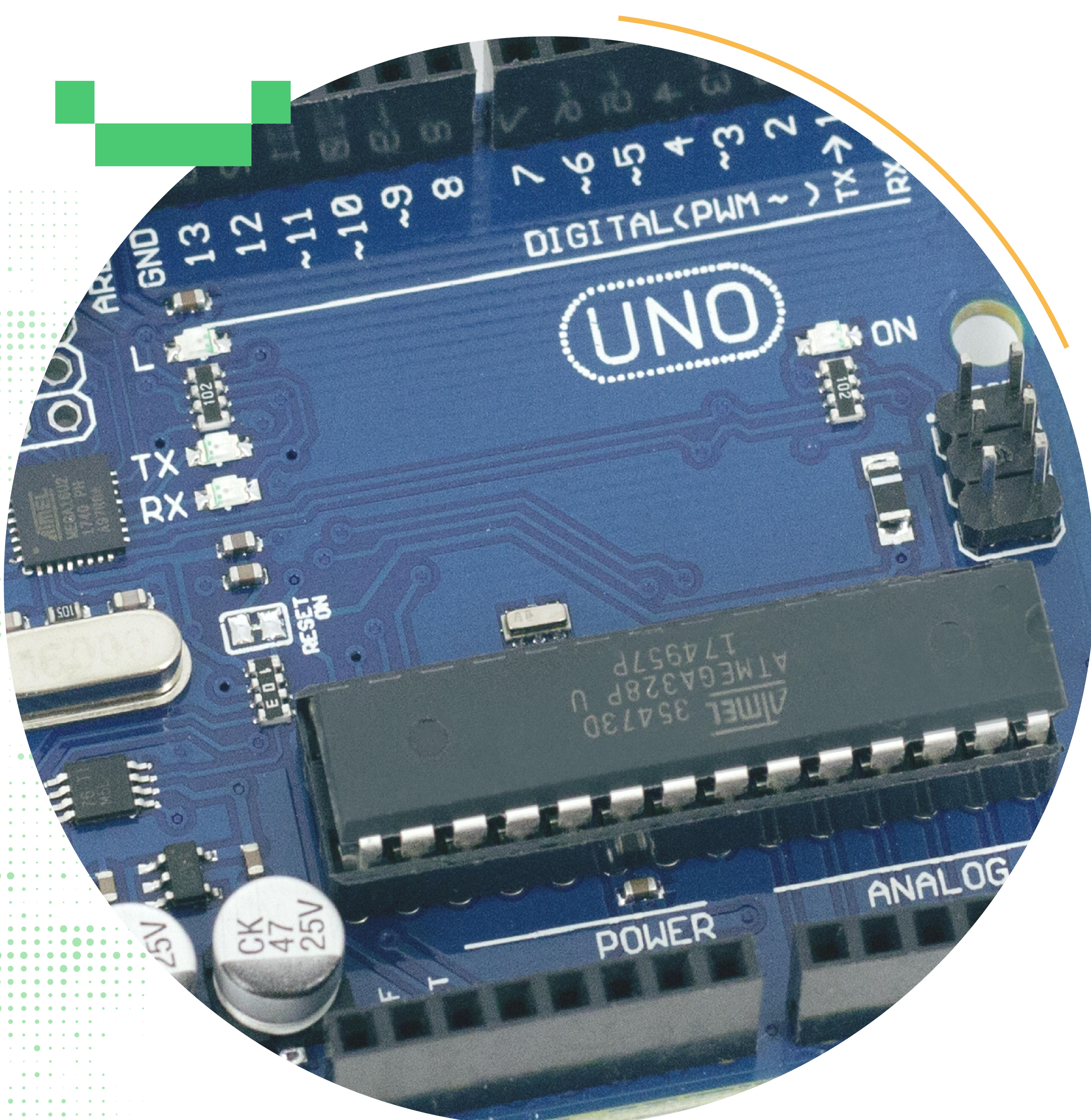


# GUIA ARDUINO PARA INICIANTES:

*Tudo que você precisa  
saber para começar*





---

## INTRODUÇÃO

O Arduino é o principal projeto de hardware open-source amplamente difundido no mundo. Muitos projetos ganham vida mundo afora graças a este poderoso hardware, especialmente o desenvolvimento de produtos IoT. Desenvolvedores, engenheiros, estudantes, empresas e makers em geral estão utilizando Arduino para inovar em seus projetos dos mais variados tipos.

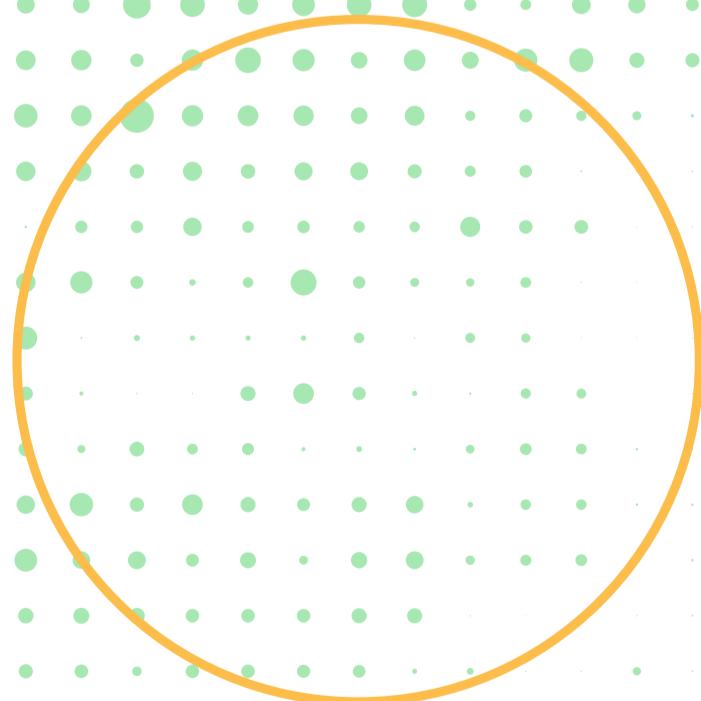
A comunidade Arduino também é muito forte ao redor do mundo e centenas de pessoas ajudam a espalhar o uso da ferramenta através de tutoriais, projetos, fóruns e conteúdos. É importante ressaltar também que o Arduino foi um grande propulsor do movimento maker no mundo todo e sua facilidade de uso permitiu que projetos eletrônicos utilizando os microcontroladores fossem adotados e desenvolvidos.

A ideia deste guia é ajudar quem está começando no mundo Arduino! Aqui você vai encontrar tudo o que precisa para dar seus primeiros passos e desenvolver projetos utilizando um dos hardwares mais incríveis do mundo da eletrônica!



\_BOA LEITURA.

# O QUE É ARDUINO

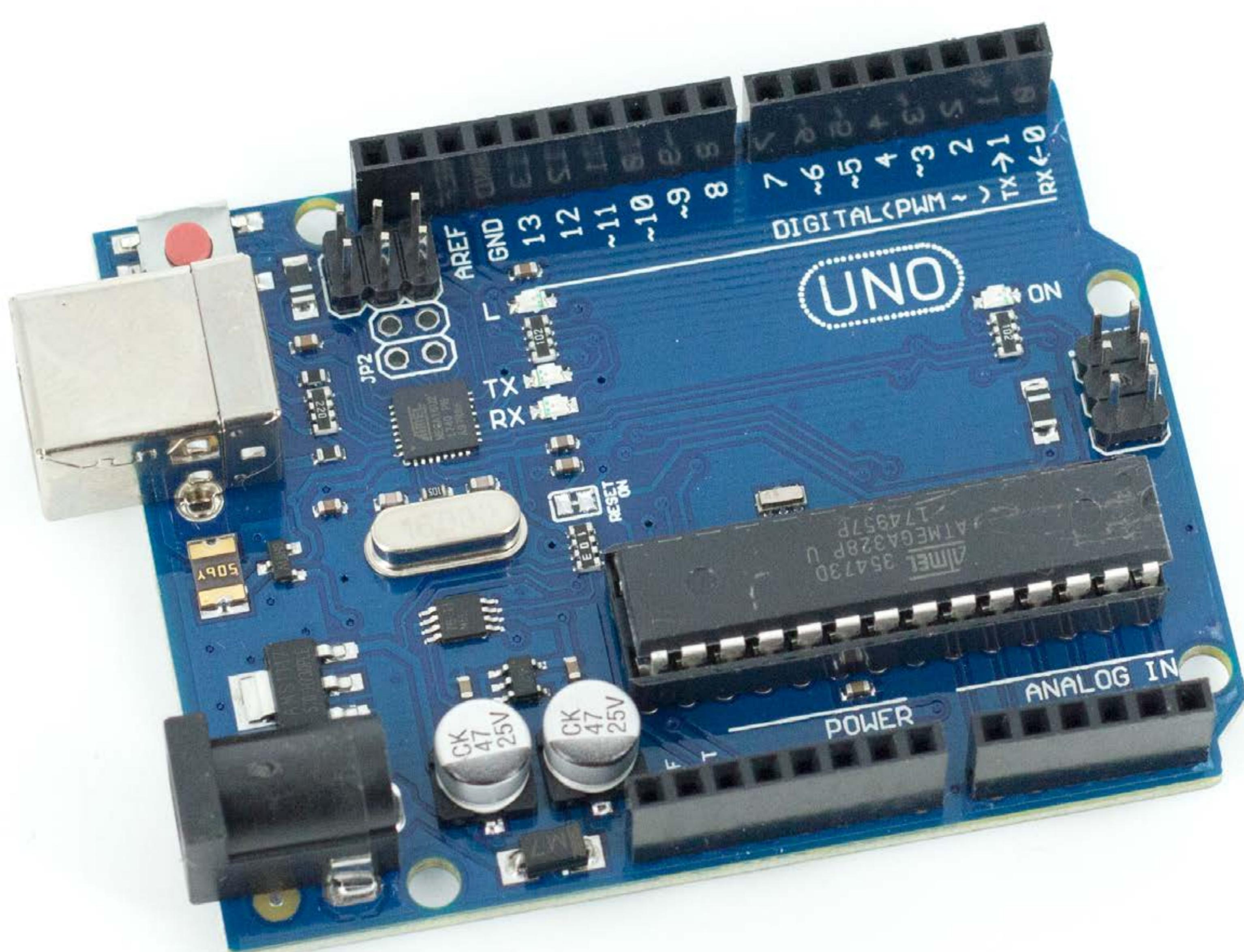




---

O **Arduino** foi criado em 2005 por um grupo de 5 pesquisadores: **Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis**. O objetivo era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores. Além disso, foi adotado o conceito de hardware livre, o que significa que qualquer um pode montar, modificar, melhorar e personalizar o Arduino, partindo do mesmo hardware básico.

Assim, foi criada uma placa composta por um **microcontrolador Atmel**, circuitos de entrada/saída e que pode ser facilmente conectada à um computador e programada via **IDE** (*Integrated Development Environment, ou Ambiente de Desenvolvimento Integrado*) utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB.



**\_ACESSÍVEL A  
ESTUDANTES  
E PROJETISTAS  
AMADORES**



Depois de programado, o microcontrolador pode ser usado de forma independente, ou seja, você pode colocá-lo para controlar um robô, uma lixeira, um ventilador, as luzes da sua casa, a temperatura do ar condicionado, pode utilizá-lo como um aparelho de medição ou qualquer outro projeto que vier à cabeça.

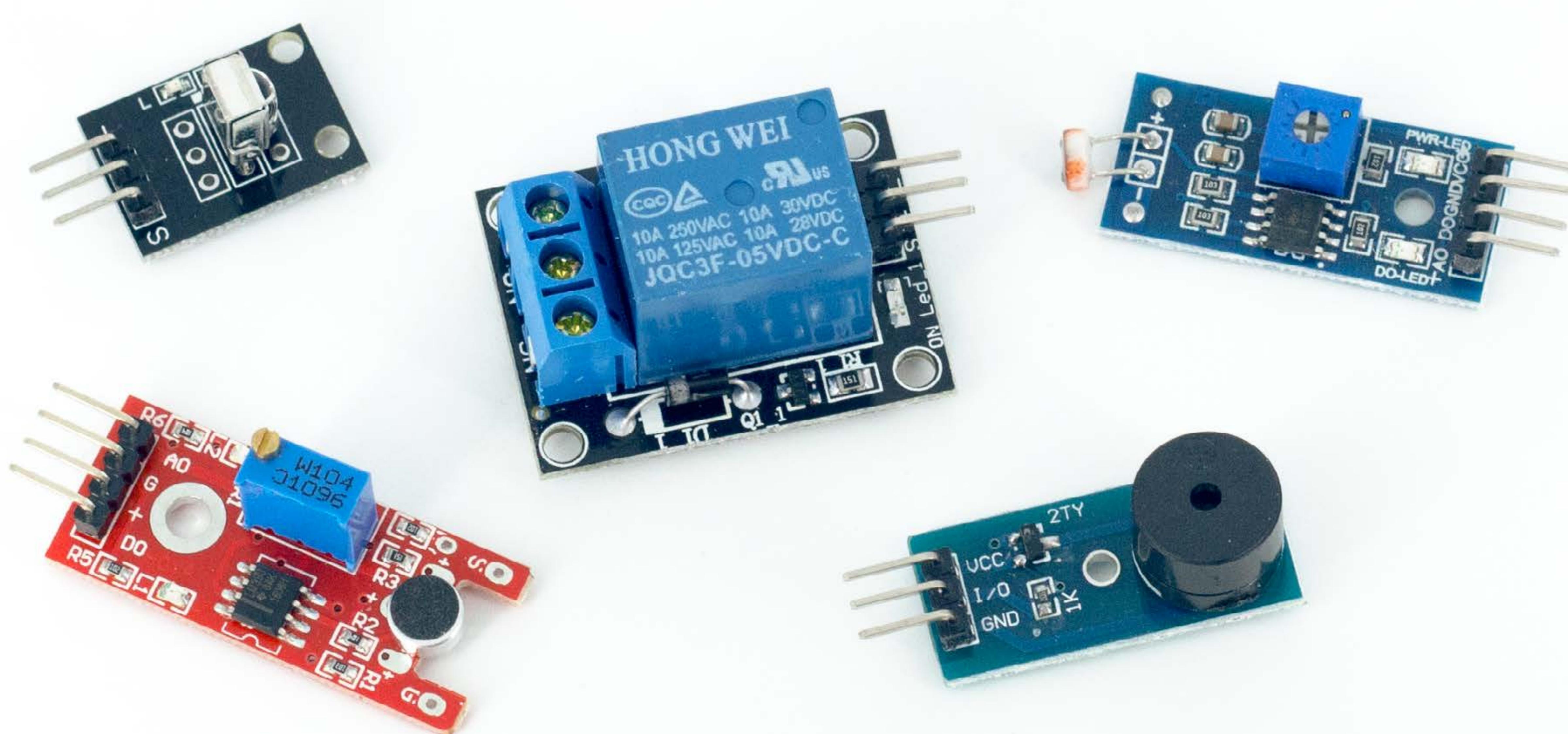
O vídeo abaixo mostra Massimo Banzi, um dos criadores do Arduino, falando um pouco sobre o processo de criação e desenvolvimento, e apresentando alguns projetos que utilizam a plataforma:



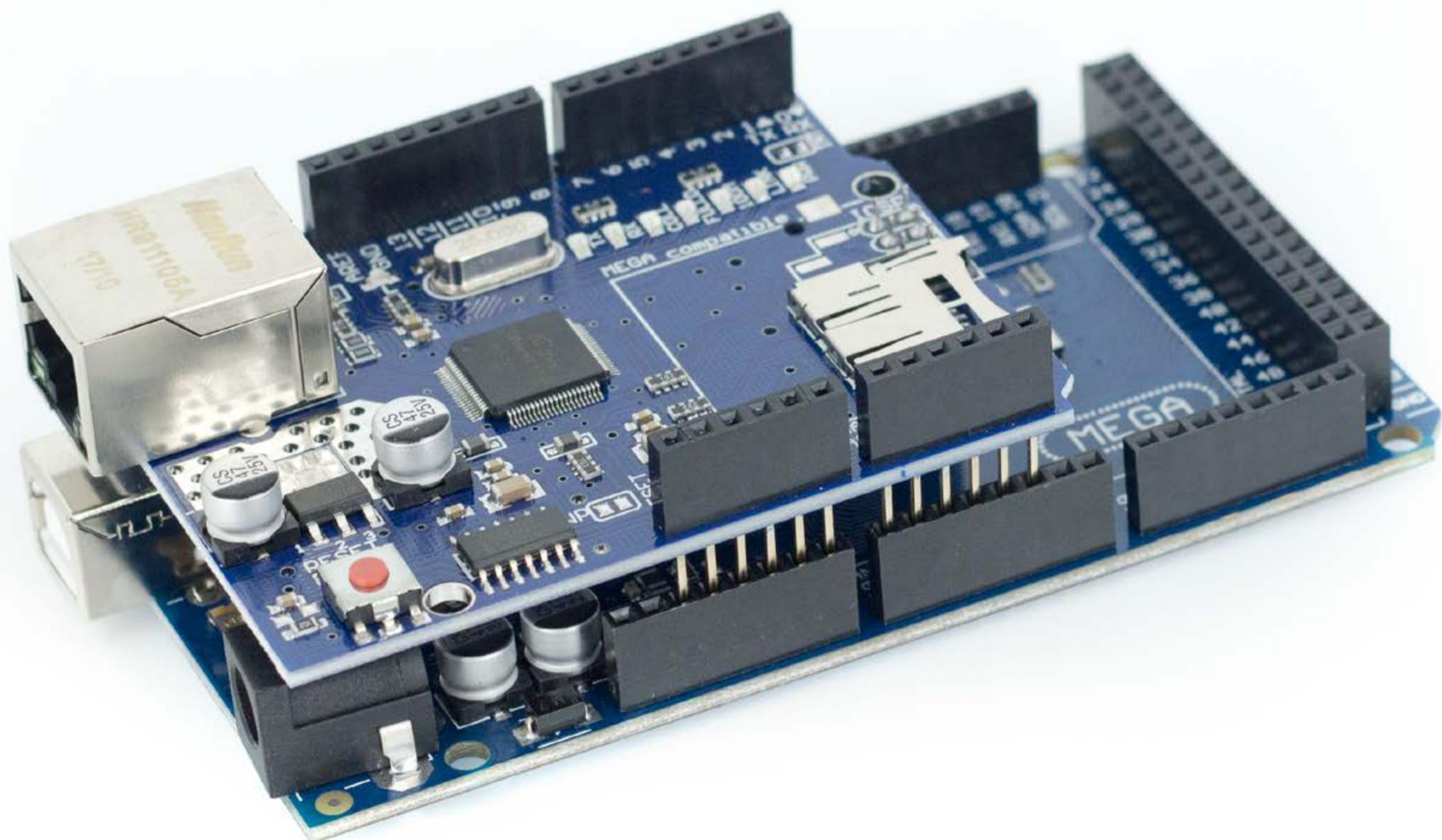
## O QUE VOCÊ PODE FAZER COM ARDUINO

A lista de possibilidades é praticamente infinita. Você pode automatizar sua casa, seu carro, seu escritório, criar um novo brinquedo, um novo equipamento ou melhorar um já existente. Tudo vai depender da sua criatividade.

