode(quem, OUTPUT);              //Coloca a porta escolhida no modo saída
10        pino = quem;                         //Armazena a porta
11    }
12
13    void piscar() {                          //Início do MÉTODO piscar
14        digitalWrite(pino, HIGH);            //Escreve um valor alto na porta 'pino'
15        delay(500);                          //Aguarda 500 ms
16        digitalWrite(pino, LOW);             //Escreve um valor baixo na porta 'pino'
17        delay(250);                          //Aguarda 250 ms
18    }
19 };                                           //Termina a classe Pisca
20
21 Pisca led1(LED1);                          //Declara um OBJETO da classe Pisca
22 Pisca led2(LED2);                          //Declara um OBJETO da classe Pisca
23
24 void setup() {
25
26     //Nada em especial a realizar
27 }
28
29 void loop() {
30     led1.piscar();                          //Chama o método piscar() para o objeto 1
31     led2.piscar();                          //Chama o método piscar() para o objeto 2
32 }

```

# Vamos trabalhar com o sensor de ultrassom





# Medidor de distância

- Baseado em seus experimentos anteriores, crie um medidor de distância

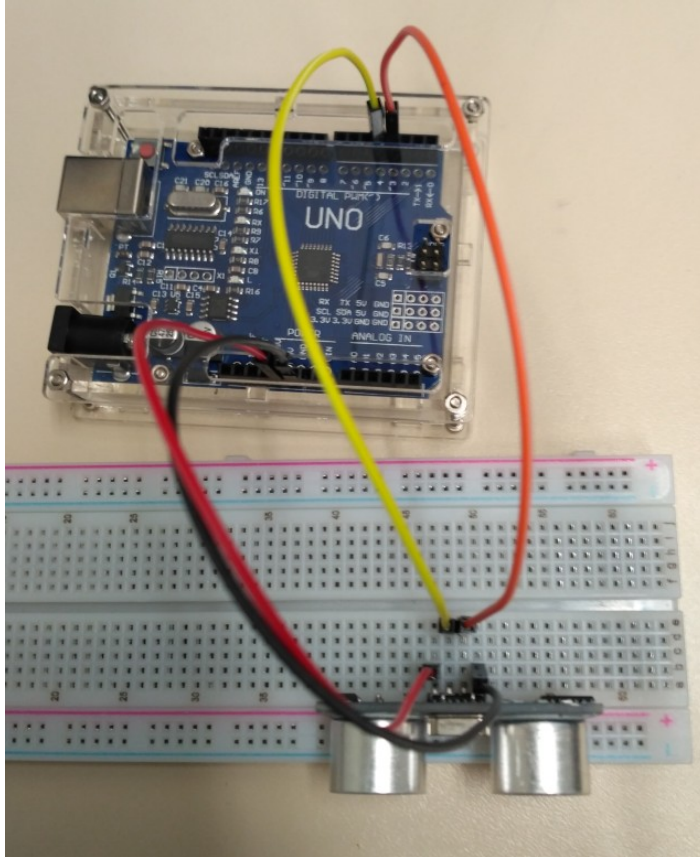
# O que você precisará?

- Arduino configurado na IDE
- Uma placa de experimentação e fios
- Um emissor – receptor de ultrassom SR-04

# Sequência

- Desligue o cabo USB
- Ligue os componentes na placa
- Conecte os fios à placa do Arduino; anote as portas utilizadas
- Elabore o código

# Ligação



O pino VCC do SR-04 é ligado ao 5V da placa do Arduino.

O pino GND do SR-04 é ligado ao GND da placa do Arduino.

O pino TRIG do SR-04 é ligado na digital 4 da placa do Arduino.

O pino ECHO do SR-04 é ligado na digital 3 da placa do Arduino.

(Se usar outras portas troque no código exemplo a seguir)

# Código



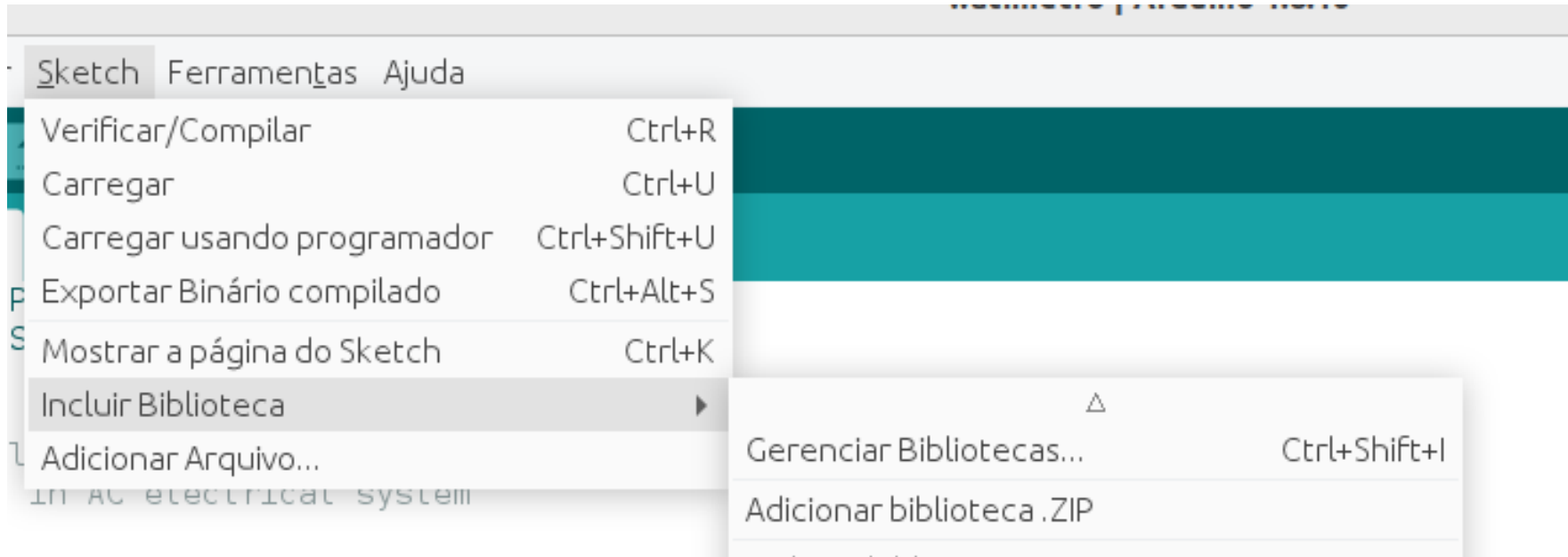
```
Arquivo  Editar  Sketch  Ferramentas  Ajuda
✓ → 📄 ⬆ ⬇ Salvar
medeDistancia
1 #include <Ultrasonic.h>
2
3 #define trig 4
4 #define echo 3
5
6 Ultrasonic ultrasonic(trig, echo);
7
8 void setup()
9 {
10   pinMode(echo, INPUT);
11   pinMode(trig, OUTPUT);
12   Serial.begin(9600);
13   Serial.println("Iniciando...\n\n");
14 }
15
16 void loop()
17 {
18   digitalWrite(trig, LOW);
19   delayMicroseconds(2);
20   digitalWrite(trig, HIGH);
21   delayMicroseconds(10);
22   digitalWrite(trig, LOW);
23   int distancia = (ultrasonic.Ranging(CM));
24   Serial.print("Distancia em CM: ");
25   Serial.println(distancia);
26   delay(2000);
27 }
```

Quem faz todo o trabalho é a biblioteca incluída no código, “Ultrasonic.h”.

SE ocorrer um erro de compilação indicando que a biblioteca não está instalada, veja o slide seguinte.



# SE for necessário incluir a biblioteca



Além da óbvia pesquisa na internet, você poderá encontrar bibliotecas em:  
**<https://www.arduino.cc/reference/en/libraries/>**

# Código