Para isso, o Arduino possui uma quantidade enorme de [sensores](#) e componentes que você pode utilizar nos seus projetos. Grande parte do material utilizado está disponível em [módulos](#), que são pequenas placas que contém os sensores e outros componentes auxiliares como resistores, capacitores e leds.



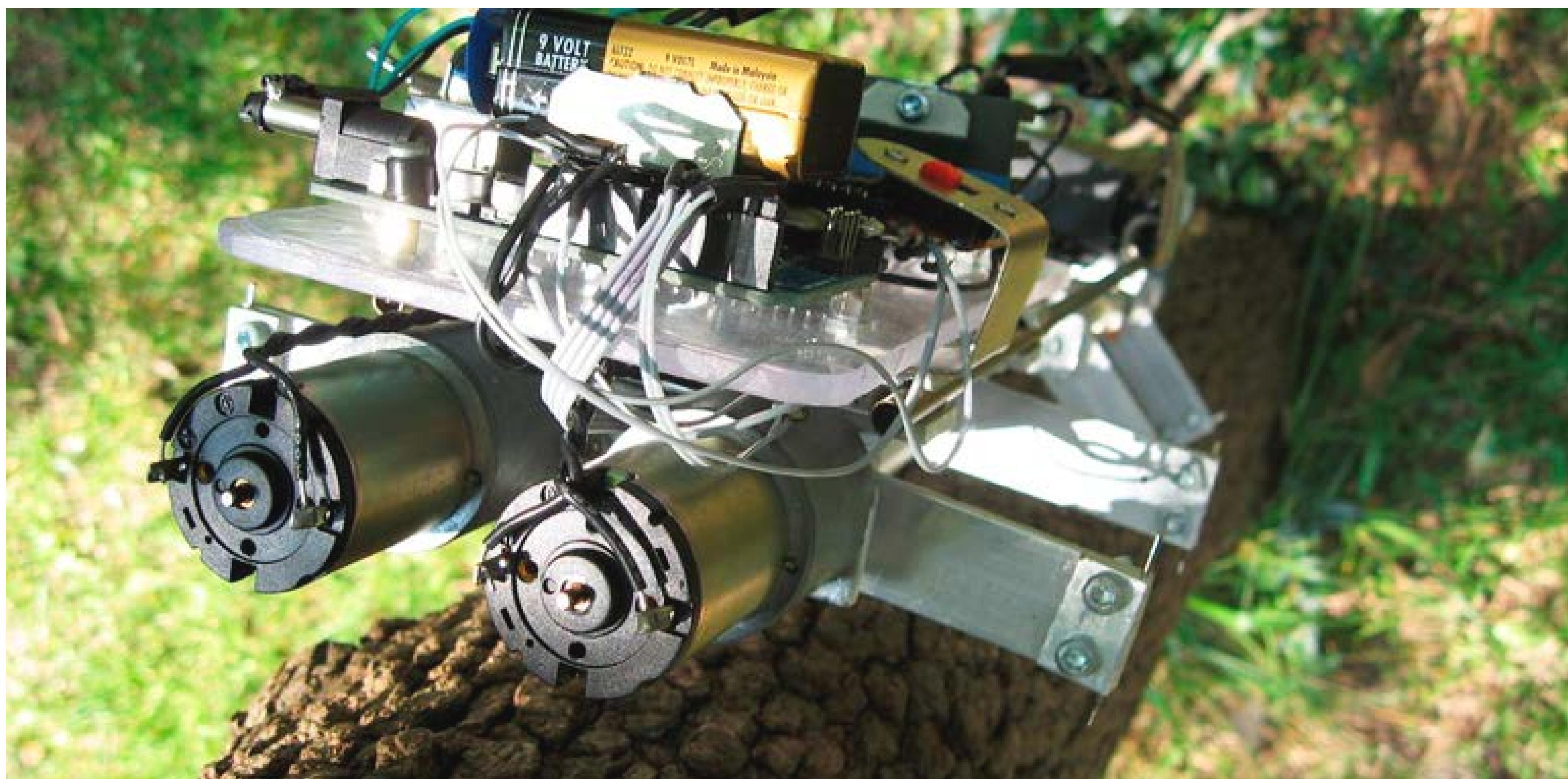
Existem também os chamados Shields, que são placas que você encaixa no Arduino para expandir suas funcionalidades. A imagem abaixo mostra um **Arduino Ethernet Shield** encaixado no Arduino Mega 2560. Ao mesmo tempo que permite o acesso à uma rede ou até mesmo à internet, mantém os demais pinos disponíveis para utilização, assim você consegue, por exemplo, utilizar os pinos para receber dados de temperatura e umidade de um ambiente, e consultar esses dados de qualquer lugar do planeta:



Para você ter uma idéia das possibilidades de criação com o Arduino, dê uma olhada nesses dois projetos (clique nas imagens para mais detalhes). O primeiro é de um tênis que se amarra sozinho.



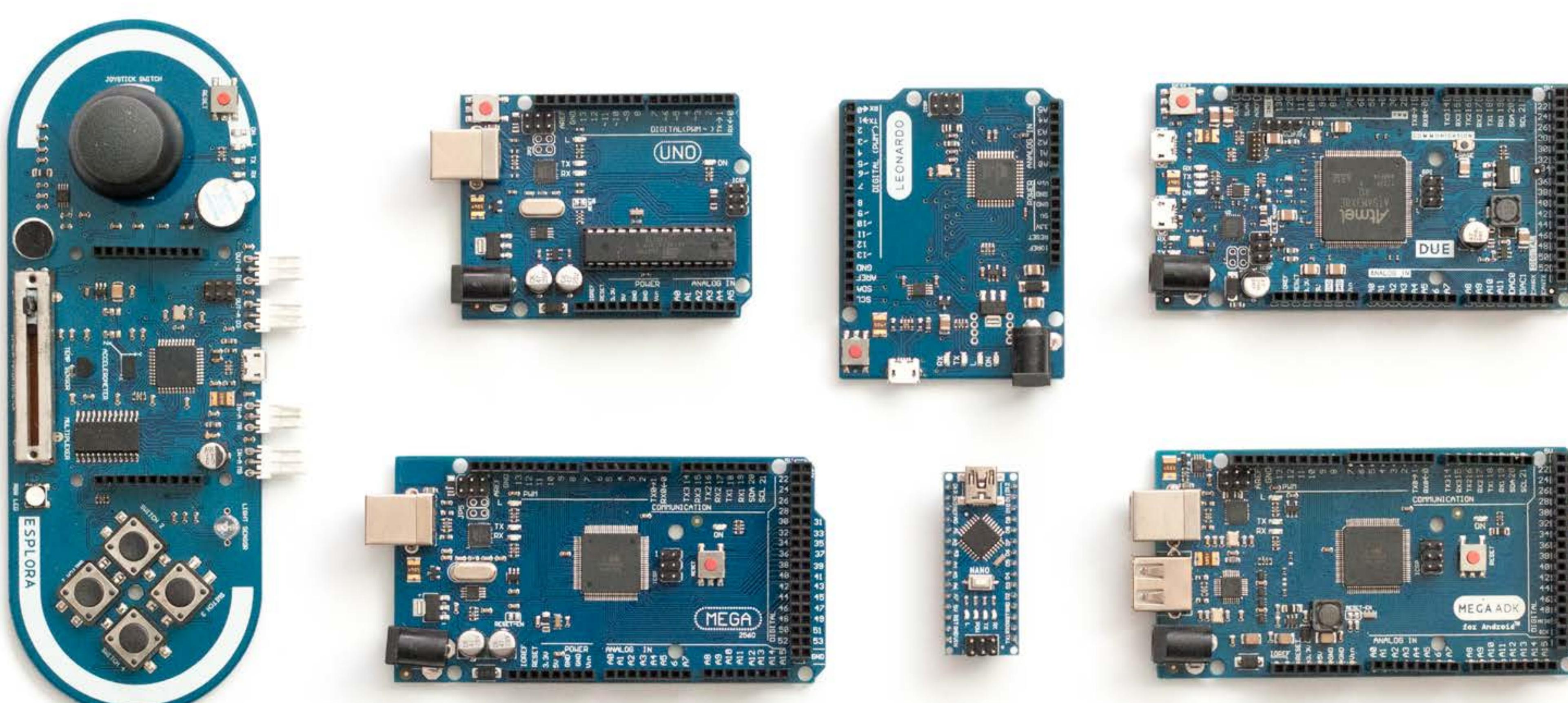
O outro é de um robô que sobe em árvores.



## MODELOS DE PLACAS ARDUINO

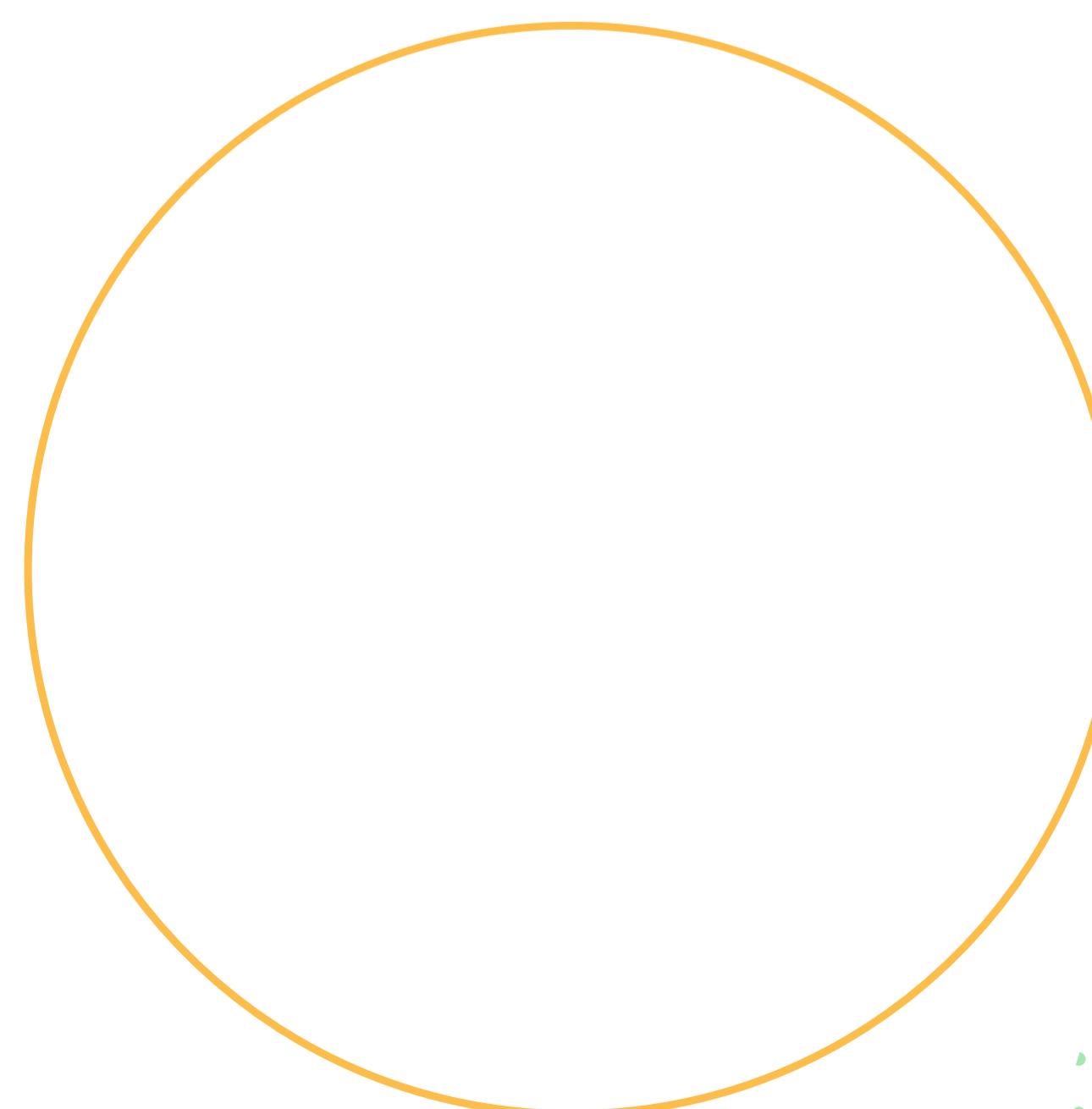
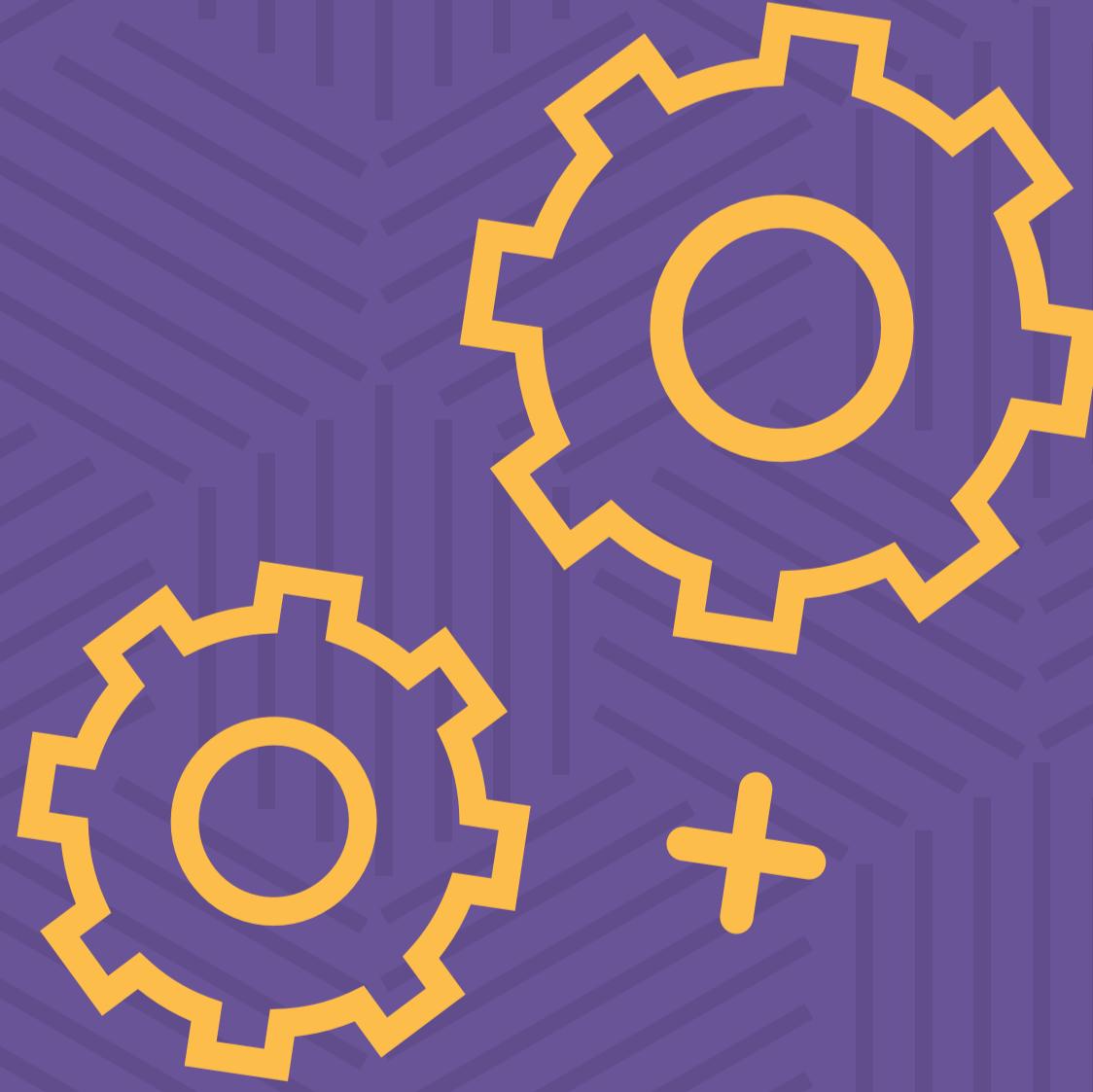
---

O tipo de placa que você vai utilizar depende muito do projeto a ser desenvolvido e o número de portas necessárias. As opções vão das mais comuns, como o [Arduino Uno](#) e suas 14 portas digitais e 6 analógicas, passando por placas com maior poder de processamento, como o [Arduino Mega](#), com microcontrolador ATmega2560 e 54 portas digitais, e o [Arduino Due](#), baseado em processador ARM de 32 bits e 512 Kbytes de memória:



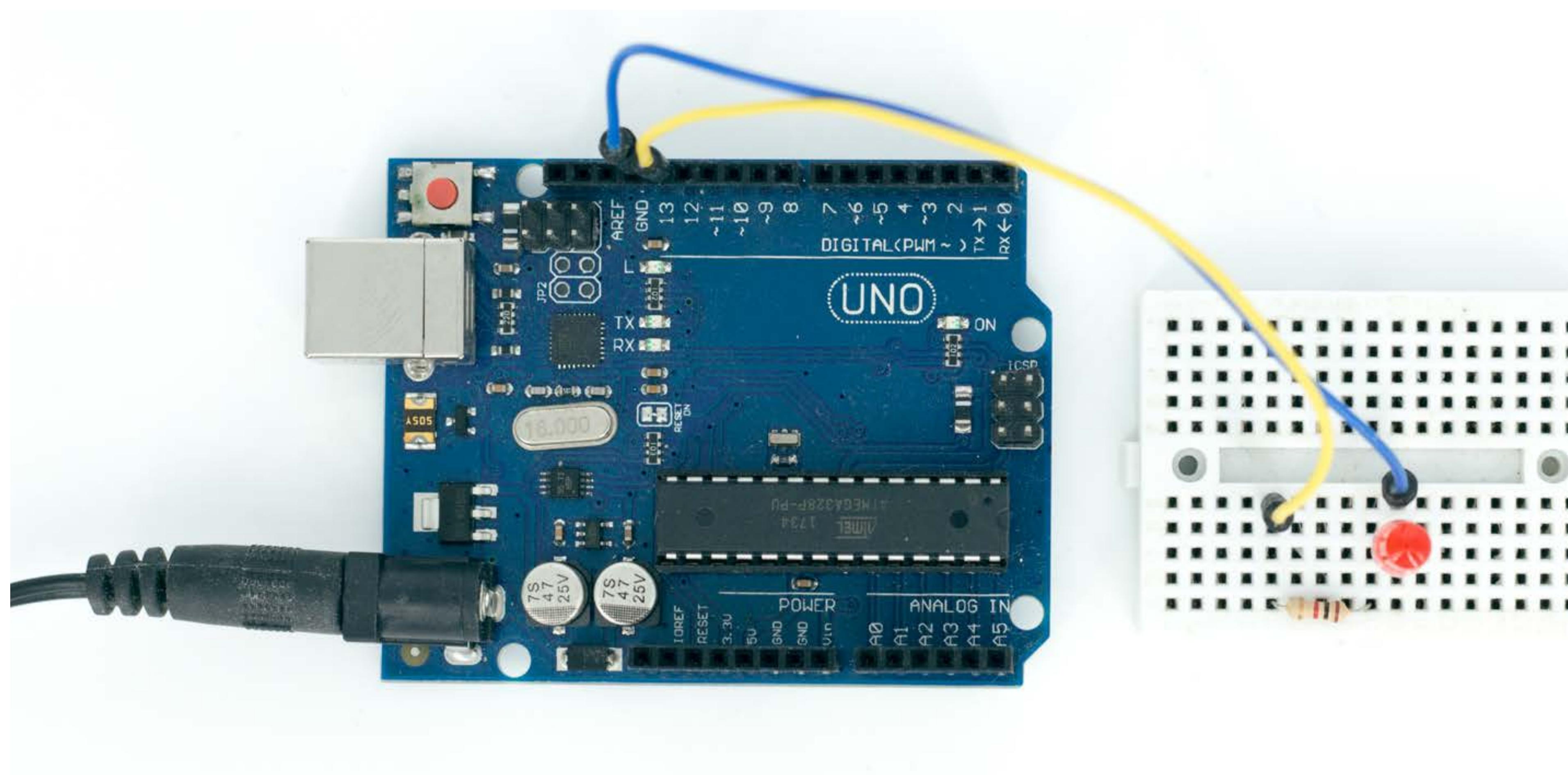


# PRIMEIROS PASSOS COM ARDUINO





Nesta seção iremos apresentar os primeiros passos com arduino, ou seja, tudo o que é necessário para que você comece a desenvolver utilizando essa plataforma. Iremos mostrar o processo de instalação da IDE Arduino, materiais necessários, estrutura de um programa e programa exemplo pisca LED.



## MATERIAIS NECESSÁRIOS

Para seguir com este tutorial e dar seus primeiros passos com arduino, você irá precisar dos seguintes componentes:

- [\*\*PLACA UNO R3 COM CABO USB\*\*](#)
- [\*\*PROTOBOARD\*\*](#)
- [\*\*LED VERMELHO\*\*](#)
- [\*\*RESISTOR 150 OHM\*\*](#)
- [\*\*JUMPERS MACHO/MACHO\*\*](#)
- [\*\*COMPUTADOR\*\*](#)

\* Caso você tenha apenas a placa Arduino poderá utilizar o LED já presente na placa

## UM POUCO DE TEORIA DE ELETROÔNICA

É importante entender um princípio básico na escolha do valor do resistor. O Arduino funciona com 5V em suas saídas digitais. Ou seja, quando ligamos um pino temos 5V e quando desligamos temos 0V. Mas é importante notar que o LED funciona apenas com 2V. Se colocarmos 5V em um LED provavelmente ele irá queimar. Para isso fazemos uso do resistor, que tem a função de diminuir a tensão do LED e limitar a corrente.

Essas informações podemos extrair do datasheet do LED. Datasheet é basicamente um documento de um componente eletrônico que contém as informações necessárias de funcionamento do respectivo componente. Analisando o [datasheet de um LED 5mm vermelho](#), podemos ver que ele funciona com uma tensão de 2V e corrente de 20mA. Agora precisamos encontrar um valor de resistor que fará o circuito chegar próximo de 2V e 20mA. Uma maneira fácil é utilizar uma calculadora online ou utilizar a seguinte fórmula:

$$V = R \times I \quad (\text{TENSÃO} = \text{RESISTÊNCIA} \times \text{CORRENTE})$$

Onde V é a tensão da fonte(5V porta Arduino) menos tensão do LED(2V), resultando em 3V. Assim teríamos:

$$3V = R \times 0,02A$$

$$R = 3V / 0,02A$$

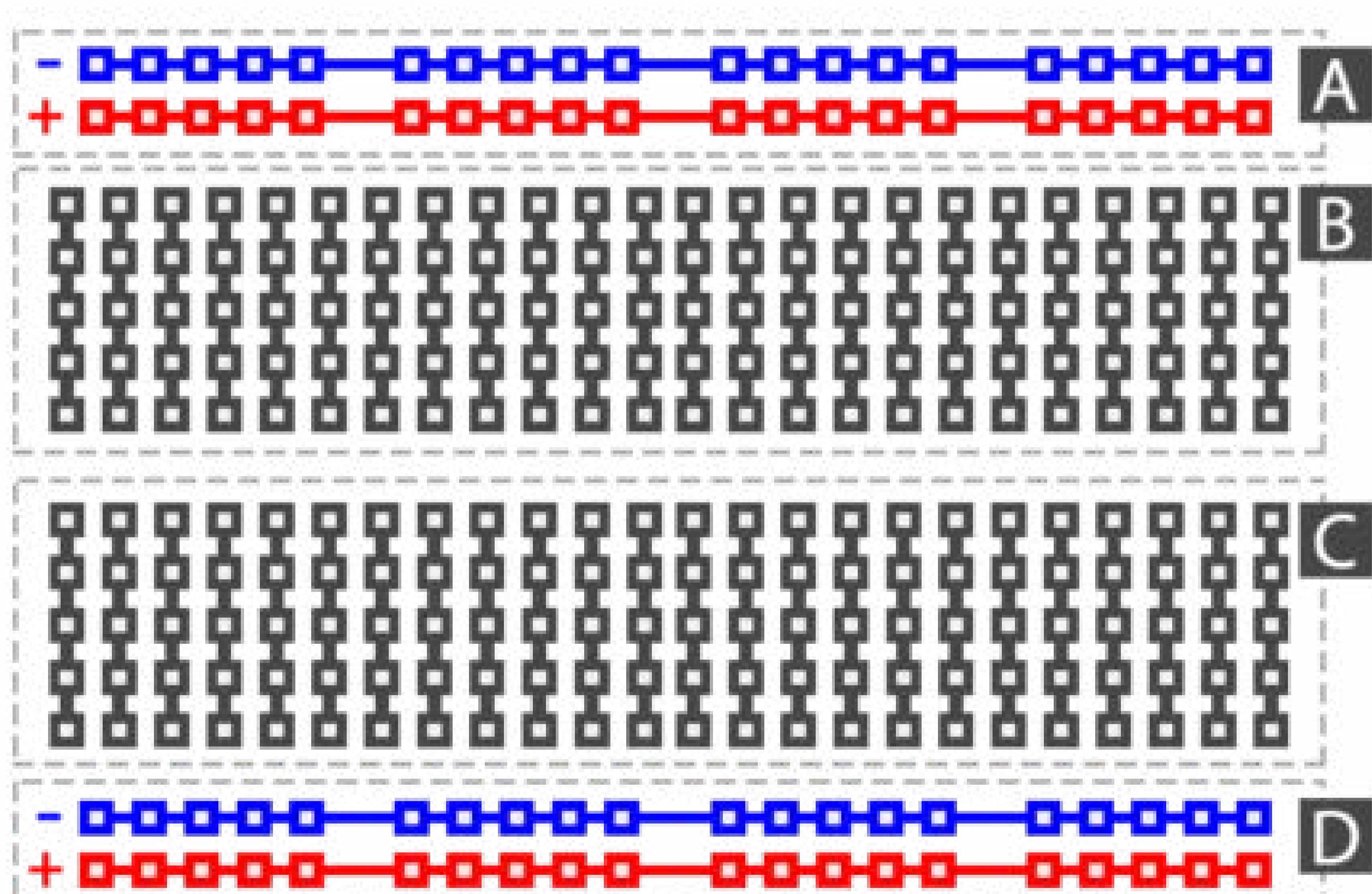
$$R = 150 \text{ OHM}$$

Caso não tenha um resistor de 150 ohms a mão, pode utilizar qualquer outro valor até de 1K. O LED irá acender, mas talvez com um brilho mais forte ou fraco dependendo do valor de resistência. Caso queira ir mais a fundo neste tema de eletrônica, estude sobre [Lei de Ohm](#) e [circuitos elétricos](#).

## COMO FUNCIONA UMA PROTOBOARD

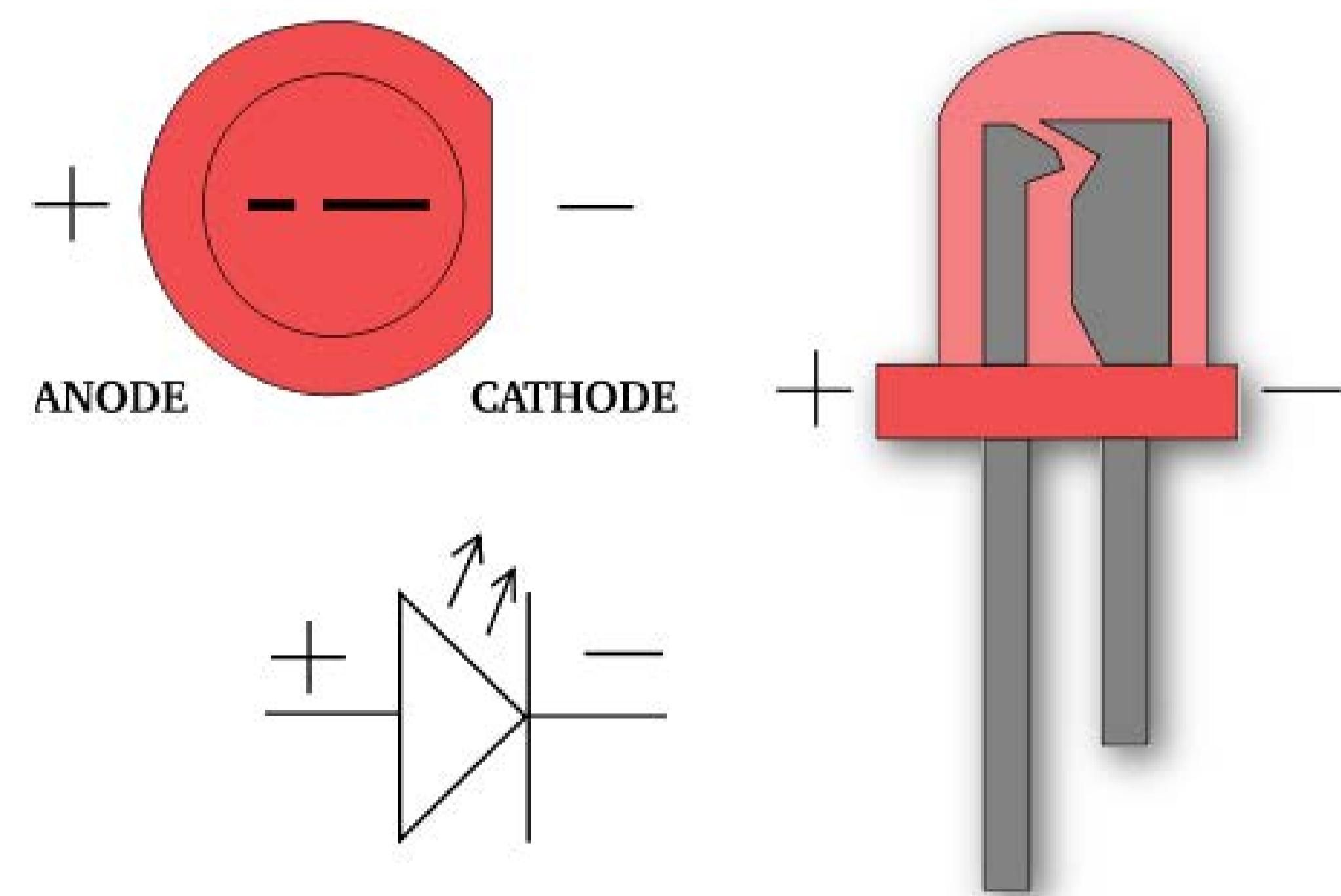
---

Uma protoboard serve para prototipagem de circuitos eletrônicos. Nas seções A e D geralmente são conectados VCC e GND. As seções B e C são utilizadas para conexão dos componentes eletrônicos. É de fácil utilização e segue o seguinte esquema de conexão interna:

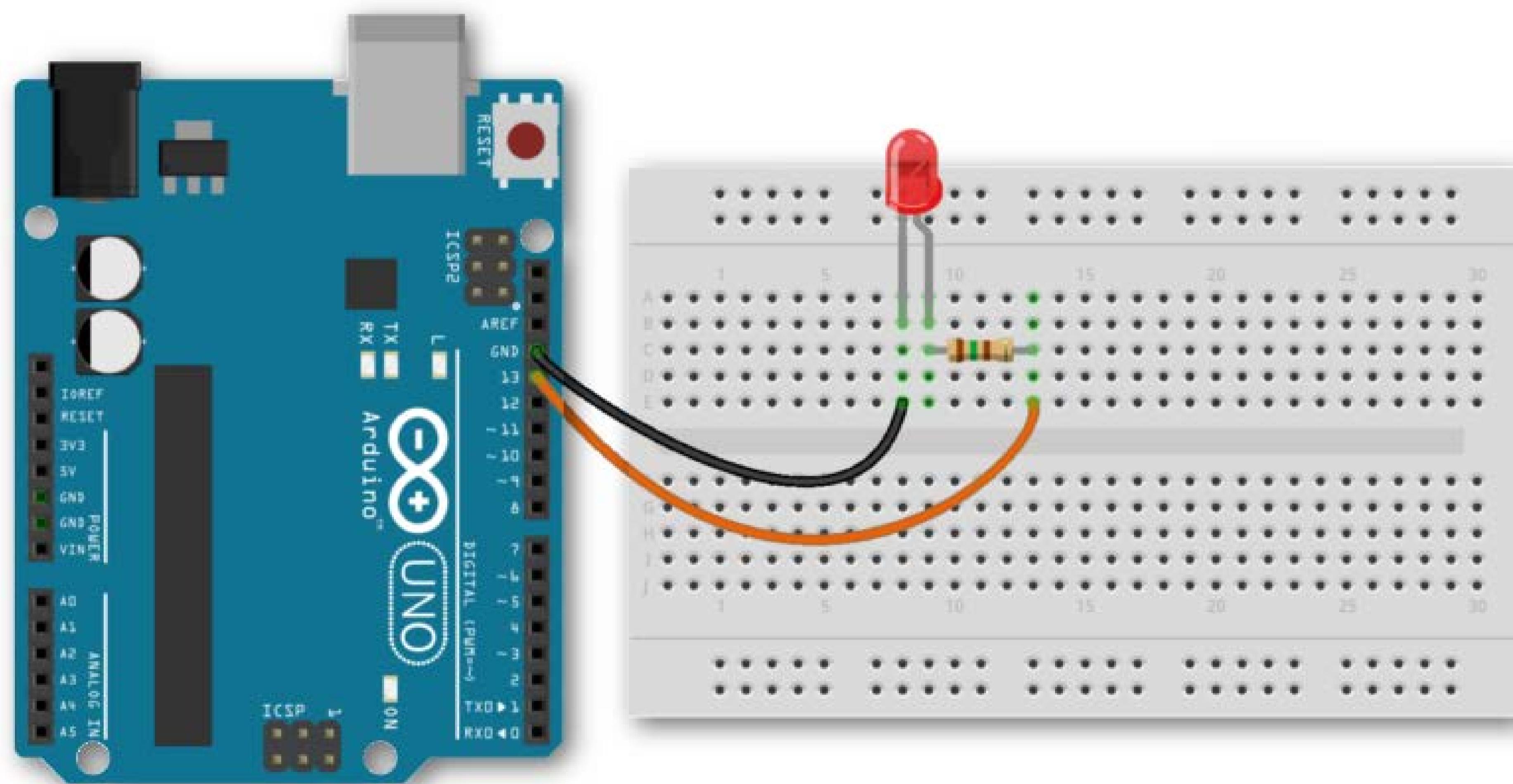


## MONTAGEM DO CIRCUITO

É importante notar que o LED possui polaridade, ou seja, terminal positivo (Anodo) e negativo (Catodo). O terminal maior do LED é o positivo e o menor é o negativo. Ou veja também pelo chanfro, que é o lado negativo.