```

Arquivo  Editar  Sketch  Ferramentas  Ajuda
✓ → 📄 ⬆ ⬇ Salvar
medeDistancia
1 #include <Ultrasonic.h>
2
3 #define trig 4
4 #define echo 3
5
6 Ultrasonic ultrasonic(trig, echo);
7
8 void setup()
9 {
10   pinMode(echo, INPUT);
11   pinMode(trig, OUTPUT);
12   Serial.begin(9600);
13   Serial.println("Iniciando...\n\n");
14 }
15
16 void loop()
17 {
18   digitalWrite(trig, LOW);
19   delayMicroseconds(2);
20   digitalWrite(trig, HIGH);
21   delayMicroseconds(10);
22   digitalWrite(trig, LOW);
23   int distancia = (ultrasonic.Ranging(CM));
24   Serial.print("Distancia em CM: ");
25   Serial.println(distancia);
26   delay(2000);
27 }
    
```

Quem faz todo o trabalho é a biblioteca incluída no código, “Ultrasonic.h”.

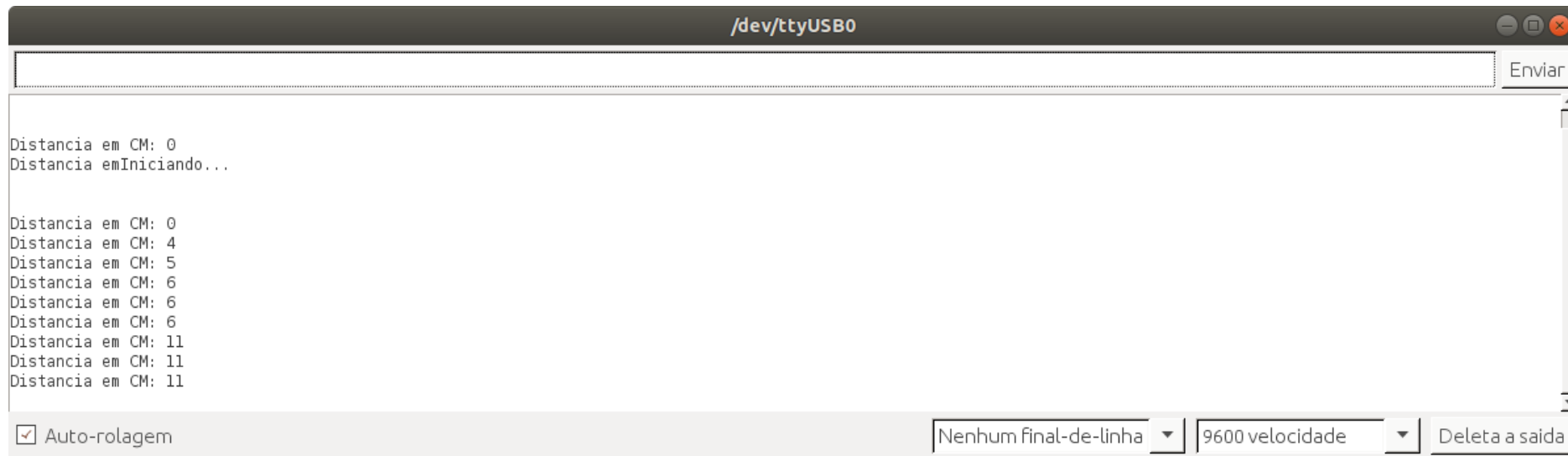
O valor da distância está sendo enviado à saída Serial.

Para vê-lo, vamos ligar o Monitor Serial.

Basta clicar no ícone à esquerda da tela:



# Ferramentas / Monitor serial



# Melhorando

medeDistanciaCMPol