O circuito deve ficar da seguinte maneira na protoboard:



fritzing

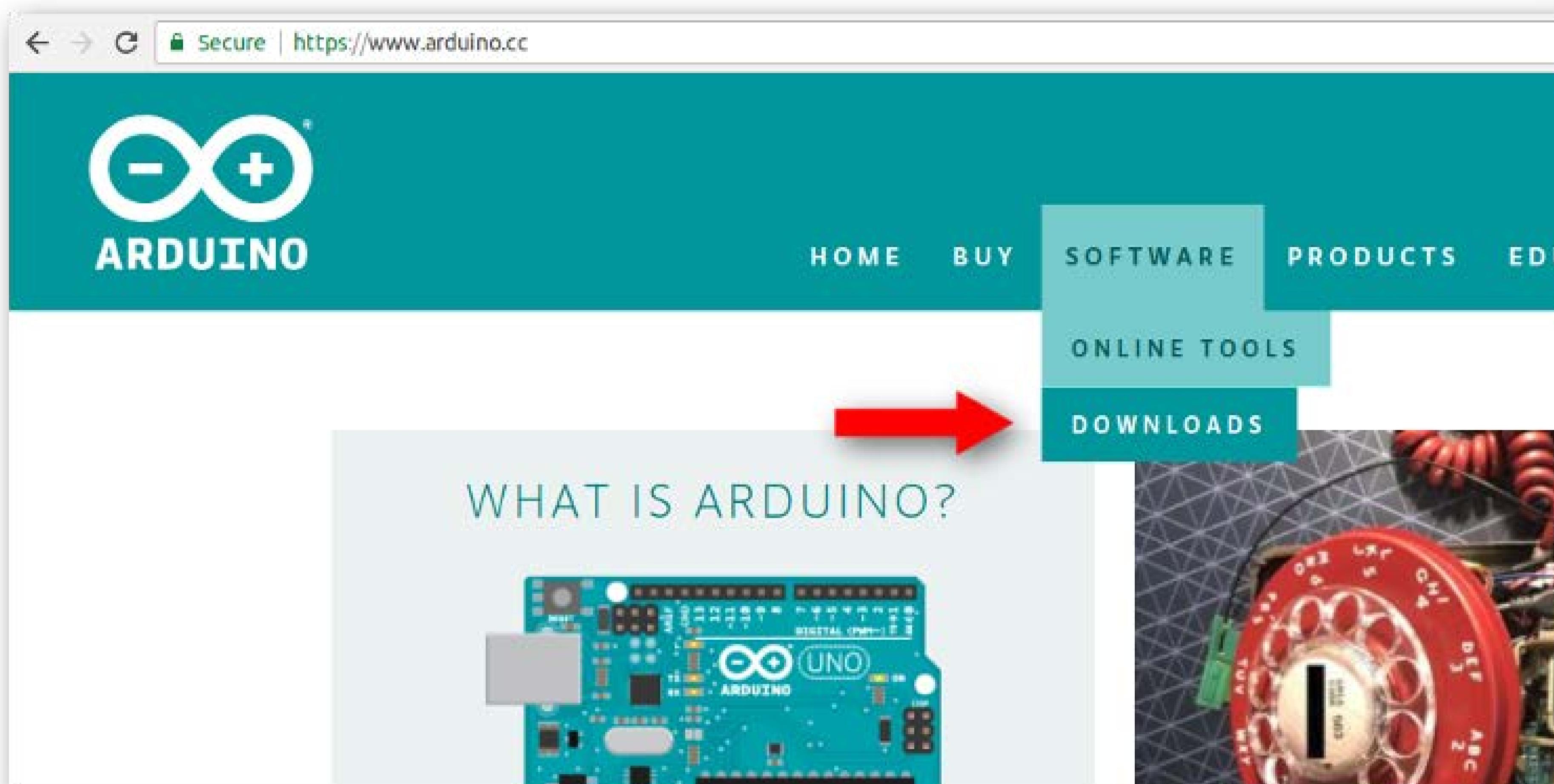
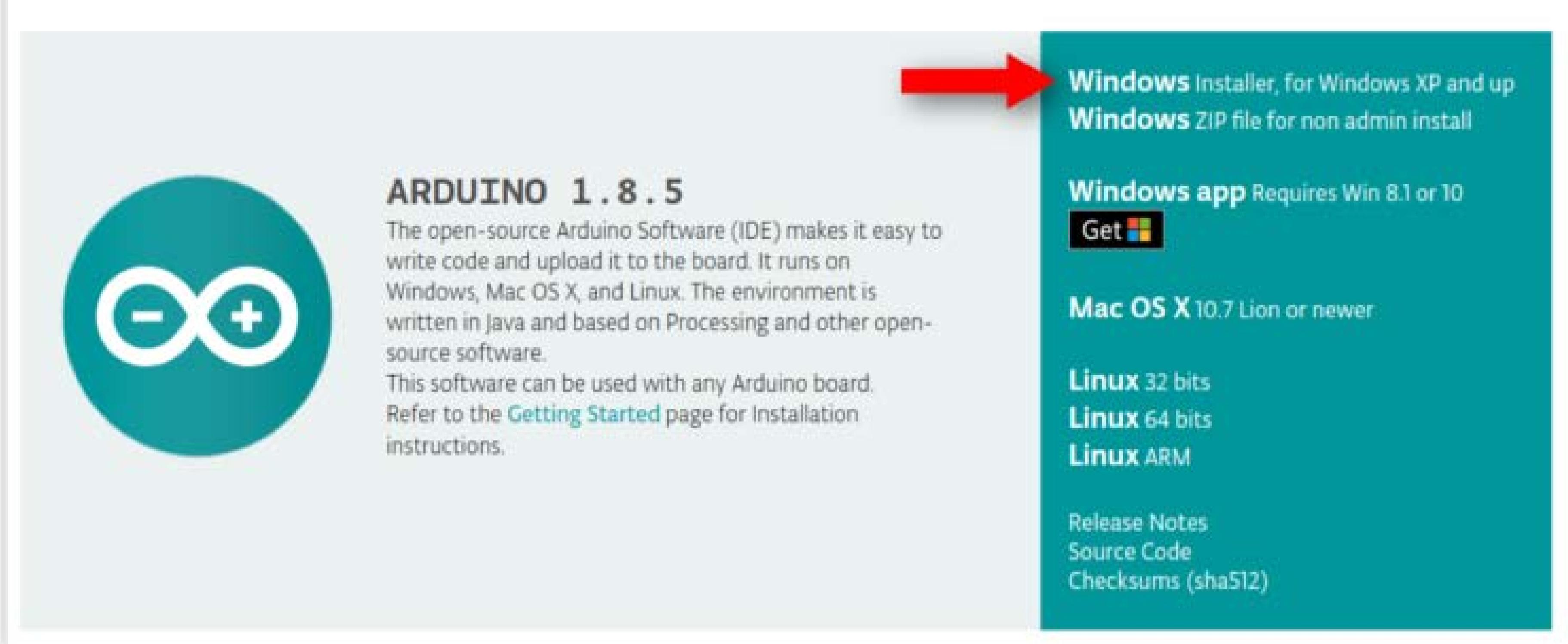
# BAIXANDO E INSTALANDO A IDE ARDUINO

Uma **IDE (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado)** é um programa de computador que possui as ferramentas necessárias para desenvolvimento de software. Basicamente possui um editor de código fonte e compilador.

Com a IDE Arduino podemos dar os primeiros passos com arduino e desenvolver programas, instalar bibliotecas adicionais e realizar a compilação e gravação dos programas na placa.

Para fazer o download da IDE Arduino entre no site oficial do Arduino na seção Software -> Downloads e escolha sua versão de sistema operacional(Windows, Linux, MacOS).

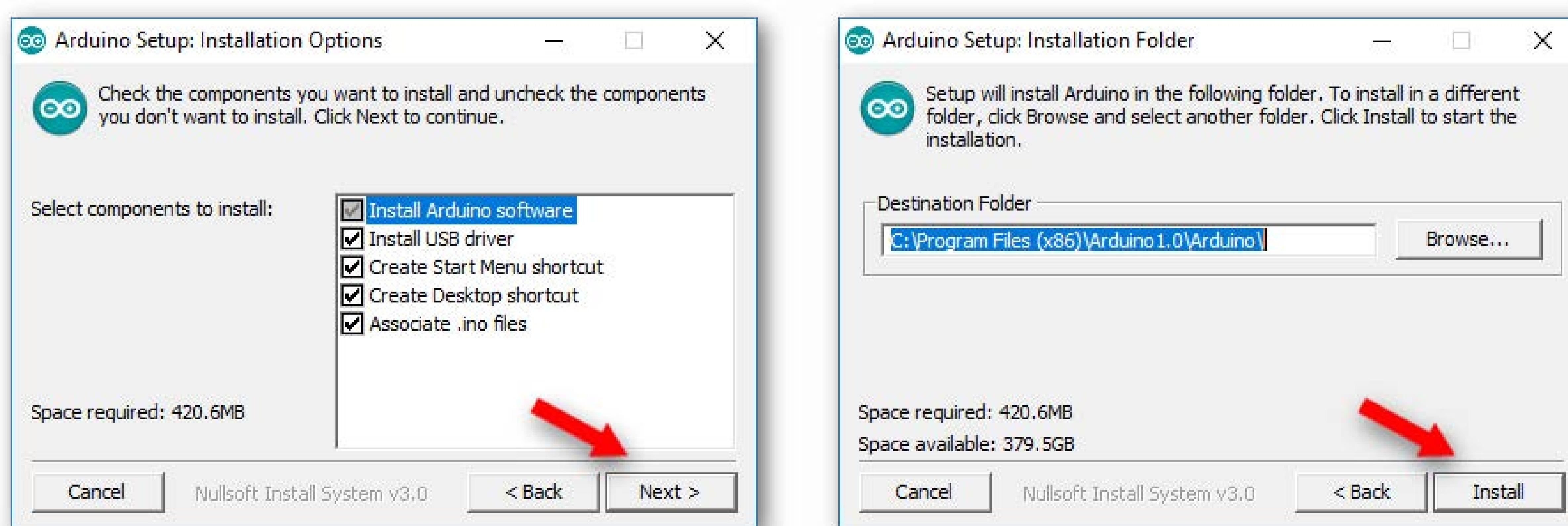
## Download the Arduino IDE



Se quiser fazer uma doação para o software Arduino este é o momento. Se não, basta clicar em *Just Download*.



Execute o arquivo instalador '.exe' e siga as instruções de instalação. Certifique-se de que todos os componentes na tela abaixo estejam selecionados. Após a instalação, abra a IDE Arduino pelo atalho criado na sua área de trabalho.



## ESTRUTURA DE UM PROGRAMA ARDUINO

Você não precisa ser um expert em linguagem C para programar com Arduino. Ao abrir a IDE Arduino você se depara com uma estrutura padrão de programa contendo as funções `setup()` e `loop()`.

- **setup()** - É nessa parte do programa que você configura as opções iniciais do seu programa: os valores iniciais de uma variável, se uma porta será utilizada como entrada ou saída, mensagens para o usuário, etc. Essa função irá executar apenas uma vez no início do programa.
- **loop()** - Diferente da função `setup()`, essa parte do programa repete uma estrutura de comandos de forma contínua ou até que alguma comando de “parar” seja enviado ao Arduino.

Vamos ver exatamente como isso funciona, levando em consideração o programa abaixo, que acende e apaga o led embutido na placa Arduino em intervalos de 1 segundo:

```
//Programa : Piscia Led Arduino
//Autor : FILIPEFLOP

void setup()
{
    //Define a porta do led como saida
    pinMode(13, OUTPUT);
}

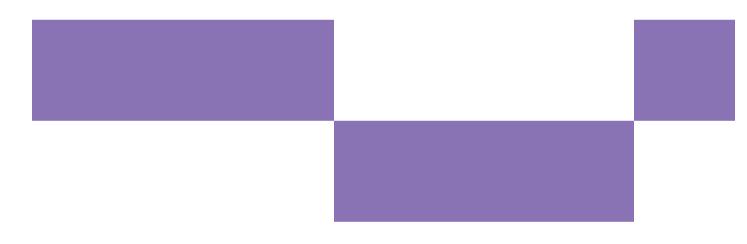
void loop()
{
    //Acende o led
    digitalWrite(13, HIGH);

    //Aguarda o intervalo especificado
    delay(1000);

    //Apaga o led
    digitalWrite(13, LOW);

    //Aguarda o intervalo especificado
    delay(1000);
}
```

**\_VOCÊ NÃO PRECISA SER UM EXPERT EM LINGUAGEM C PARA PROGRAMAR COM ARDUINO.**



## EXEMPLO PISCA LED

Vimos acima um programa exemplo para piscar nosso LED, o famoso “Hello World” da eletrônica e um dos primeiros passos com arduino. O exemplo Blink LED.

A primeira coisa que fazemos no início do programa é colocar uma pequena observação sobre o nome do programa, sua função e quem o criou:

```
// Programa : Pisca Led Arduino
// Autor : FILIPEFLOP
```

Comece uma linha com barras duplas ( // ) e tudo o que vier depois dessa linha será tratado como um comentário. Uma das boas práticas de programação é documentar o seu código por meio das linhas de comentário. Com elas, você pode inserir observações sobre como determinada parte do programa funciona ou o que significa aquela variável **xyz** que você criou. Isso será útil não só para você, se precisar alterar o código depois de algum tempo, como também para outras pessoas que utilizarão o seu programa.

Após os comentários, vem a estrutura do **setup()**. É nela que definimos que o pino 13 do Arduino será utilizado como saída.

```
void setup()
{
    //Define a porta do led como saida
    pinMode(13, OUTPUT);
}
```

Por último, temos o **loop()**, que contém as instruções para acender e apagar o led, e também o intervalo entre essas ações:

```
void loop()
{
    //Acende o led
    digitalWrite(13, HIGH);

    //Aguarda o intervalo especificado
    delay(1000);

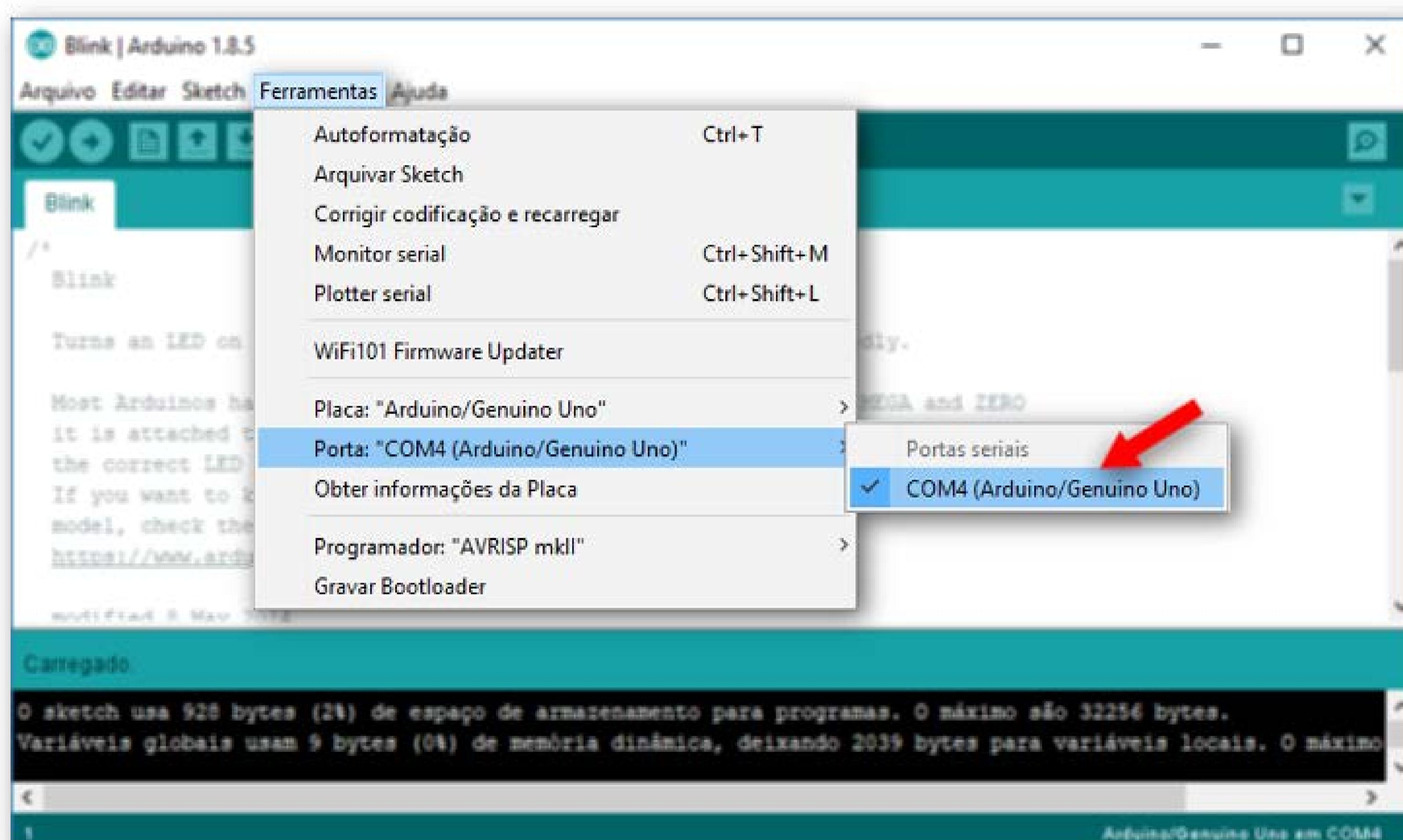
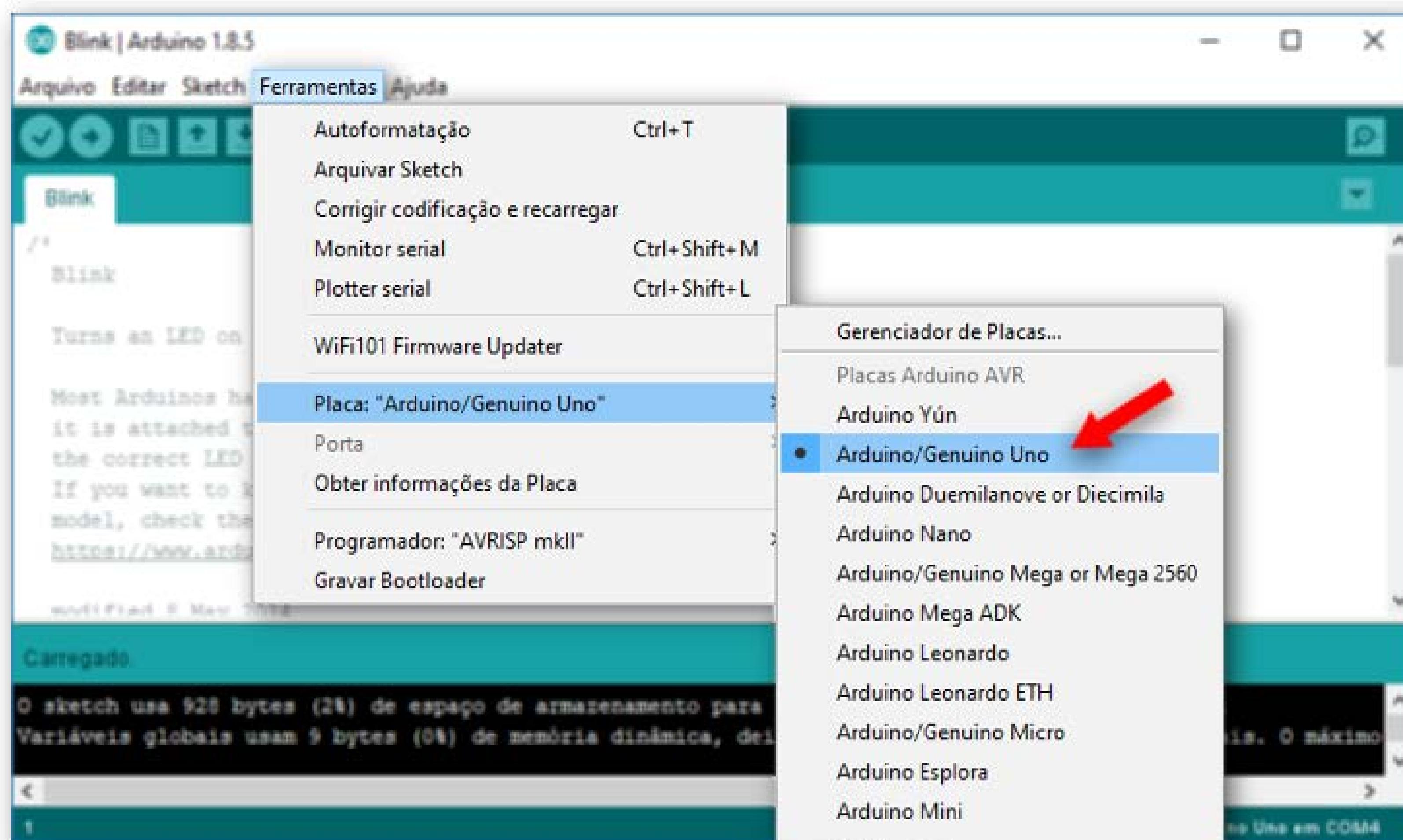
    //Apaga o led
    digitalWrite(13, LOW);

    //Aguarda o intervalo especificado
    delay(1000);
}
```

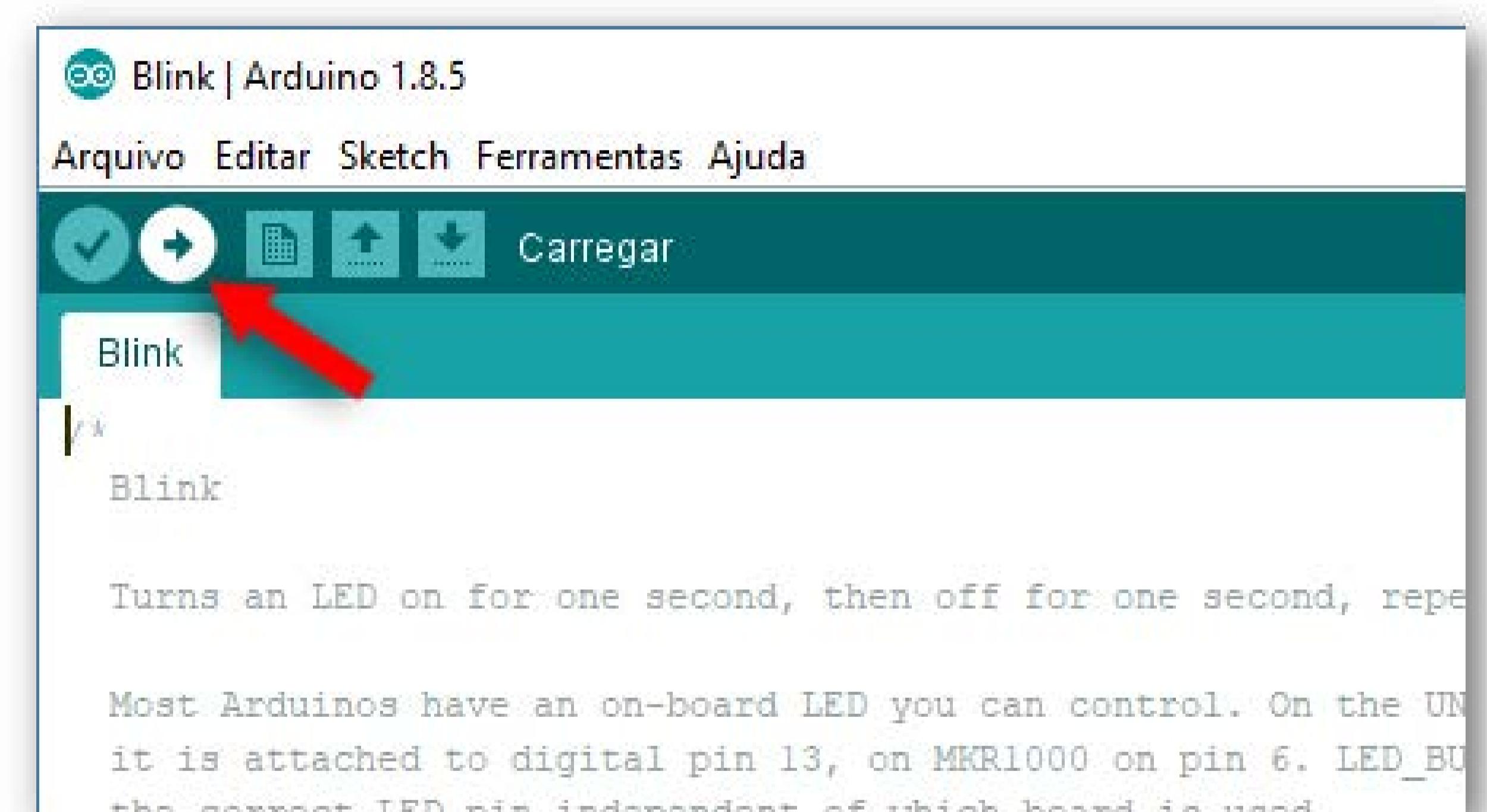
A linha do código contendo **digitalWrite(13, HIGH)** coloca a porta 13 em nível alto (HIGH, ou 1), acendendo o led embutido na placa. O comando **delay(1000)**, especifica o intervalo, em milissegundos, no qual o programa fica parado antes de avançar para a próxima linha.

O comando **digitalWrite(13, LOW)**, apaga o led, colocando a porta em nível baixo (LOW, ou 0), e depois ocorre uma nova parada no programa, e o processo é então reiniciado.

Quando o código estiver pronto para ser carregado na placa, conecte a placa Arduino no seu computador, entre no menu ferramentas, escolha o modelo da placa e a porta na qual a mesma está conectada:

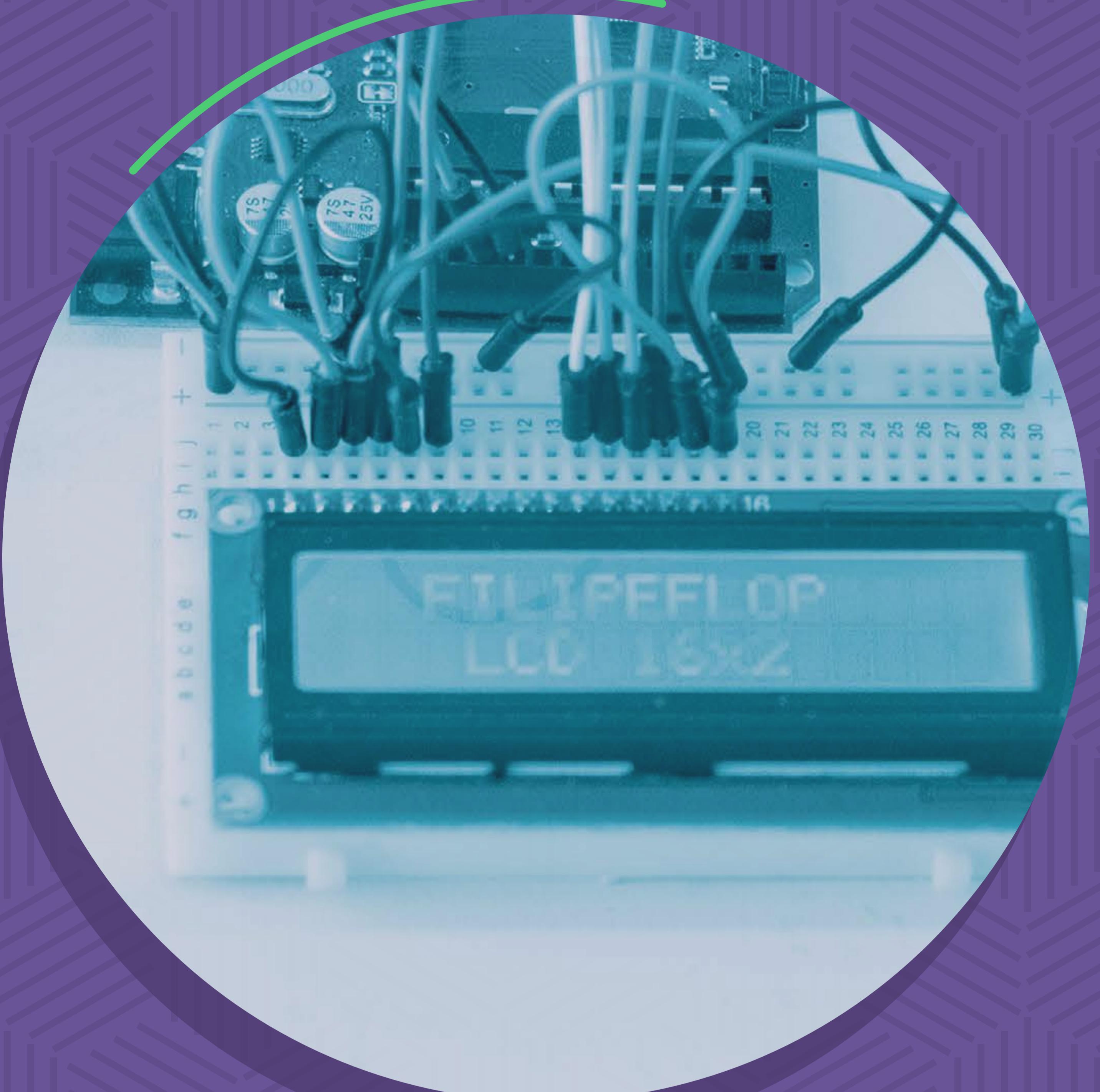
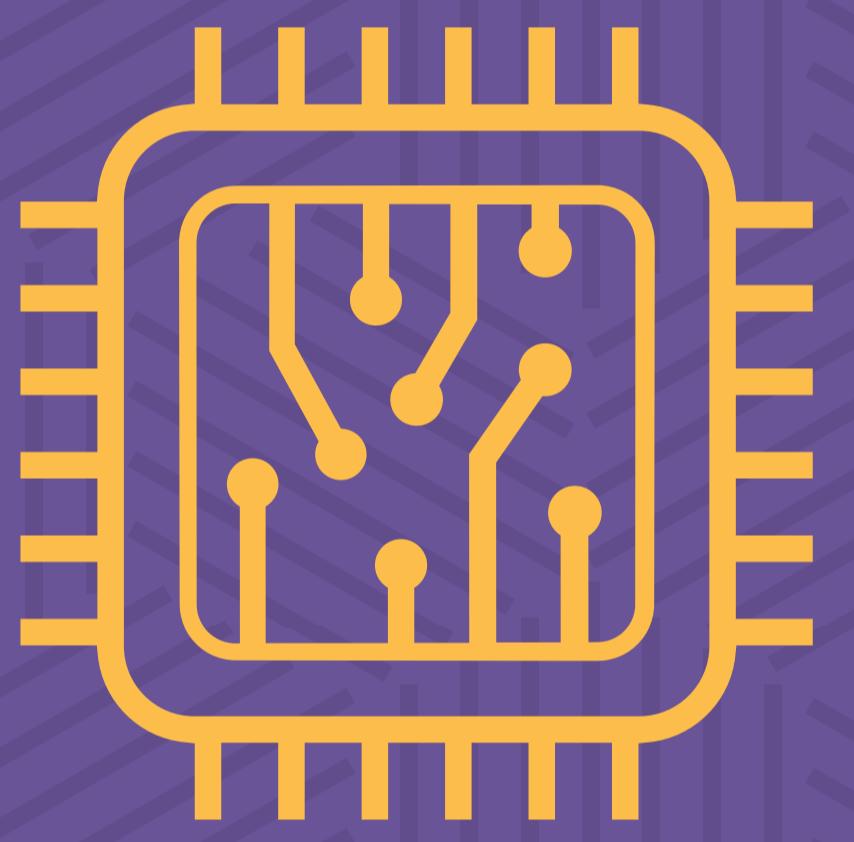


Se estiver tudo OK, clique no botão carregar. Isso irá gravar o programa na placa.



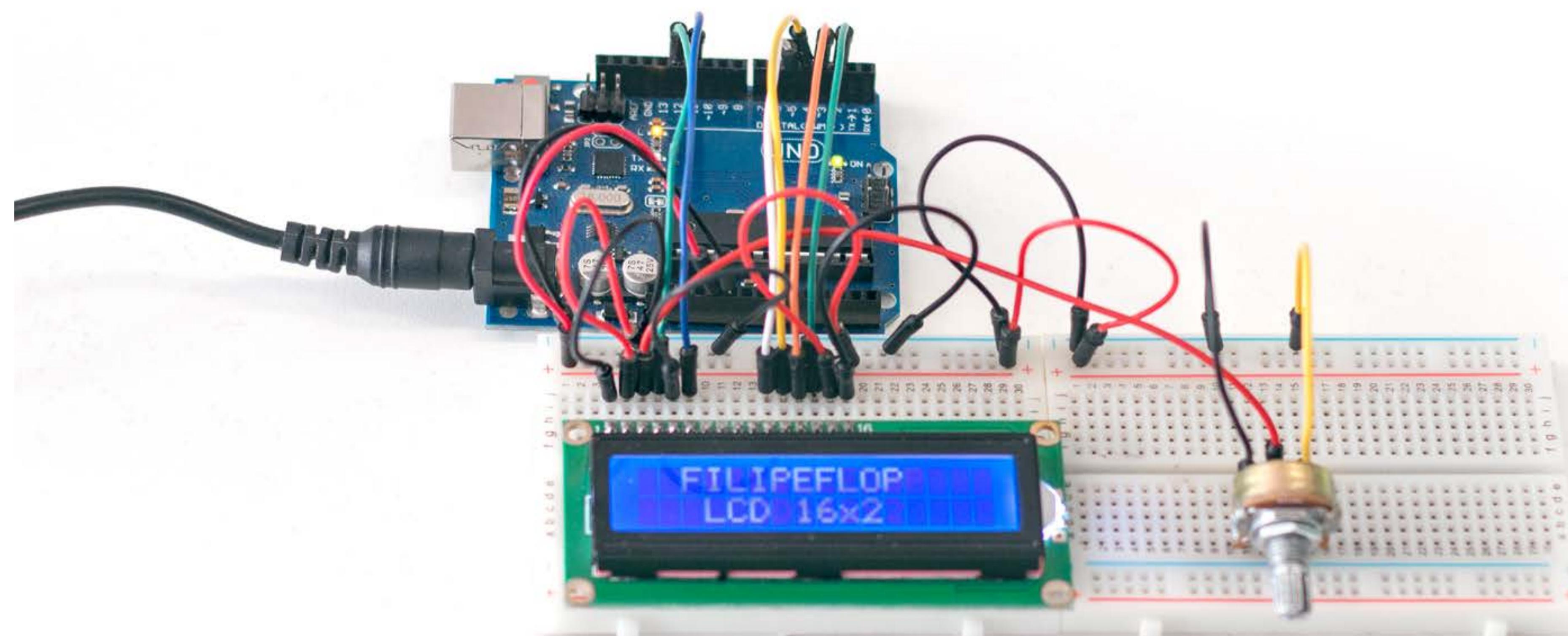
Caso não apareça nenhum erro, você deverá ver o LED piscando em um intervalo de 1 segundo.

# CONTROLANDO UM LCD 16X2 COM ARDUINO





Nesta seção você vai encontrar alguns testes básicos do [display LCD 16x2](#) com arduino, um display muito comum com controlador HD44780, que se adapta aos mais diversos projetos, podendo ser usado com vários modelos de placas e microcontroladores como Arduino, Raspberry Pi, PIC, etc.



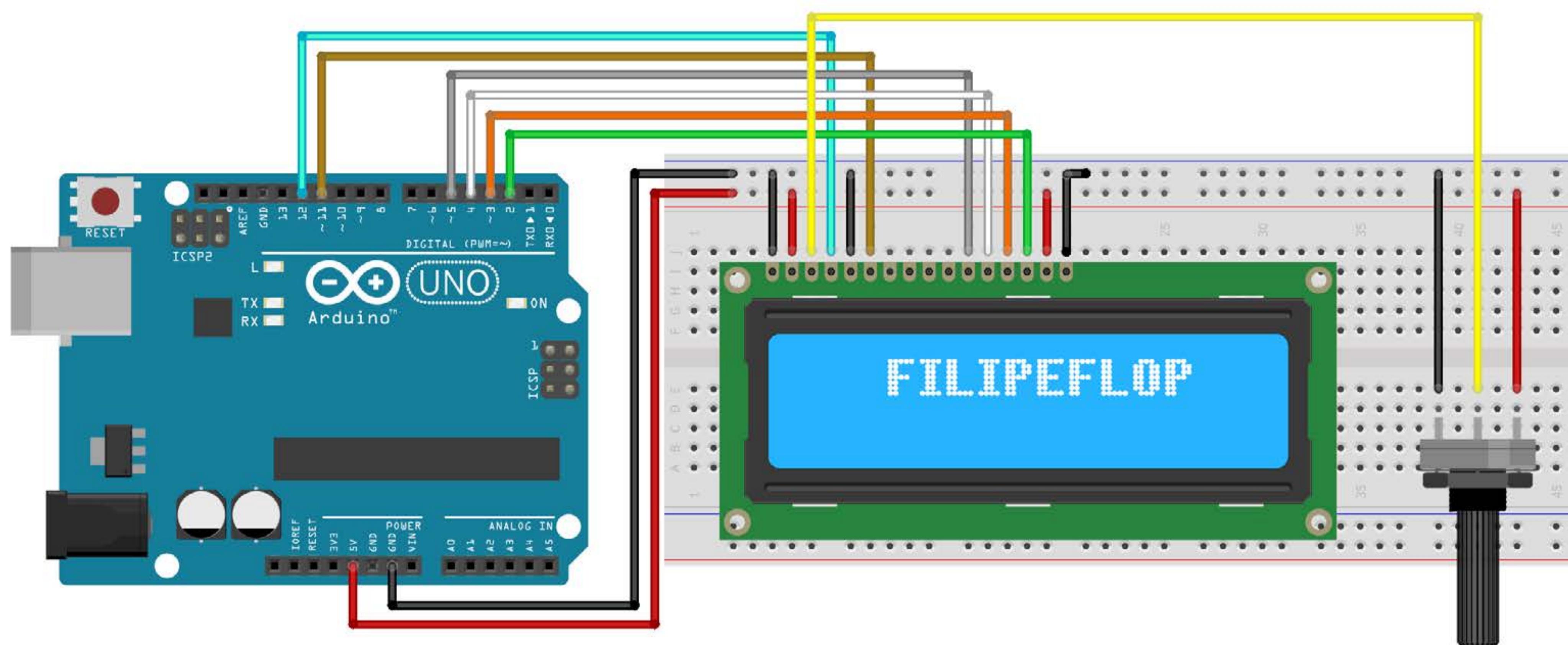
Esse **display LCD** tem 16 colunas e 2 linhas, com backlight (luz de fundo) azul e letras na cor branca. Para conexão, são 16 pinos, dos quais usamos 12 para uma conexão básica, já incluindo as conexões de alimentação (pinos 1 e 2), backlight (pinos 15 e 16) e contraste (pino 3).