```
1 #include <Ultrasonic.h>
2
3 #define trig 4
4 #define echo 3
5
6 Ultrasonic ultrasonic(trig, echo);
7
8 void setup()
9 {
10   pinMode(echo, INPUT);
11   pinMode(trig, OUTPUT);
12   Serial.begin(9600);
13   Serial.println("Iniciando...\n\n");
14 }
15
16 void loop()
17 {
18   digitalWrite(trig, LOW);
19   delayMicroseconds(2);
20   digitalWrite(trig, HIGH);
21   delayMicroseconds(10);
22   digitalWrite(trig, LOW);
23   int centimetros = (ultrasonic.Ranging(CM));
24   int polegadas = (ultrasonic.Ranging(INC));
25   Serial.print("Distancia em CM: ");
26   Serial.println(centimetros);
27   Serial.print("Distancia em Pol: ");
28   Serial.println(polegadas);
29   delay(2000);
30 }
```

/dev/ttyUSB0

Enviar

Distancia em Pol: 963  
Distancia em CM: 2458  
Distancia em Pol: 963  
Distancia em CM: 7  
Distancia em Pol: 1  
Distancia em CM: 7  
Distancia em Pol: 0  
Distancia em CM: 7  
Distancia em Pol: 2  
Distancia em CM: 6  
Distancia em Pol: 2  
Distancia em CM: 6  
Distancia em Pol: 2  
Distancia em CM: 6  
Distancia em Pol: 2  
Distancia em CM: 6  
Distancia em Pol: 2  
Distancia em CM: 2459  
Distancia

☒ Auto-rolagem

Nenhum final-de-linha

9600 velocidade

Deleta a saida

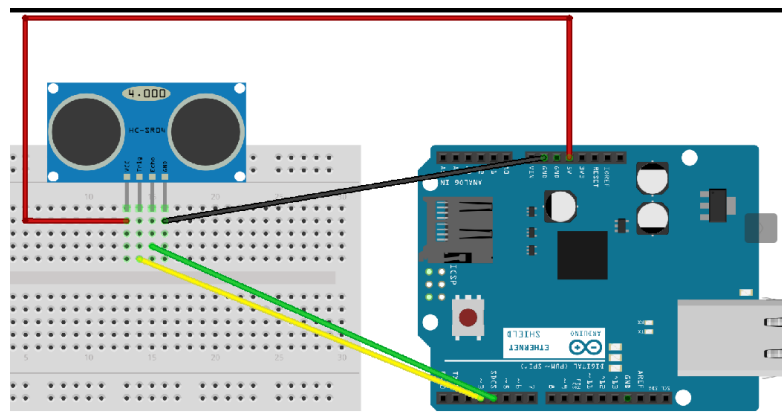
# Sem biblioteca

- Se você tiver informações a respeito do módulo que está utilizando (ou se você desenvolveu o módulo...), poderá controlar suas características diretamente, sem utilizar uma biblioteca.
  - Em geral as características do módulo ou de seu circuito integrado de controle estão disponíveis na forma de *datasheets* (folhas de dados)



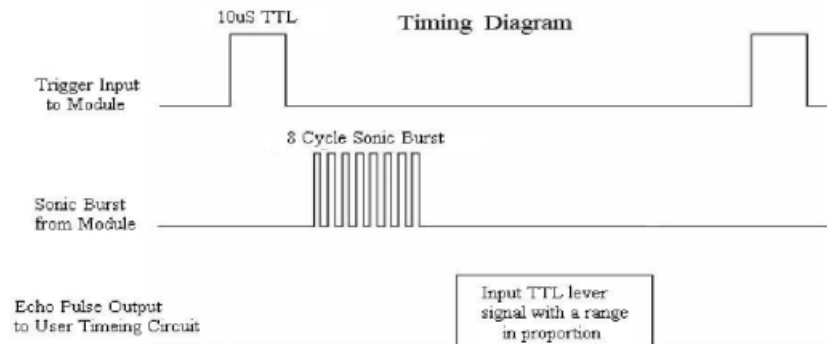


# Ultrassom



```
1 int pinoTrigger = 3;
2 int pinoEcho = 4;
3
4 void setup()
5 {
6   Serial.begin(9600);
7   pinMode(pinoTrigger, OUTPUT);
8   pinMode(pinoEcho, INPUT);
9 }
```

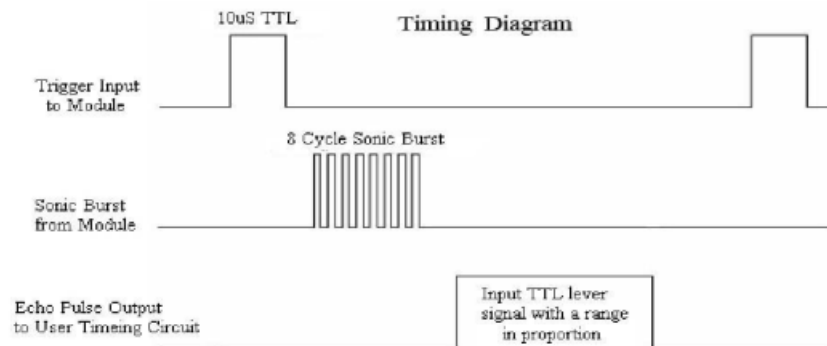
The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion. You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula:  $\mu\text{S} / 58 = \text{centimeters}$  or  $\mu\text{S} / 148 = \text{inch}$ ; or: the range = high level time \* velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.

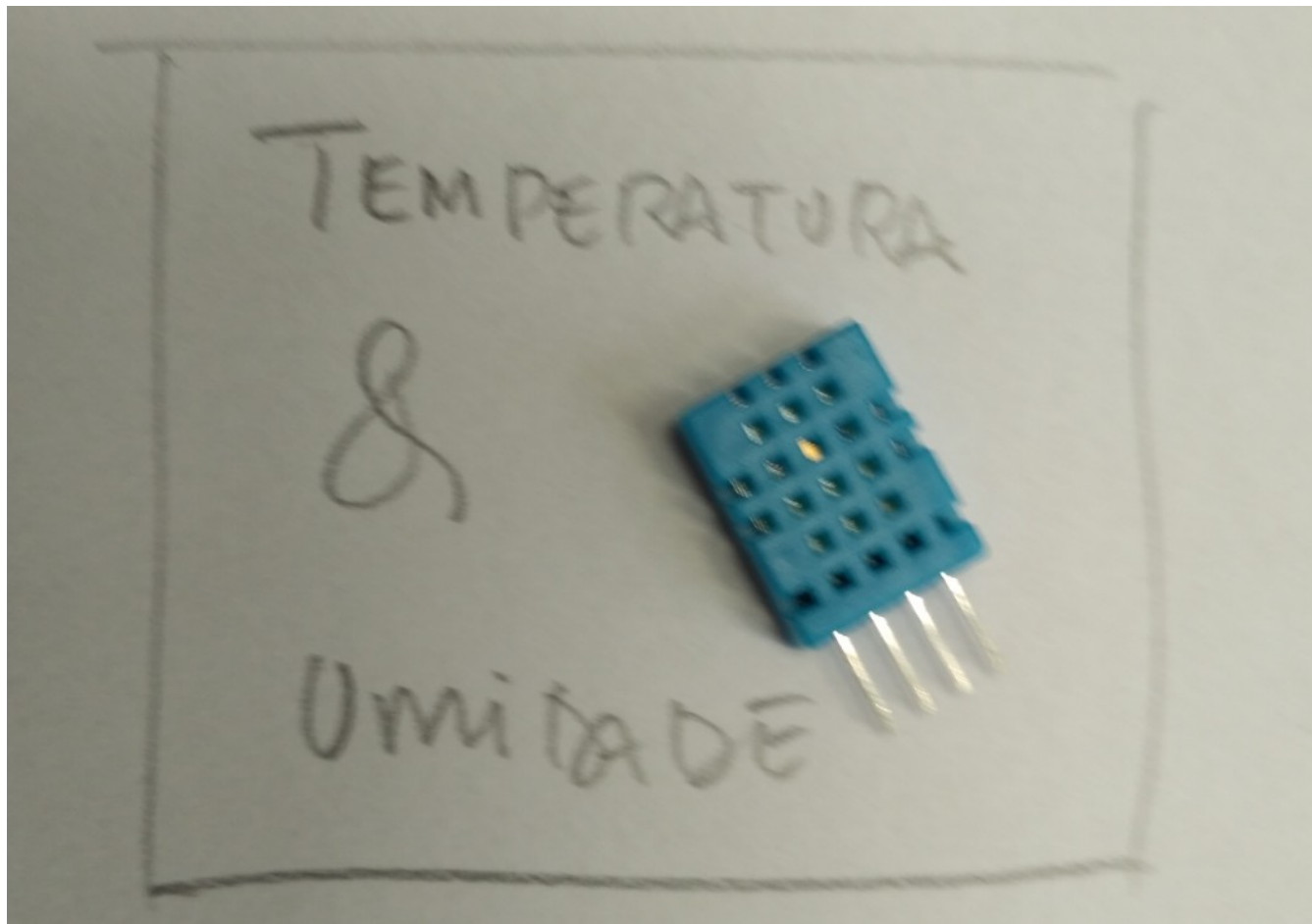


# Ultrassom

```
11 float mede()
12 {
13   digitalWrite(pinoTrigger, LOW);
14   delayMicroseconds(3);
15   digitalWrite(pinoTrigger, HIGH);
16   delayMicroseconds(10);
17   digitalWrite(pinoTrigger, LOW);
18   float tempoUs = pulseIn(pinoEcho, HIGH);
19   return (tempoUs / 58);
20 }
21
22 void loop()
23 {
24   float distancia = mede();
25   Serial.print("Distancia medida: ");
26   Serial.print(distancia);
27   Serial.println(" centimetros");
28   delay(500);
29 }
```

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion. You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula:  $\mu\text{S} / 58 = \text{centimeters}$  or  $\mu\text{S} / 148 = \text{inch}$ ; or: the range = high level time \* velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.

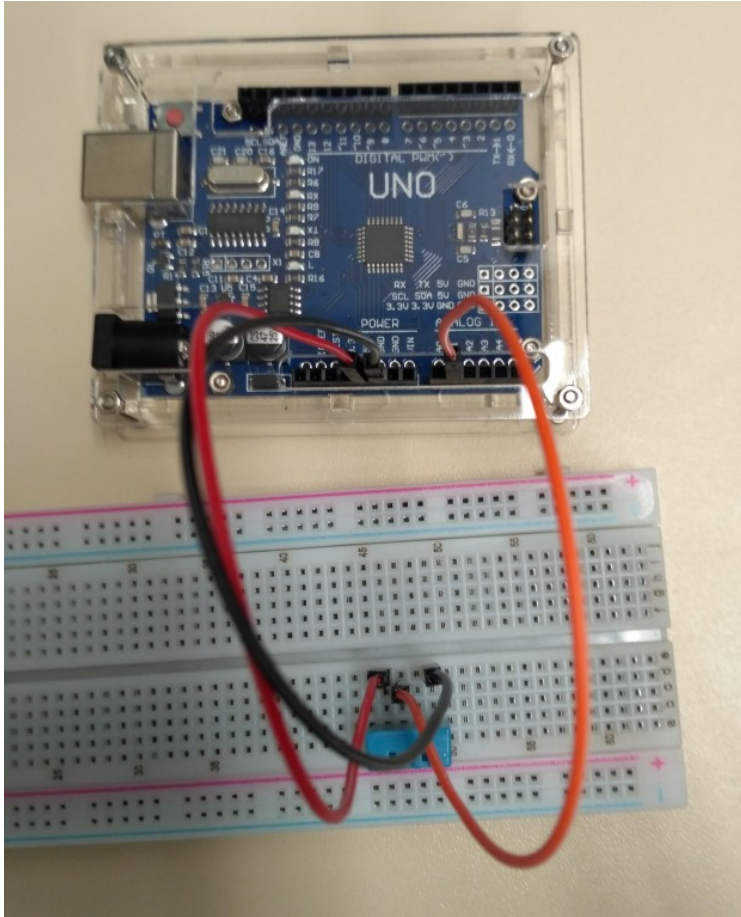




# O que você precisará?

- Arduino configurado na IDE
- Uma placa de experimentação e fios
- Um sensor de temperatura e umidade DHT-11

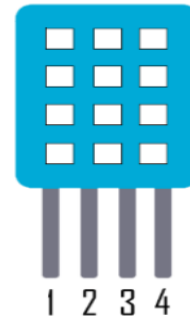
# Ligação



O pino VCC do DHT11 é o pino 1, e é ligado ao 5V da placa do Arduino.

O pino GND do DHT11 é o pino 4, e é ligado ao GND da placa do Arduino.

O pino de saída do DHT11 é ligado na entrada analógica A1 da placa do Arduino.



**Pinos do  
Sensor  
DHT-11**

1 = VCC  
2 = Saída  
3 = sem uso  
4 = GND



# Atenção



# Sequência

- Desligue o cabo USB
- Ligue os componentes na placa
- Conecte os fios à placa do Arduino; anote as portas utilizadas
- Elabore o código

tempUmidadeDHT11