Conexões LCD 16x2 - HD44780		
Pino LCD	Função	Ligaçāo
1	Vss	GND
2	Vdd	Vcc 5V
3	V0	Pino central potenciômetro
4	RS	Pino 12 Arduino
5	RW	GND
6	E	Pino 11 Arduino
7	D0	Não conectado
8	D1	Não conectado
9	D2	Não conectado
10	D3	Não conectado
11	D4	Pino 5 Arduino
12	D5	Pino 4 Arduino
13	D6	Pino 3 Arduino
14	D7	Pino 2 Arduino
15	A	Vcc 5V
16	K	GND

## CONEXÃO DISPLAY 16X2 COM ARDUINO

Na conexão do display ao [Arduino Uno](#) vamos utilizar apenas 4 pinos de dados (pinos digitais 2, 3, 4 e 5), e 2 pinos de controle (pinos digitais 11 e 12).

Para o ajuste do contraste, usamos um [potenciômetro](#) de 100K, mas você pode testar com outros valores como 10K ou 50K, por exemplo.



Se preferir, você também pode utilizar um potenciômetro para regular a luz de fundo, nos pinos 15 e 16 do display. Outra opção é usar um resistor em um desses pinos.

## PROGRAMA DE CONTROLE LCD

O controle desse display pode ser feito utilizando-se a biblioteca **LiquidCrystal**, já embutida na IDE do Arduino.

No início do programa (linha 8), definimos os pinos que serão utilizados pelo displays, nesse formato:

## LiquidCrystal lcd(<pino RS>, <pino enable>, <pino D4>, <pino D5>, <pino D6>, <pino D7>)

No setup, inicializamos o display definindo o número de colunas e linhas com o comando **lcd.begin(16,2)**.

```
//Programa: Teste de Display LCD 16 x 2
//Autor: FILIPEFLOP

//Carrega a biblioteca LiquidCrystal
#include <LiquidCrystal.h>

//Define os pinos que serão utilizados para ligação ao display
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

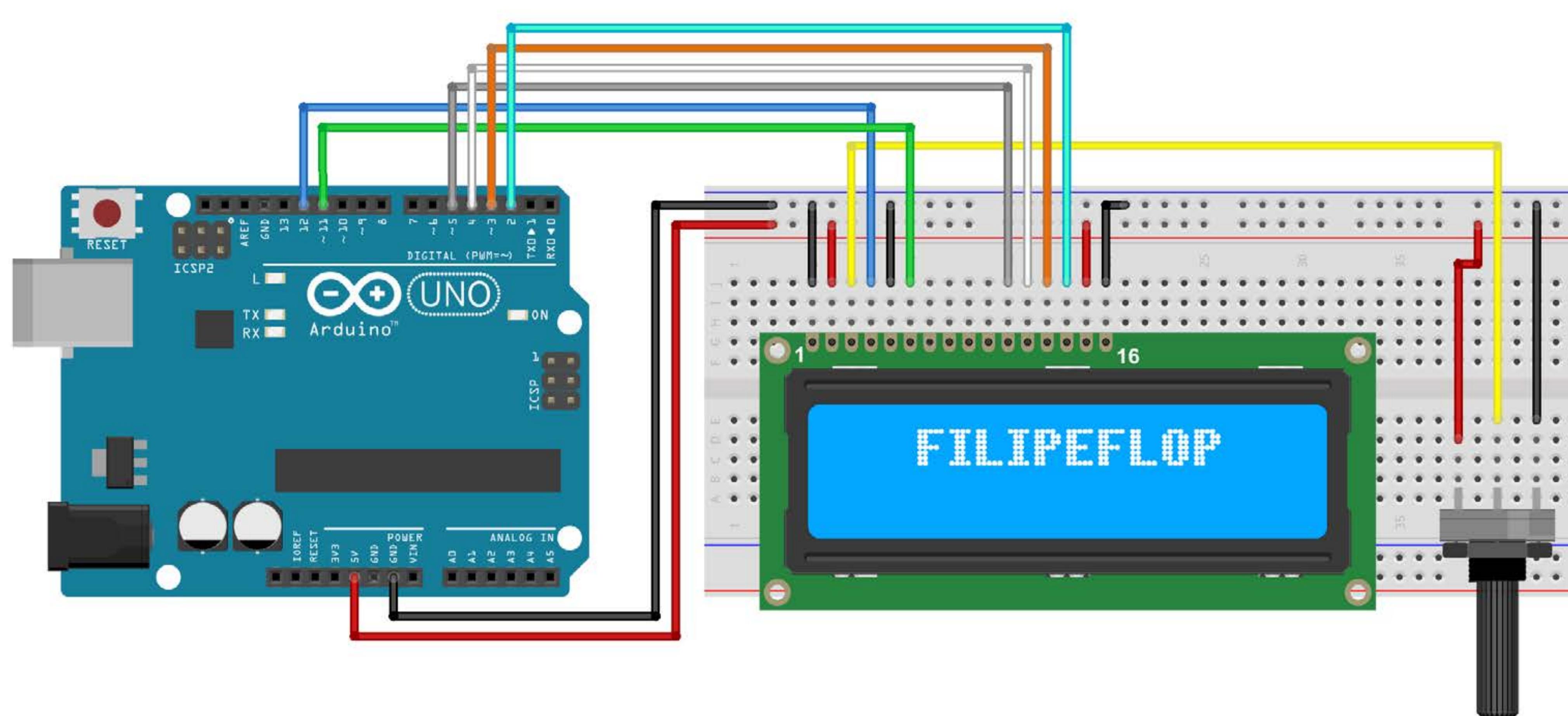
void setup()
{
    //Define o número de colunas e linhas do LCD
    lcd.begin(16, 2);
}

void loop()
{
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(3, 0);
    //Envia o texto entre aspas para o LCD
    lcd.print("FILIPEFLOP");
    lcd.setCursor(3, 1);
    lcd.print(" LCD 16x2");
    delay(5000);

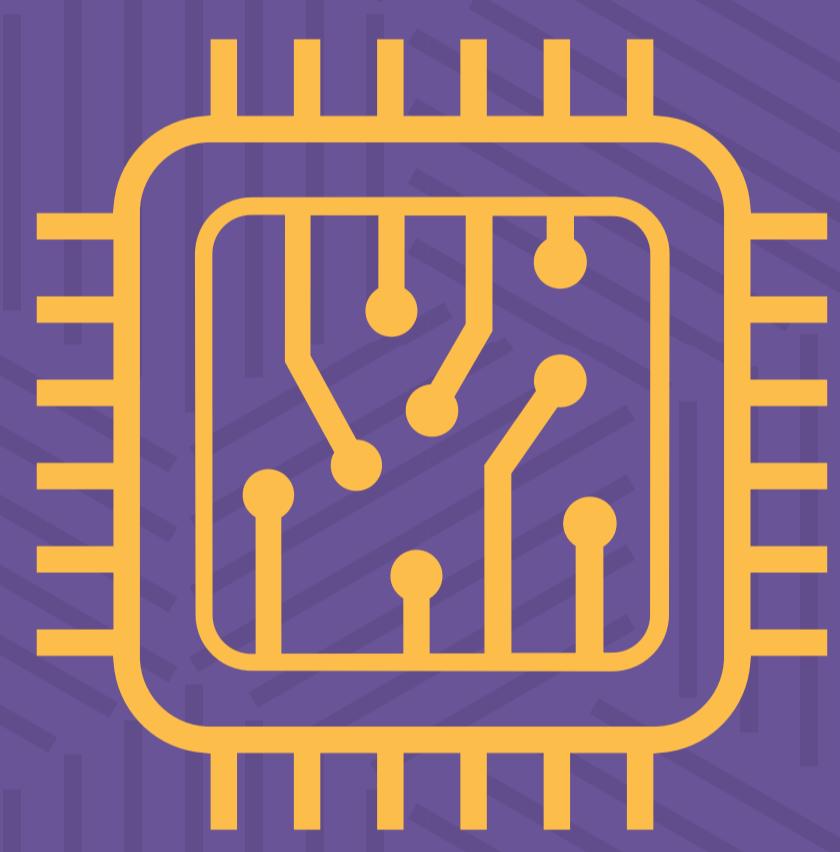
    //Rolagem para a esquerda
    for (int posicao = 0; posicao < 3; posicao++)
    {
        lcd.scrollDisplayLeft();
        delay(300);
    }

    //Rolagem para a direita
    for (int posicao = 0; posicao < 6; posicao++)
    {
        lcd.scrollDisplayRight();
        delay(300);
    }
}
```

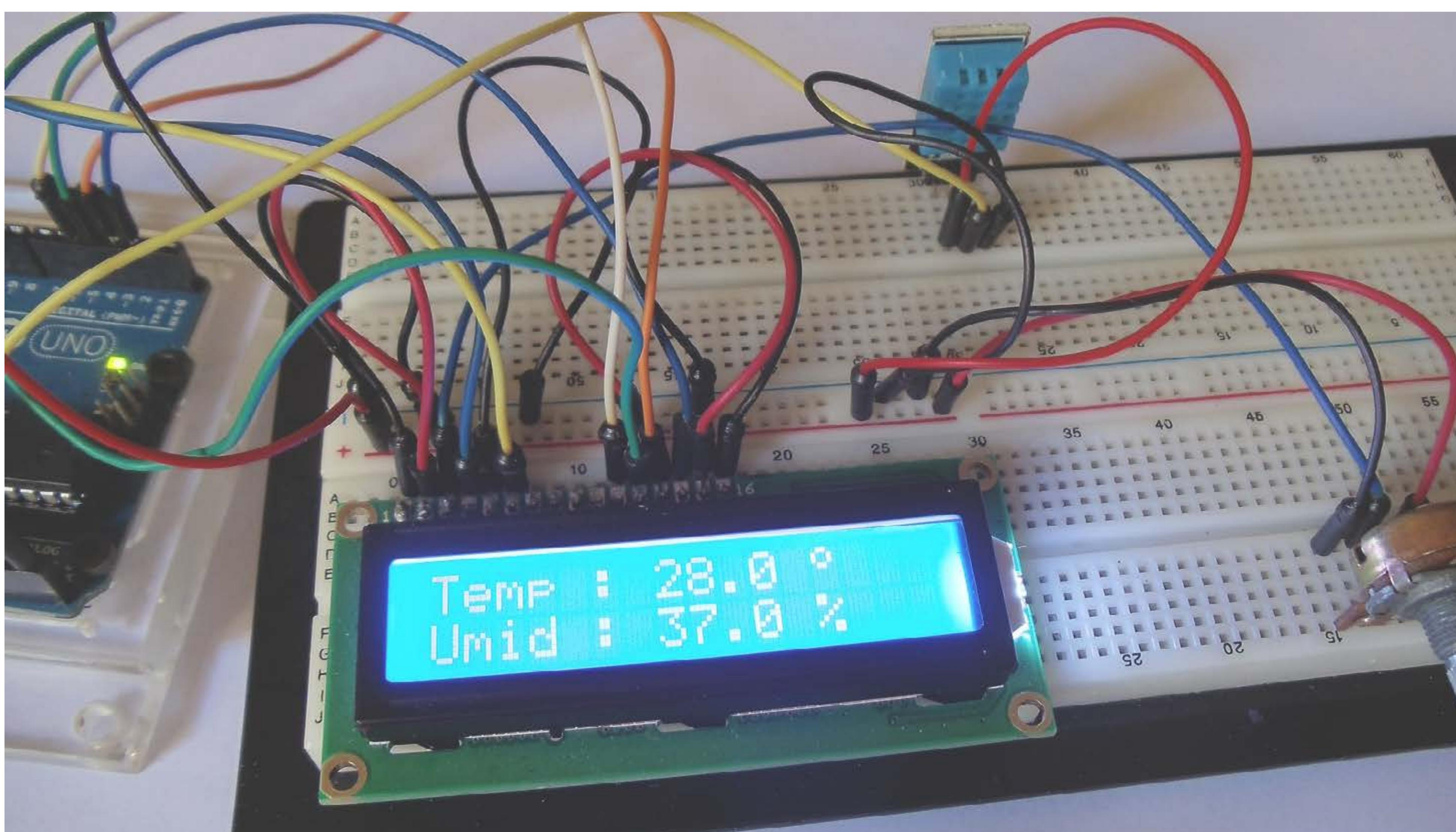
O programa acima posiciona o texto na posição desejada utilizando o comando **Lcd.setCursor()**, e imprime a string na tela usando **Lcd.print("Texto")**. Depois de 5 segundos, são utilizados os comandos **scrollDisplayLeft()** e **scrollDisplayRight()** para “mover” os caracteres para a esquerda e para a direita, respectivamente.



# MOSTRANDO A TEMPERATURA NO LCD 16x2 COM O SENSOR DHT11

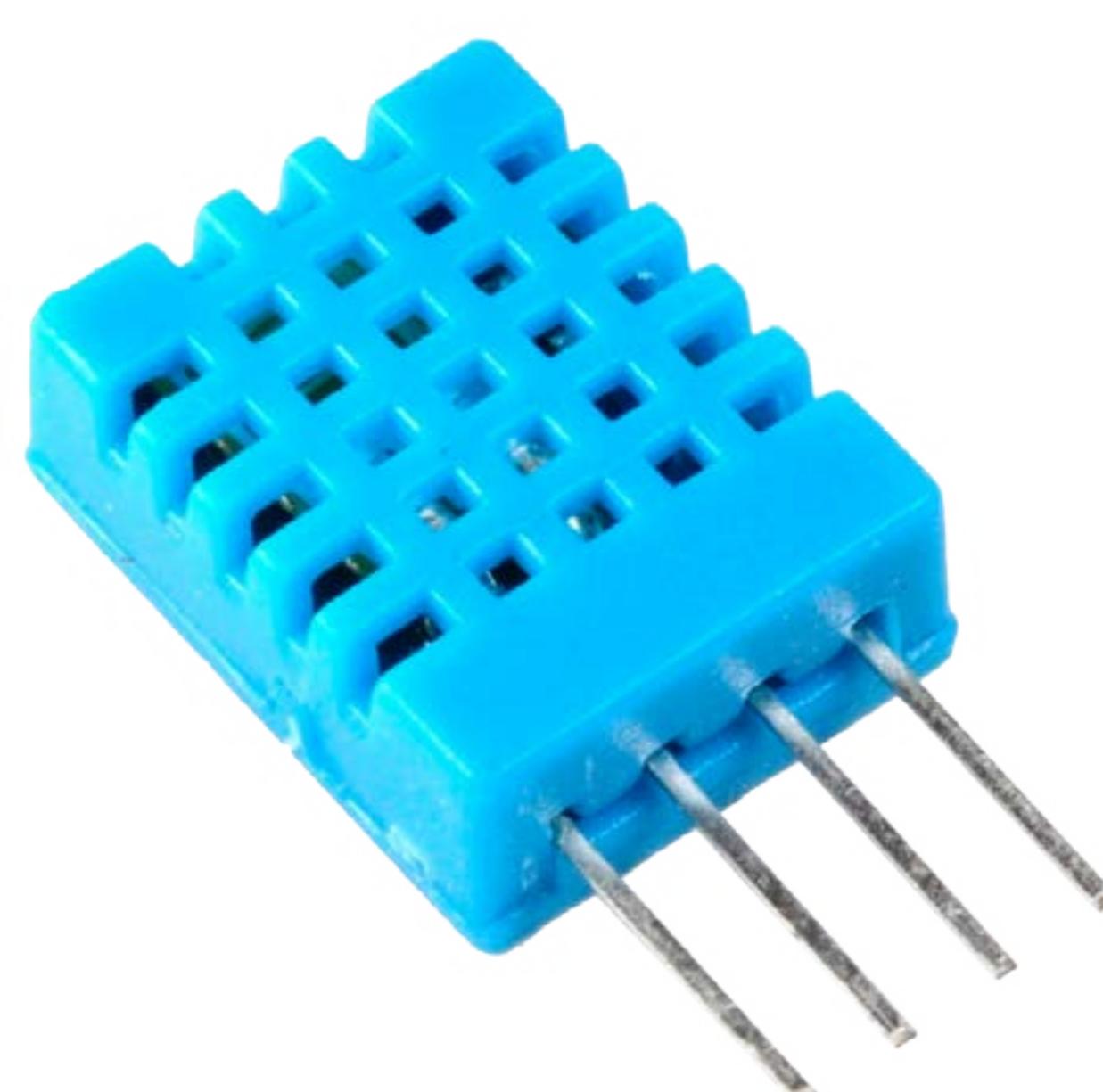


**Neste artigo** mostramos como acompanhar as informações de temperatura e umidade no monitor serial, método ideal para quem ainda não tem um display LCD. Para quem já tem um LCD 16×2 e quer melhorar o projeto, vamos mostrar como ligar o sensor **DHT11** juntamente com o display e mostrar nele as informações que precisamos.



## UTILIZANDO O SENSOR DHT11

Para este circuito, você pode utilizar o módulo DHT11 ou apenas o [sensor DHT11](#). A ligação dos dois é idêntica e utiliza apenas um pino para ligação ao Arduino.

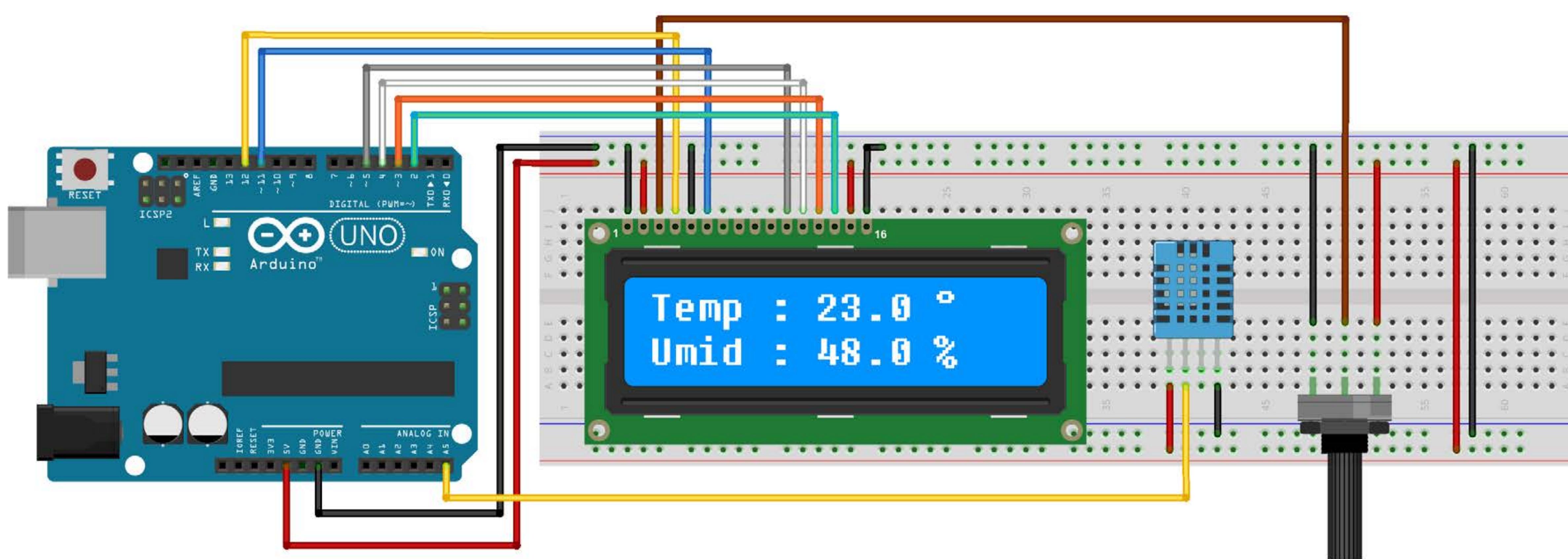


Uma característica do sensor DHT11 é que ele não fornece informações “quebradas” de temperatura. Isso significa que o sensor vai mostrar as informações de, por exemplo, 18, 20, 25 graus, mas não as casas decimais de 18,2 ou 25,6 graus.