```
1 #include "DHT.h"
2
3 #define sensor A1
4 #define tipo DHT11
5
6 DHT dht(sensor, tipo);
7
8 void setup()
9 {
10   Serial.begin(9600);
11   Serial.println("Iniciando...\n\n");
12   dht.begin();
13 }
14
15 void loop()
16 {
17
18   float temperatura = dht.readTemperature();
19   float umidade = dht.readHumidity();
20
21   if (isnan(temperatura) || isnan(umidade))
22   {
23     Serial.println("Comunicação falhou!\n");
24   }
25   else
26   {
27     Serial.print("Temperatura = ");
28     Serial.print(temperatura);
29     Serial.print(" C, e umidade = ");
30     Serial.print(umidade);
31     Serial.println(" %");
32
33     delay(2000);
34   }
35 }
```

# Código



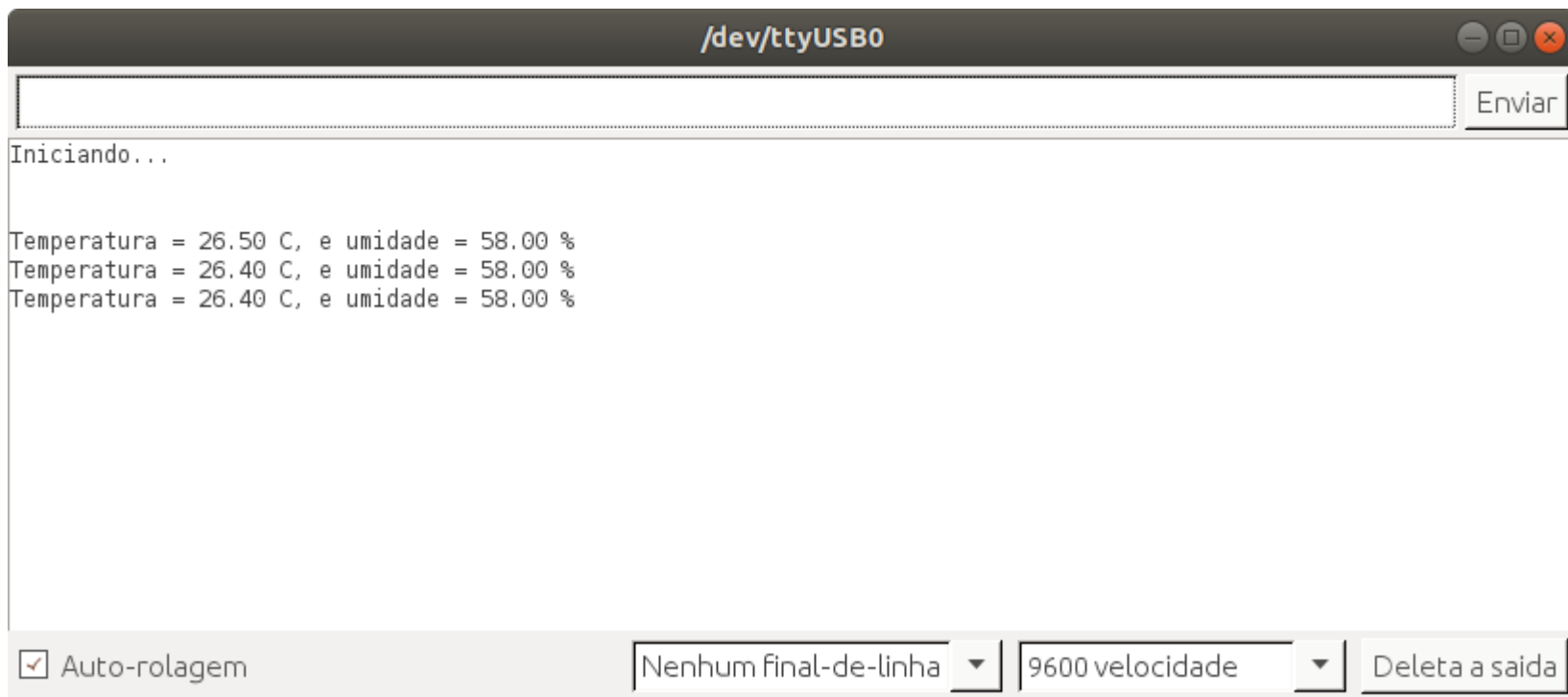
Quem faz todo o trabalho agora é a biblioteca incluída no código, “DHT.h”.

Para ver o resultado, vamos ligar o Monitor Serial.

Basta clicar no ícone à esquerda da tela:



# Ferramentas / Monitor serial



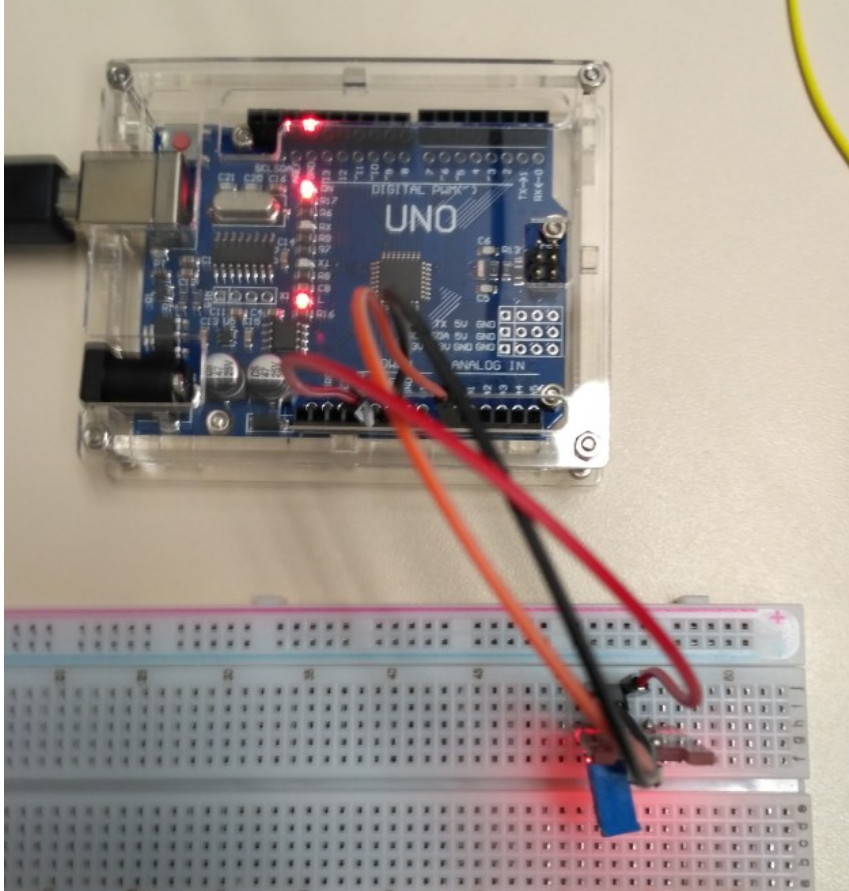


# O que você precisará?

- Arduino configurado na IDE
- Uma placa de experimentação e fios
- Um sensor de som KY-037



# Ligação



O pino A0 do KY-037 é ligado na entrada analógica A0 da placa do Arduino.

O pino G do KY-037 é ligado ao GND da placa do Arduino.

O pino + do KY-037 é ligado ao 5V da placa do Arduino.

# Sequência

- Desligue o cabo USB
- Ligue os componentes na placa
- Conecte os fios à placa do Arduino; anote as portas utilizadas
- Elabore o código



# Código

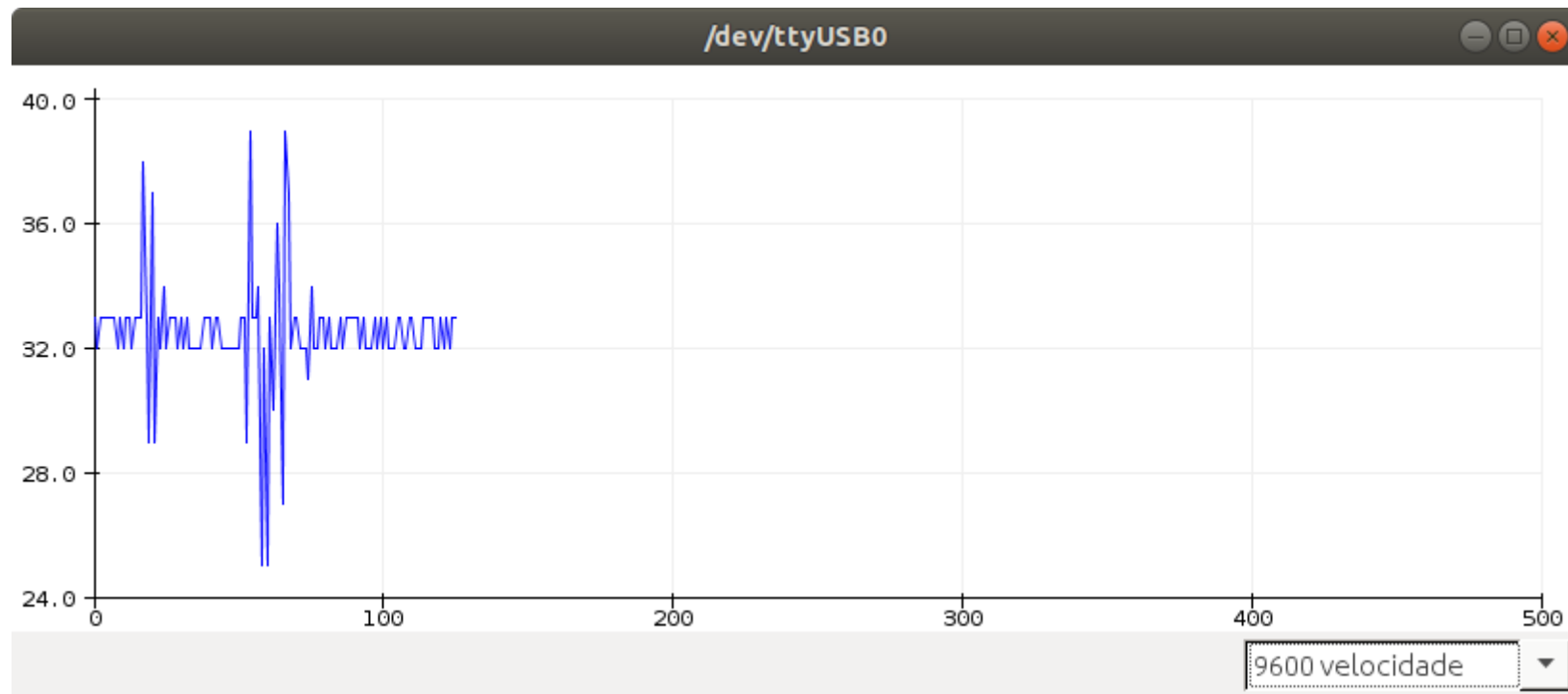
Para ver o resultado, vamos ligar o `Plotter Serial`.

## Ferramentas / **Plotter Serial**

A screenshot of the Arduino IDE interface. The title bar at the top reads "somKY037 | Arduino 1.8.5". Below the title bar is a menu bar with "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". Under the "Sketch" menu, a dropdown is open showing "somKY037". The main workspace contains the following code:

```
1  
2 void setup() {  
3  
4   Serial.begin(9600);  
5  
6 }  
7  
8 void loop() {  
9  
10  Serial.println(analogRead(A0));  
11  
12  delay(100);  
13  
14 }
```

# Ferramentas / Plotter Serial



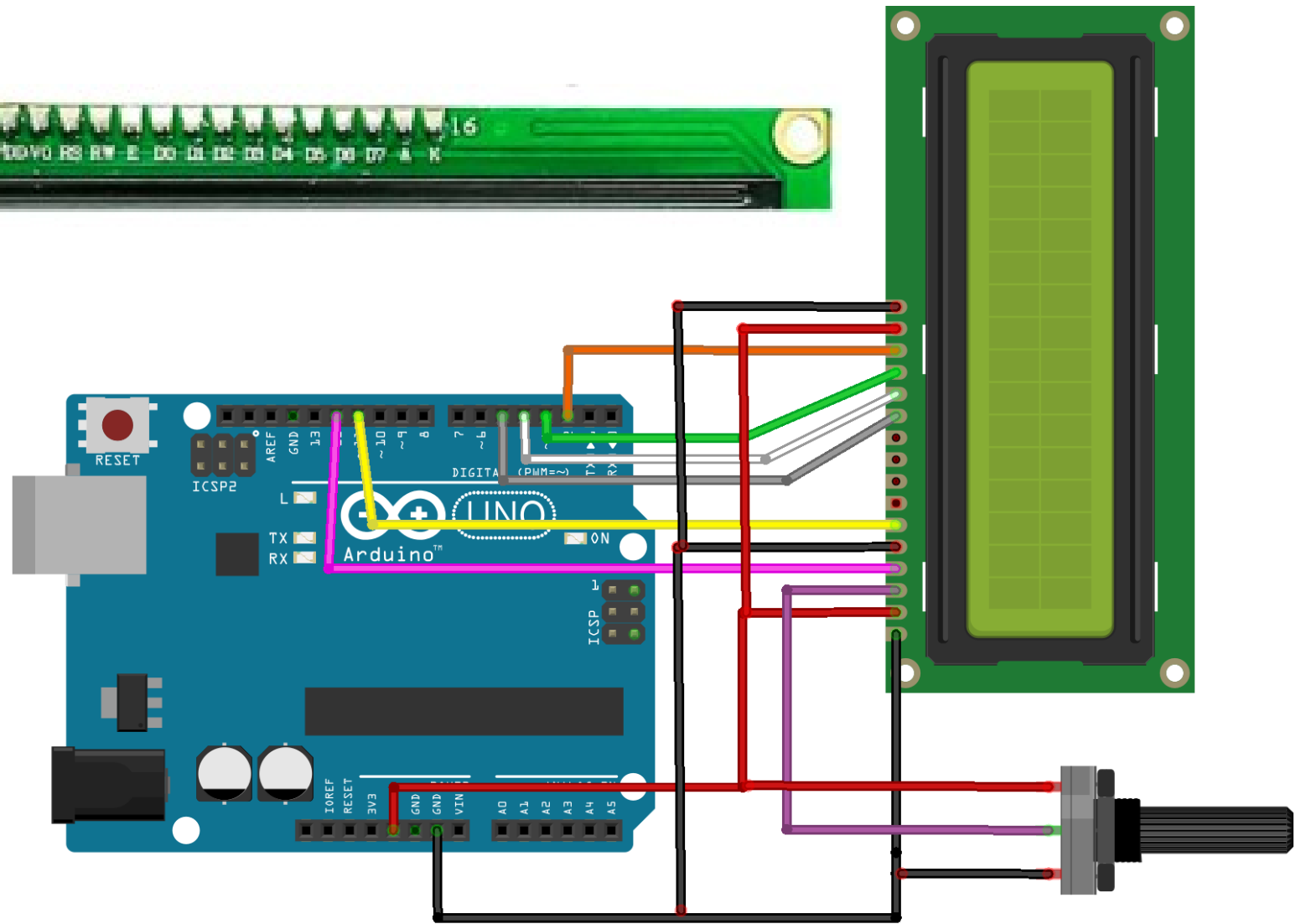
# Display LCD



Pino	Identificação	Função
1	VSS	Ligação com o GND (0Volts)
2	VDD	Ligação com o VCC (+5Volts)
3	V0	Tensão de controle do contraste do LCD
4	RS	Seletor de registro ( <i>register select</i> ), utilizado para controle do <i>display</i>
5	RW	Seletor leitura/ escrita ( <i>read/ write</i> ), utilizado para controle do <i>display</i> . No código, este pino é visto como opcional em algumas aplicações, de foma que pode aparecer nos circuitos conectado ao GND, sem conexão ao Arduino.
6	E	Seletor de habilitação ( <i>enable</i> ), utilizado para controle do <i>display</i>
7	D0	Bit menos significativo da palavra de dados, data 0
8	D1	Bit 1 da palavra de dados
9	D2	Bit 2 da palavra de dados
10	D3	Bit 3 da palavra de dados
11	D4	Bit 4 da palavra de dados
12	D5	Bit 5 da palavra de dados
13	D6	Bit 6 da palavra de dados
14	D7	Bit mais significativo da palavra de dados, data 7



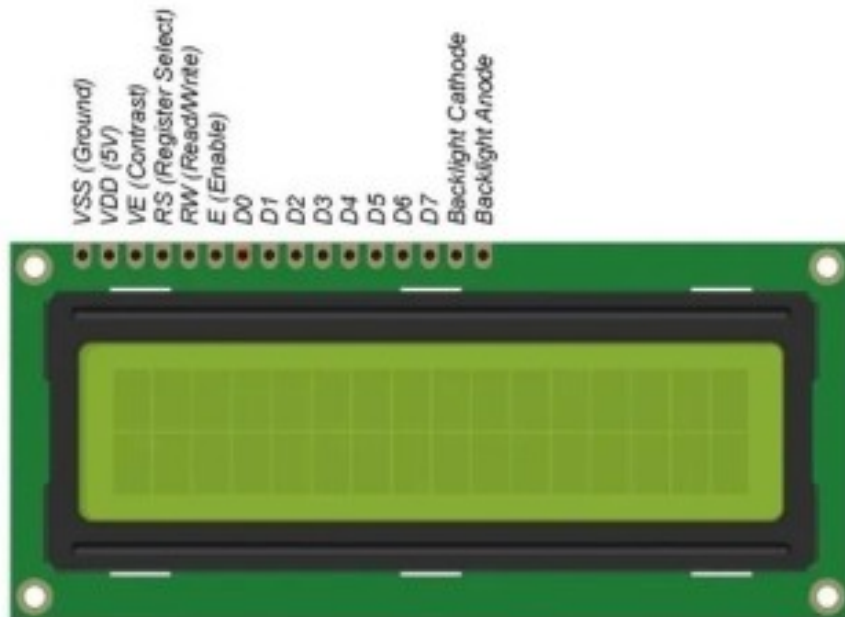
<p>Nem todos os <i>displays</i> possuem um LED de luz de fundo (<i>backlight</i>). Portanto, podem não possuir os dois pinos que seguem (o pino 15 e o pino 16 – neste caso tem somente os 14 pinos anteriores), relativos à conexão do LED.</p> <p>A propósito, se você não conectar o LED de <i>backlight</i>, o <i>display</i> funcionará normalmente, só terá menos luminosidade e, em algumas aplicações, em função da iluminação do ambiente, um contraste mais difícil de perceber.</p>		
15	A	Ânodo do LED de <i>backlight</i> , ligar ao +5V
16	K	Cátodo do LED de <i>backlight</i> ; ligar o GND (0V) por meio de resistor limitador de corrente (pode acontecer de o resistor estar incluído no módulo, mas é sempre bom prevenir... - o resistor também pode ser ligado no pino 15, ao invés de no 16, é indiferente).



fritzing



# Display LCD



O *display* ao lado segue a nomenclatura e numeração mostradas na tabela acima.  
Às vezes, os identificadores vem impressos:



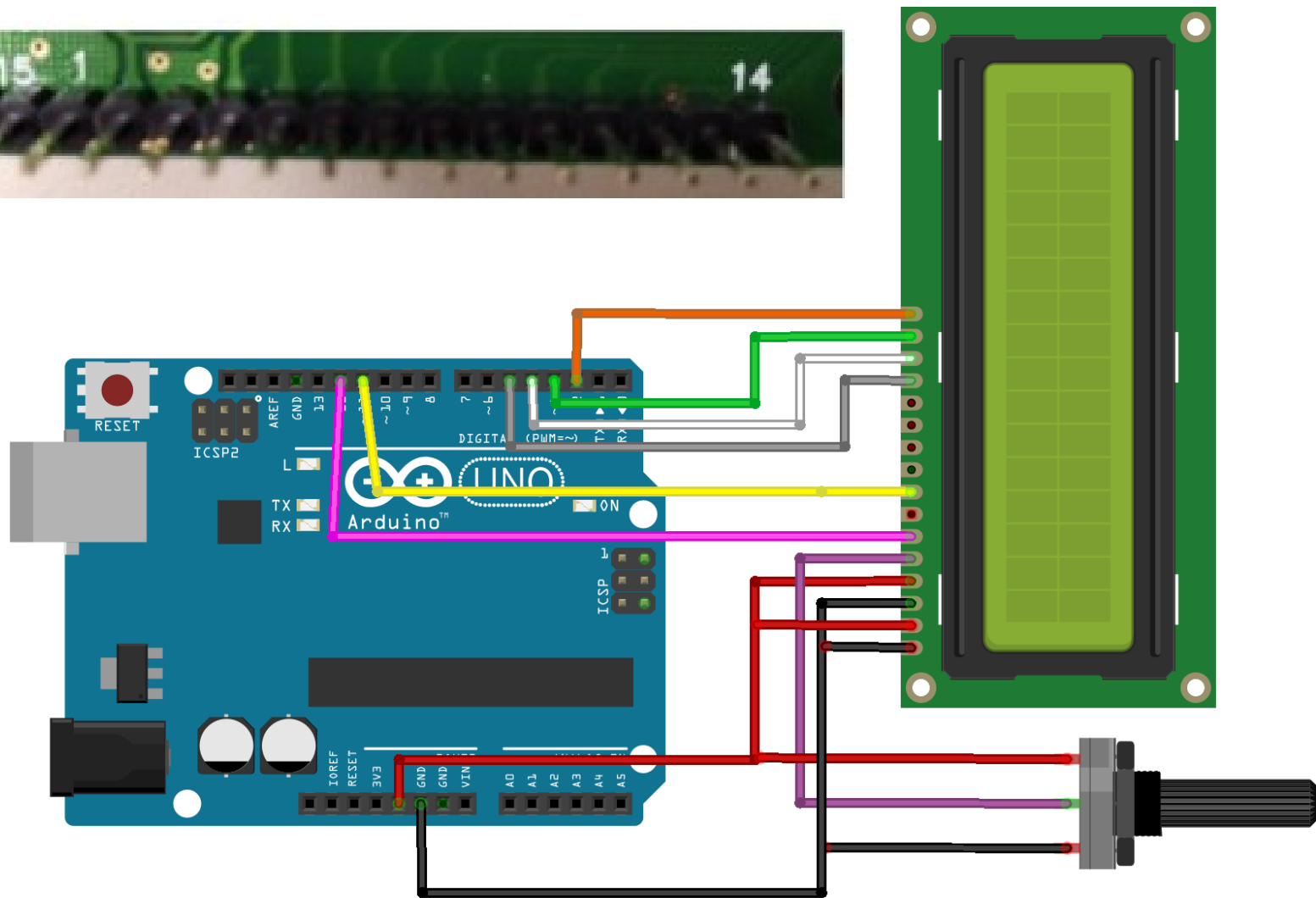
Em geral a correspondência entre o sinal/identificador e o número do pino não muda, PORÉM, pode haver mudança na sequência de pinos:



Um exemplo de declaração no IDE do Arduino:

```
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;  
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

Quer dizer que a ligação Arduino x *Display*, será: o pino 12 do Arduino no pino 4 do *display*; o pino 11 do Arduino no pino 6 do *display*; o pino 5 do Arduino no pino 11 do *display*; ...



fritzing



```
//Adaptado de: https://docs.arduino.cc/learn/electronics/lcd-displays
//A declaração a seguir informa o compilador que iremos utilizar a biblioteca 'LiquidCrystal'
#include <LiquidCrystal.h>
/*
 * Os termos, usados nas constantes a seguir, rs, en, d4, d5, d6 e d7 referem-se às conexões de
 * controle do display. Cada display pode tê-los em posições diferentes
 * os números utilizados são os pinos do Arduino que você vai utilizar, e podem ser modificados
 * se você desejar
 */
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

/*
 * A declaração a seguir cria um objeto 'lcd', baseado na biblioteca LiquidCrystal, e informa,
 * na sequência, os pinos de controle
 * As formas de inicialiação / controle, são:
 *   LiquidCrystal(rs, enable, d4, d5, d6, d7)
 *   LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
 *   LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
 *   LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
 * Cada uma destas formas está ligada a diferentes conexões físicas e diferentes possibilidades
 * de uso do display
 */
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

```
const int columnas = 16, linhas = 2;

/*
 * Para desenhar caracteres especiais:
 * https://maxpromer.github.io/LCD-Character-Creator/
 * https://omerk.github.io/lcdchargen/
 */
byte smiley[8] = {
    B01110,
    B10111,
    B11110,
    B11111,
    B11000,
    B11111,
    B11010,
    B10010,
};

void setup()
{
    lcd.begin(columnas, linhas);
    lcd.createChar(0, smiley);
}
```





```
void loop()
{
    lcd.clear();
    lcd.display();
    lcd.setCursor(4, 0);
    lcd.print("CyberRex");
    lcd.setCursor(0, 1);
    lcd.print("E.M. Omar Sabbag");
    delay(1000);
    lcd.noDisplay();
    delay(500);
    lcd.display();
    delay(500);
    lcd.noDisplay();
    delay(500);
    lcd.display();
    delay(1000);
}
```

```
lcd.clear();
lcd.home();
for (int pos = 0; pos<16; pos++)
{
    lcd.setCursor(pos, 0);
    lcd.write(byte(0));
    delay(100);
}
for (int pos = 15; pos>0; pos--)
{
    lcd.setCursor(pos, 1);
    lcd.write(byte(0));
    delay(100);
}
for (int pos = 0; pos<16; pos++)
{
    lcd.clear();
    lcd.setCursor(pos, 0);
    lcd.write(byte(0));
    delay(150);
}
```



```
for (int pos = 15; pos>0; pos--)  
{  
    lcd.clear();  
    lcd.setCursor(pos, 1);  
    lcd.write(byte(0));  
    delay(150);  
}
```

//Depois de testar este, vejam este: <https://create.arduino.cc/projecthub/aqzuonyt/arduino-dino-game-using-lcd-663aeb>

Este código para controle do *display* está disponível em:

<https://tiaplicada.ufpr.br/wp-content/uploads/2022/10/displaylcd.zip>

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

1. Utilizar bibliotecas de funções
2. Documentar usando o Fritzing
3. Pesquisar formas de controlar os módulos sem usar bibliotecas

# Parabéns!