O [display LCD 16x2](#) que vamos utilizar, baseado no controlador HD44780, é um display com backlight azul e caracteres na cor branca, com os pinos de conexão na parte superior numerados de 1 a 16. A conexão básica ao Arduino usa 6 pinos :

- **Pino 4 (RS)** ligado ao pino **12** do Arduinou;
- **Pino 6 (E)** ligado ao pino **11** do Arduino;
- **Pino 11 (D4)** ligado ao pino **5** do Arduino;
- **Pino 12 (D5)** ligado ao pino **4** do Arduino;
- **Pino 13 (D6)** ligado ao pino **3** do Arduino;
- **Pino 14 (D7)** ligado ao pino **2** do Arduino.

O pino 3 do display será ligado ao pino central de um potenciômetro de 10K, que tem a função de regular o contraste. As demais ligações são feitas ao GND (pinos 1, 5 e 16) e aos 5v do Arduino (pinos 2 e 15), e qualquer inversão pode impedir a exibição dos caracteres:



No programa, vamos utilizar a biblioteca LiquidCrystal para controle do LCD (esta biblioteca já vêm instalada na IDE), e também a biblioteca DHT, que pode ser baixada neste [link](#).

Para mostrar o símbolo do grau (°), podemos utilizar um dos caracteres especiais disponíveis nesse display, usando o comando

***Lcd.print((char)223);***

Ou criar um caractere customizado, com a forma mais arredondada. Para isso, criamos um array e desenhamos nosso próprio símbolo, e para utilizá-lo no programa, usamos o comando

***Lcd.createChar(valor, data);***

onde **valor** se refere ao nome que daremos ao caractere especial, podendo ser um número de 0 a 7, e **data** se refere ao array criado para formar o símbolo do grau.

O comando **delay** no final do programa não deve ter um valor abaixo de 2000 (2 segundos), que é o valor mínimo para que o sensor possa fornecer os dados corretamente.

```
//Programa : Temperatura e umidade com o DHT11 e LCD 16x2
//Autor : FILIPEFLOP

#include <LiquidCrystal.h> //Carrega a biblioteca LCD
#include <DHT.h> //Carrega a biblioteca DHT

//Define a ligação ao pino de dados do sensor
#define DHTPIN A5

//Define o tipo de sensor DHT utilizado
#define DHTTYPE DHT11

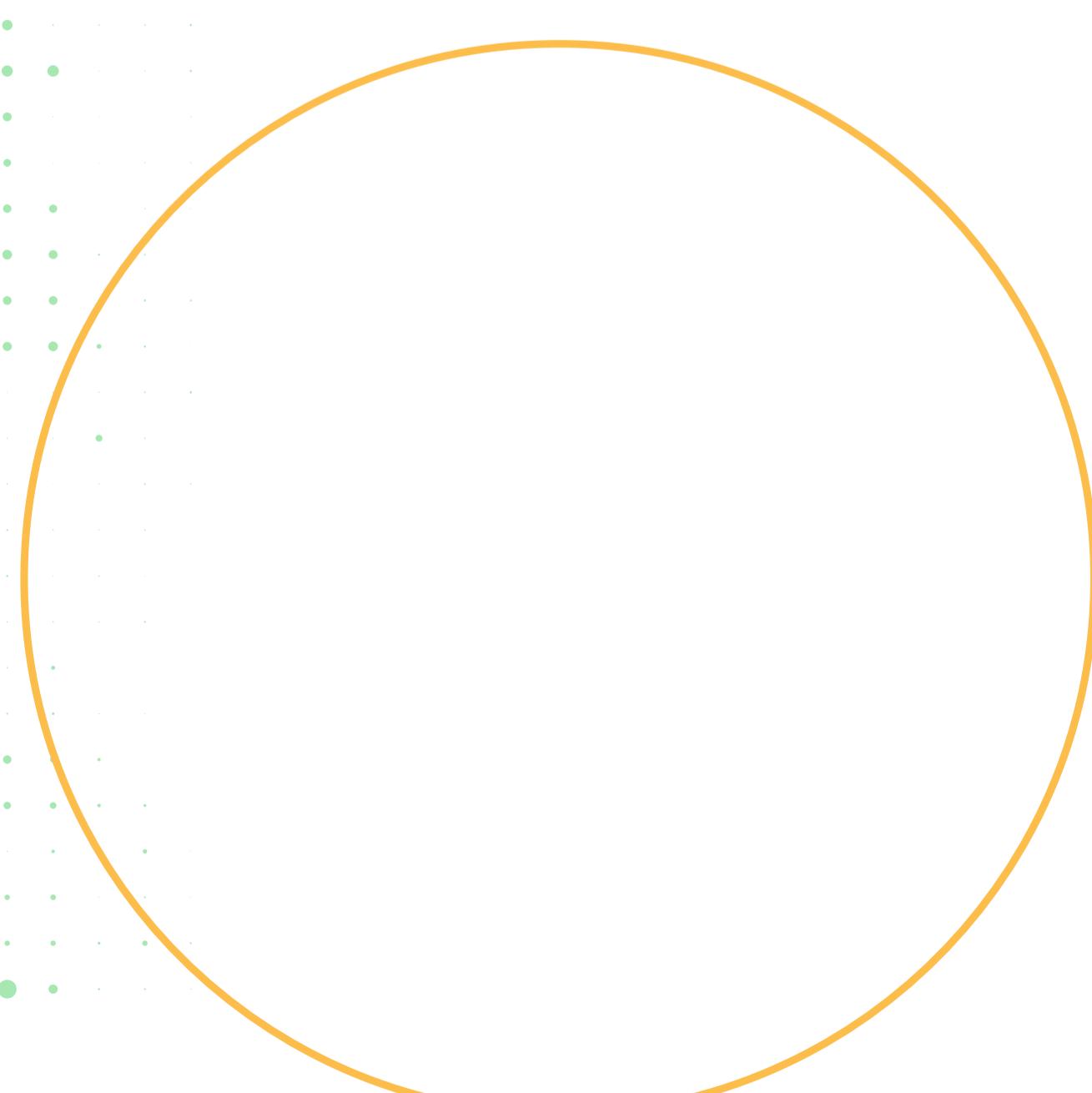
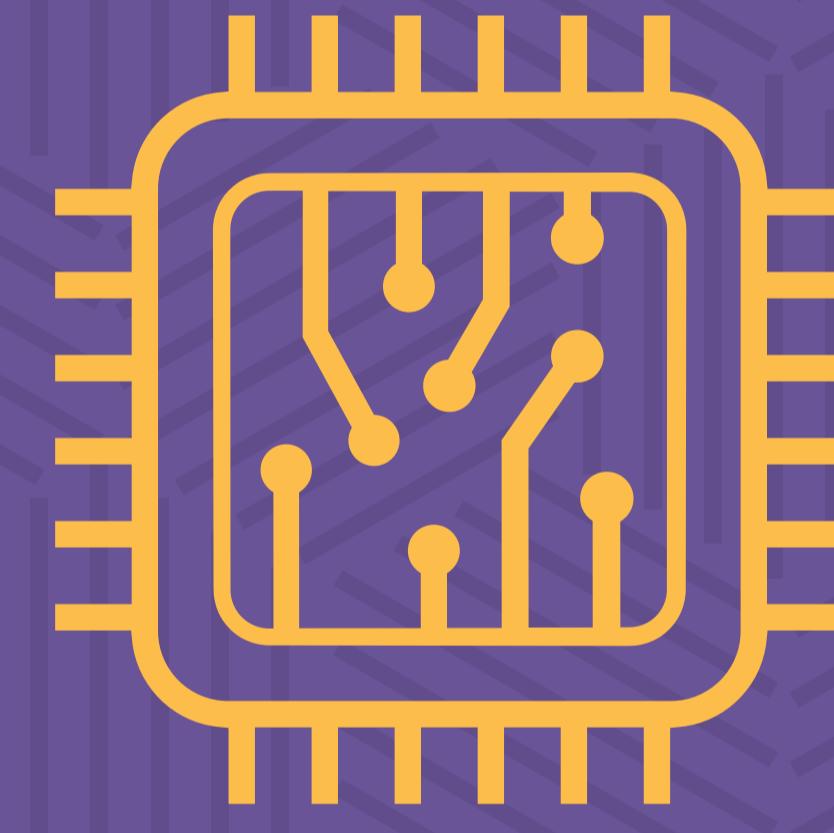
DHT dht(DHTPIN, DHTTYPE);

//Define os pinos que serão ligados ao LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

```
.....  
  
//Array simbolo grau  
byte grau[8] ={ B00001100,  
                B00010010,  
                B00010010,  
                B00001100,  
                B00000000,  
                B00000000,  
                B00000000,  
                B00000000,};  
  
void setup()  
{  
    Serial.begin(9600); //Inicializa a serial  
    lcd.begin(16,2); //Inicializa LCD  
    lcd.clear(); //Limpa o LCD  
    //Cria o caractere customizado com o simbolo do grau  
    lcd.createChar(0, grau);  
}  
  
void loop()  
{  
    float h = dht.readHumidity(); //Le o valor da umidade  
    float t = dht.readTemperature(); //Le o valor da temperatura  
    lcd.setCursor(0,0);  
    lcd.print("Temp : ");  
    lcd.print(" ");  
    lcd.setCursor(7,0);  
    lcd.print(t,1);  
    lcd.setCursor(12,0);  
  
    //Mostra o simbolo do grau formado pelo array  
    lcd.write((byte)0);  
  
    //Mostra o simbolo do grau quadrado  
    //lcd.print((char)223);  
  
    lcd.setCursor(0,1);  
    lcd.print("Umid : ");  
    lcd.print(" ");  
    lcd.setCursor(7,1);  
    lcd.print(h,1);  
    lcd.setCursor(12,1);  
    lcd.print("%");  
  
    //Intervalo recomendado para leitura do sensor  
    delay(2000);  
}
```



# CONTROLANDO LÂMPADAS COM MÓDULO RELÉ ARDUINO

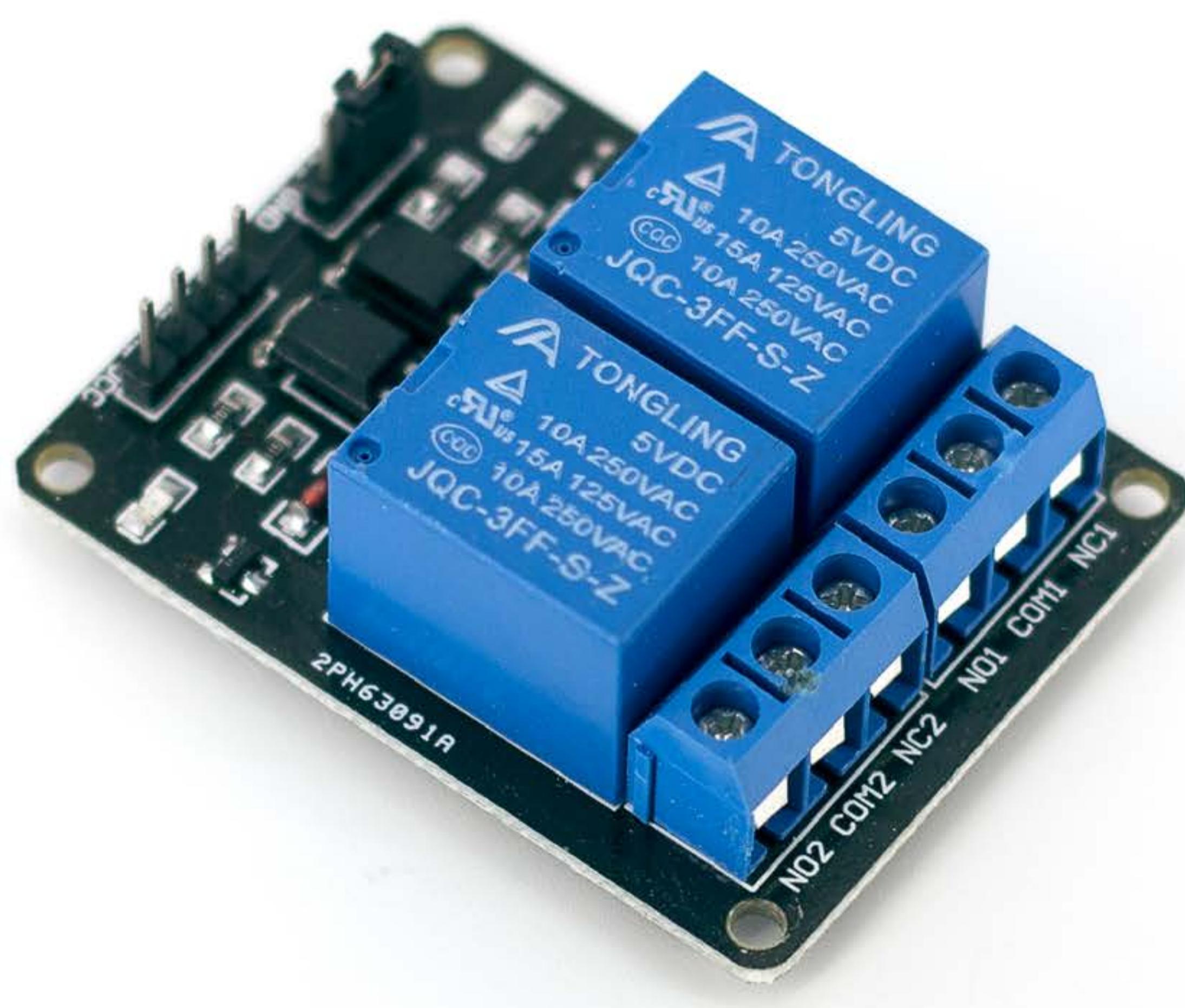




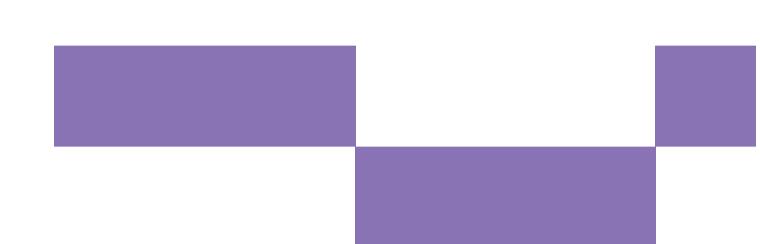
---

Nesta seção vamos explicar como usar o [Módulo Relé](#) Arduino e montar 2 circuitos: um circuito com **acionamento temporizado de 2 lâmpadas** e outro circuito **com acionamento por botões** de uma lâmpada e um ventilador (pode substituir por outro eletrodoméstico por exemplo).

Este relé 5V pode ser também usado com AVR, PIC, Raspberry, 8051, ARM ou até mesmo o seu circuito eletrônico personalizado. Usando dois pinos do módulo relé arduino, você controla cargas como lâmpadas, motores, fechaduras e eletrodomésticos, desde que a corrente de operação não ultrapasse 10 A (ampéres).



**\_VOCÊ CONTROLA  
CARGAS COMO  
LÂMPADAS, MOTORES,  
FECHADURAS E  
ELETRODOMÉSTICOS**



Cada relé desse módulo suporta cargas de até 10 A, em 125 VAC, 250 VAC ou 30 VDC. Leds indicadores mostram o estado do relé (ligado/desligado) em cada canal. O módulo já contém todo o circuito de proteção para evitar danos ao microcontrolador, e possui baixa corrente de operação.

## ESPECIFICAÇÕES MÓDULO RELÉ 5V

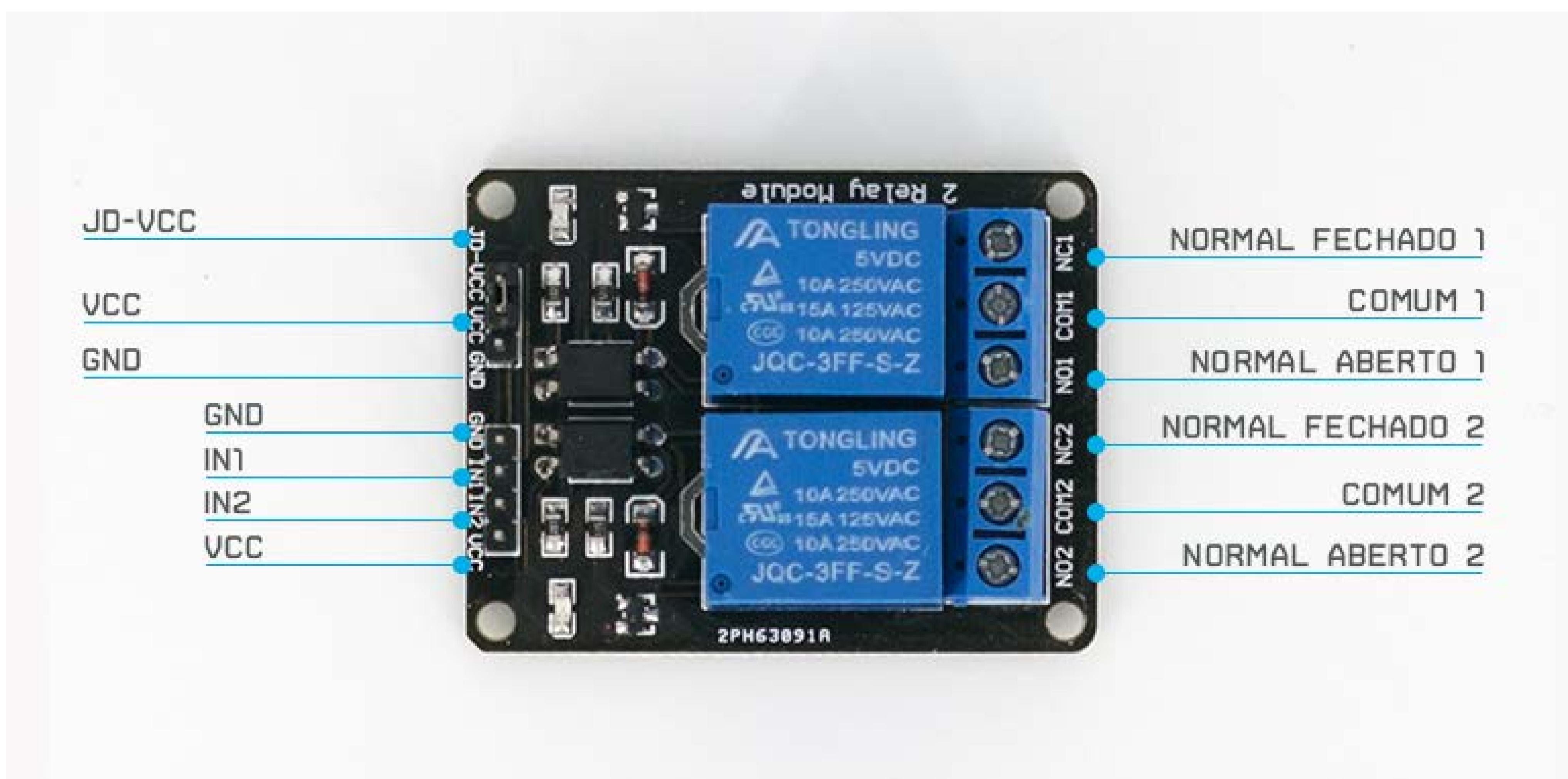
---

- Tensão de operação : 5 VDC;
- Modelo Relé : SRD-05VDC-SL-C (Datasheet);
- Permite controlar cargas de 220V AC;
- Nível de sinal dos pinos IN1 e IN2 : 5 VDC;
- Corrente de operação : 15 ~ 20 mA;
- Tempo de resposta : 5 ~ 10 ms;
- 4 furos de 3mm para fixação, nas extremidades da placa;
- Dimensões reduzidas : 51 x 38 x 20 mm.

## PINAGEM MÓDULO RELÉ 5V

---

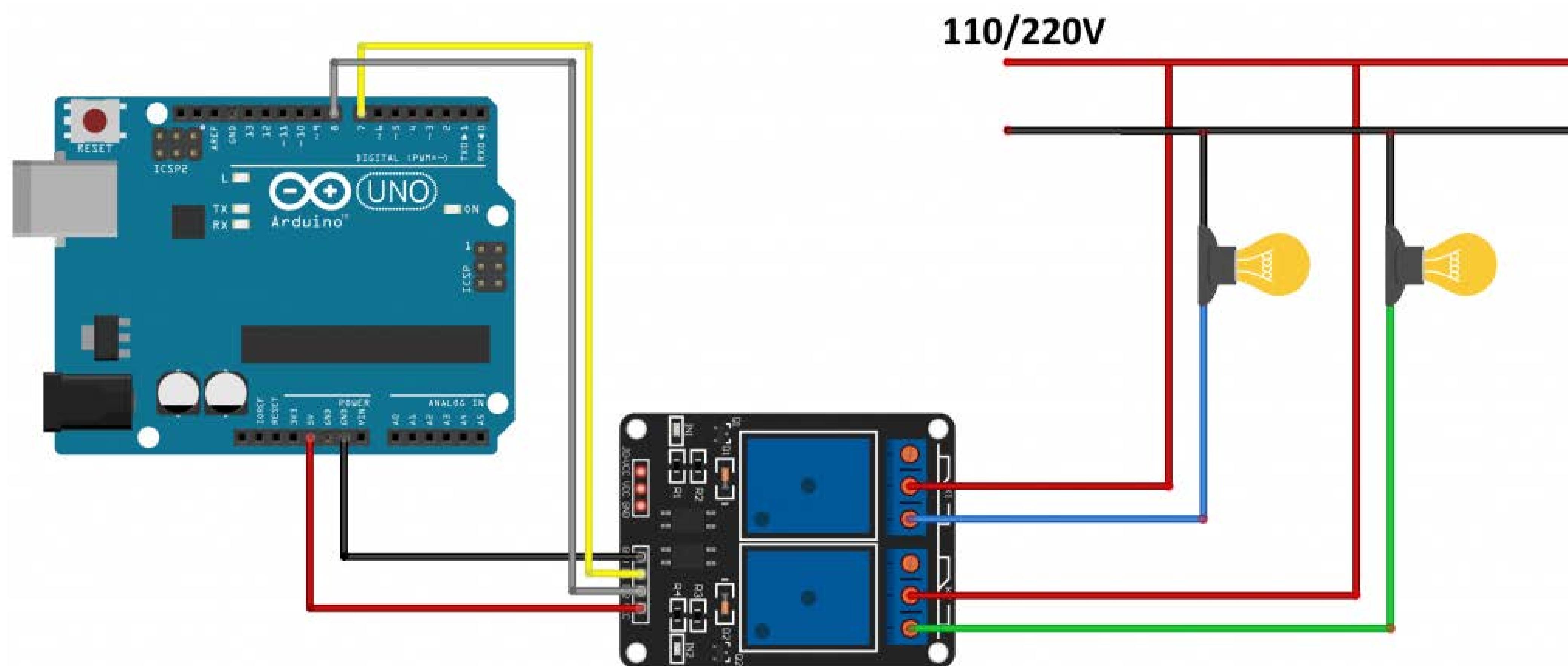
Na imagem abaixo você pode observar a pinagem do módulo relé arduino. No lado esquerdo superior os pinos **JD-Vcc**, **Vcc** e **GND**, que permitem que seja conectada uma fonte externa de 5V. Abaixo, os pinos **GND**, **IN1** (aciona o relé 1), **IN2** (aciona o relé 2), e o **Vcc**. Ao lado dos relés, os contatos **NC (Normal Fechado)**, **C (Comum)**, e **NA (normal aberto)**:



Acionar os dois relés é muito simples e para isso você deve apenas mudar o estado das portas digitais ligadas aos pinos **IN1** e **IN2**. Um detalhe importante desse módulo é que os relés são ativados em nível baixo, ou seja, quando o estado da porta estiver em **LOW**, o relé será acionado.

## MÓDULO RELÉ COM ARDUINO: CIRCUITO 2 LÂMPADAS

No exemplo abaixo, vamos acionar duas lâmpadas alternadamente. Os dois relés irão funcionar como chaves eletrônicas, fechando o contato **NA (Normal Aberto)**, e acendendo as lâmpadas a cada 2 segundos. Recomendamos muito cuidado tanto no manuseio como na ordem de ligação dos fios, pois estamos trabalhando com 110 V (ou 220 v) da rede elétrica.



O controle do **relé 1** é feito pela porta **7** do [Arduino Uno](#), e o **relé 2** é controlado pela porta 8. As duas portas são definidas como saídas e alternam os estados **LOW** e **HIGH**, lembrando que o estado baixo (**LOW**), é que aciona o relé:

```
//Programa : Teste Modulo Rele Arduino 2 canais - Lampadas
//Autor : FILIPEFLOP

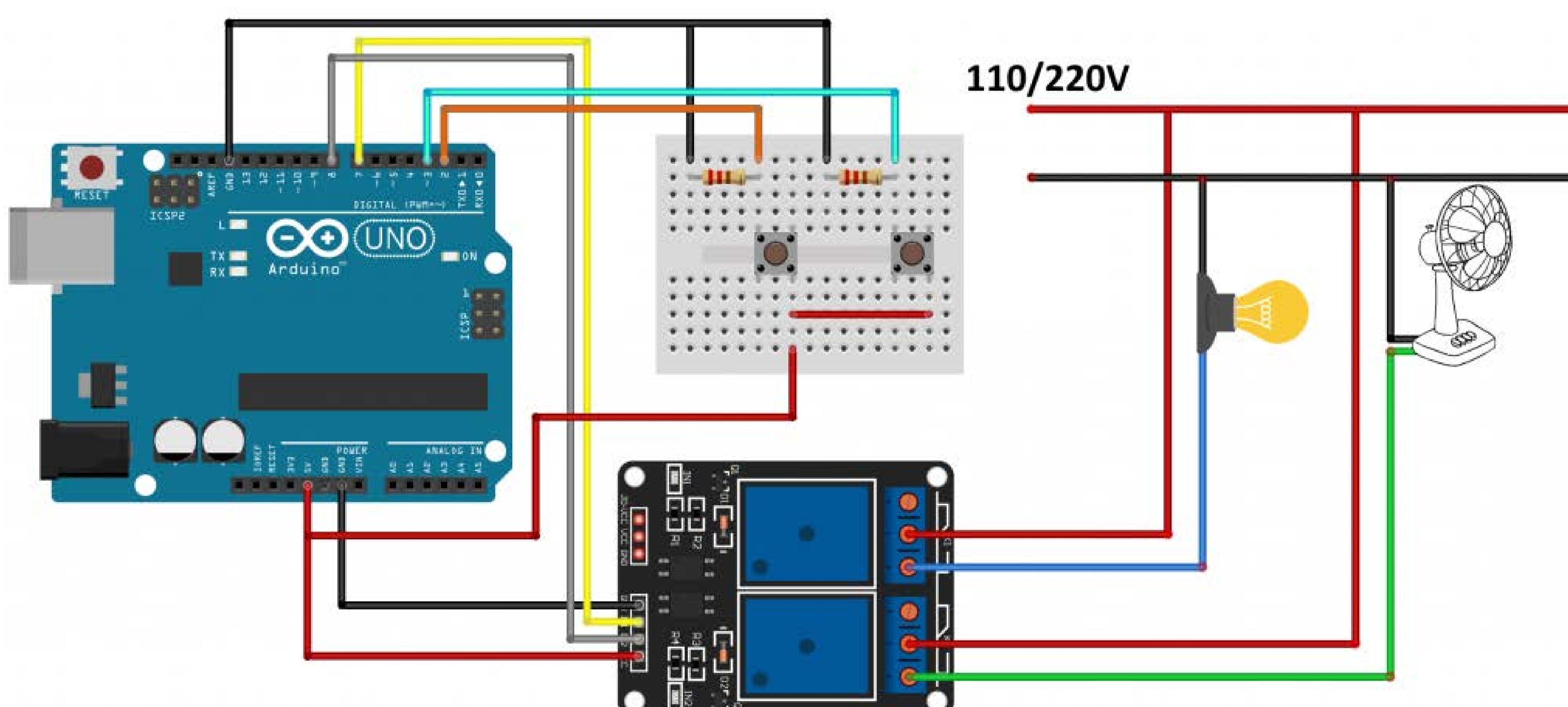
//Porta ligada ao pino IN1 do modulo
int porta_rele1 = 7;
//Porta ligada ao pino IN2 do modulo
int porta_rele2 = 8;

void setup()
{
    //Define pinos para o rele como saida
    pinMode(porta_rele1, OUTPUT);
    pinMode(porta_rele2, OUTPUT);
}

void loop()
{
    digitalWrite(porta_rele1, LOW); //Liga rele 1
    digitalWrite(porta_rele2, HIGH); //Desliga rele 2
    delay(2000);
    digitalWrite(porta_rele1, HIGH); //Desliga rele 1
    digitalWrite(porta_rele2, LOW); //Liga rele 2
    delay(2000);
}
```

## MÓDULO RELÉ ARDUINO: CIRCUITO 2 BOTÕES

Vamos melhorar esse circuito adicionando dois botões para que você mesmo controle o acionamento dos relés. Vamos utilizar dois push-buttons, mas o mesmo circuito pode ser utilizado com outro tipo de interruptor e até mesmo sensores, como LDR ou [sensor óptico](#). O **botão da esquerda** aciona o **relé 1**, que por sua vez está ligado à uma lâmpada. Já o **botão da direita** controla o **relé 2**, que no nosso exemplo liga e desliga um ventilador, mas pode ser substituído por qualquer aparelho de sua preferência, como uma cafeteira ou um motor , desde que a corrente exigida não ultrapasse 10A:



Utilizamos novamente as portas 7 e 8, e vamos apenas alterar o programa, para que seja feita a leitura dos botões e o respectivo acionamento dos relés. Como estamos utilizando push-buttons, a cada acionamento o estado do relé será invertido, ligando ou desligando o dispositivo.

```
//Programa : Teste Módulo Rele Arduino - Botoes
//Autor : FILIPEFLOP

//Porta ligada ao pino IN1 do modulo
int porta_rele1 = 7;
//Porta ligada ao pino IN2 do modulo
int porta_rele2 = 8;
//Porta ligada ao botao 1
int porta_batao1 = 2;
//Porta ligada ao botao 2
int porta_batao2 = 3;

//Armazena o estado do rele - 0 (LOW) ou 1 (HIGH)
int estadorele1 = 1;
int estadorele2 = 1;
//Armazena o valor lido dos botoes
int leitura1 = 0;
int leitura2 = 0;

void setup()
{
    //Define pinos para o rele como saida
    pinMode(porta_rele1, OUTPUT);
    pinMode(porta_rele2, OUTPUT);
    //Define pinos dos botoes como entrada
    pinMode(porta_batao1, INPUT);
    pinMode(porta_batao2, INPUT);
    //Estado inicial dos reles - desligados
    digitalWrite(porta_rele1, HIGH);
    digitalWrite(porta_rele2, HIGH);
}

void loop()
{
    //Verifica o acionamento do botao 1
    leitura1 = digitalRead(porta_batao1);
    if (leitura1 != 0)
    {
        while(digitalRead(porta_batao1) != 0)
        {
            delay(100);
        }
        //Inverte o estado da porta
        estadorele1 = !estadorele1;
        //Comandos para o rele 1
        digitalWrite(porta_rele1, estadorele1);
    }
}
```

```
.....  
//Verifica o acionamento do botao 2  
leitura2 = digitalRead(porta_bota02);  
if (leitura2 != 0)  
{  
    while(digitalRead(porta_bota02) != 0)  
    {  
        delay(100);  
    }  
    //Inverte o estado da porta  
    estadorele2 = !estadorele2;  
    //Comandos para o rele 2  
    digitalWrite(porta_rele2, estadorele2);  
}  
}
```





## CONCLUSÃO

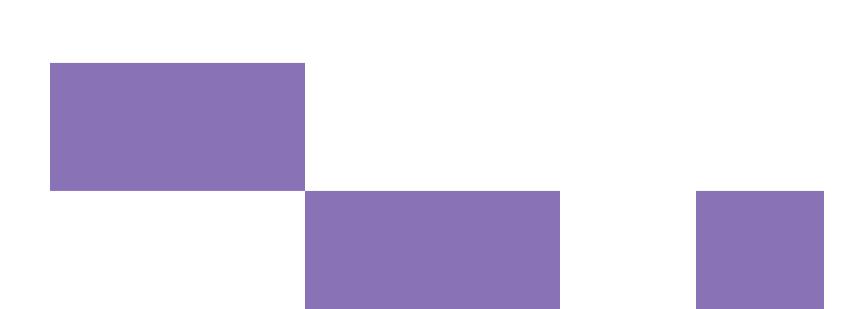
Sem dúvida o Arduino é uma plataforma incrível e a possibilidade de projetos possíveis de serem feitos é praticamente infinita.

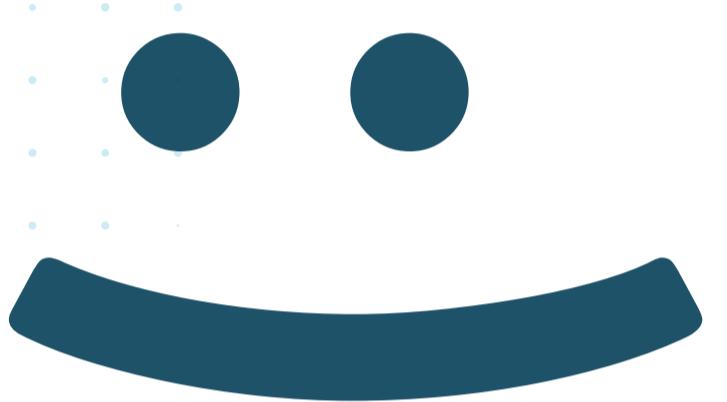
Esperamos que você tenha compreendido os primeiros passos com Arduino e que o guia tenha sido esclarecedor. Faça você mesmo! Faça agora!

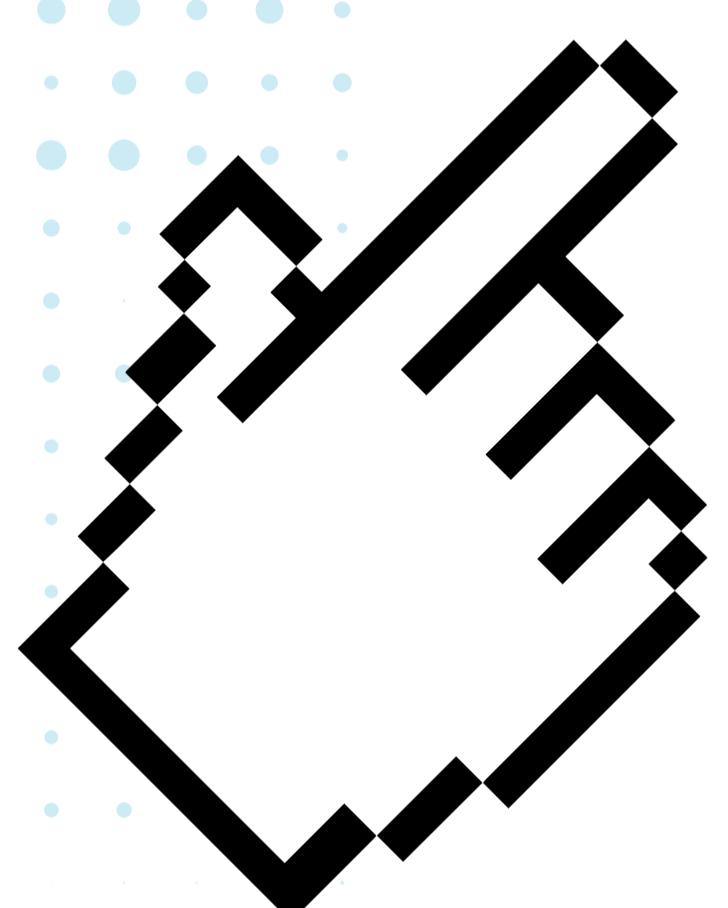
Visite nosso Blog e descubra muito mais ideias de projetos com Arduino. E se você quiser compartilhar o seu próprio projeto, tirar dúvidas e interagir com outros makers, acesse nosso Fórum!

[VISITAR BLOG](#)

[ACESSAR FÓRUM](#)



WORK  
**HARD**  
HAVE   
**FUN**  
**& MAKE THINGS.**



---

Acompanhe  
nossas redes sociais

