

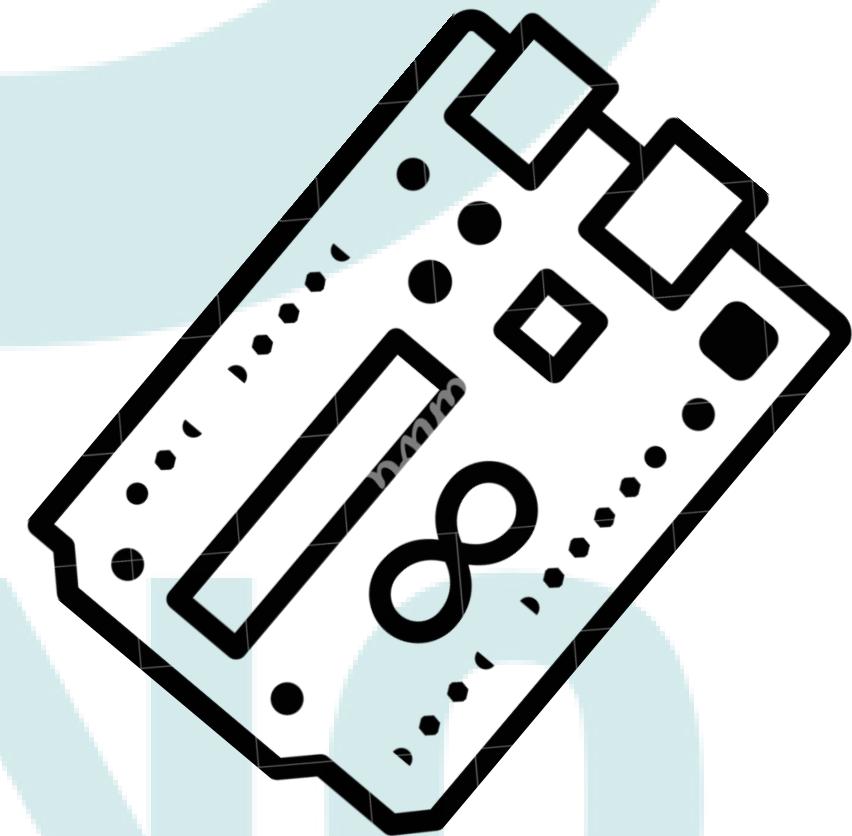
Arduino: Conceitos, Programação e Aplicação Prática



ARDUINO

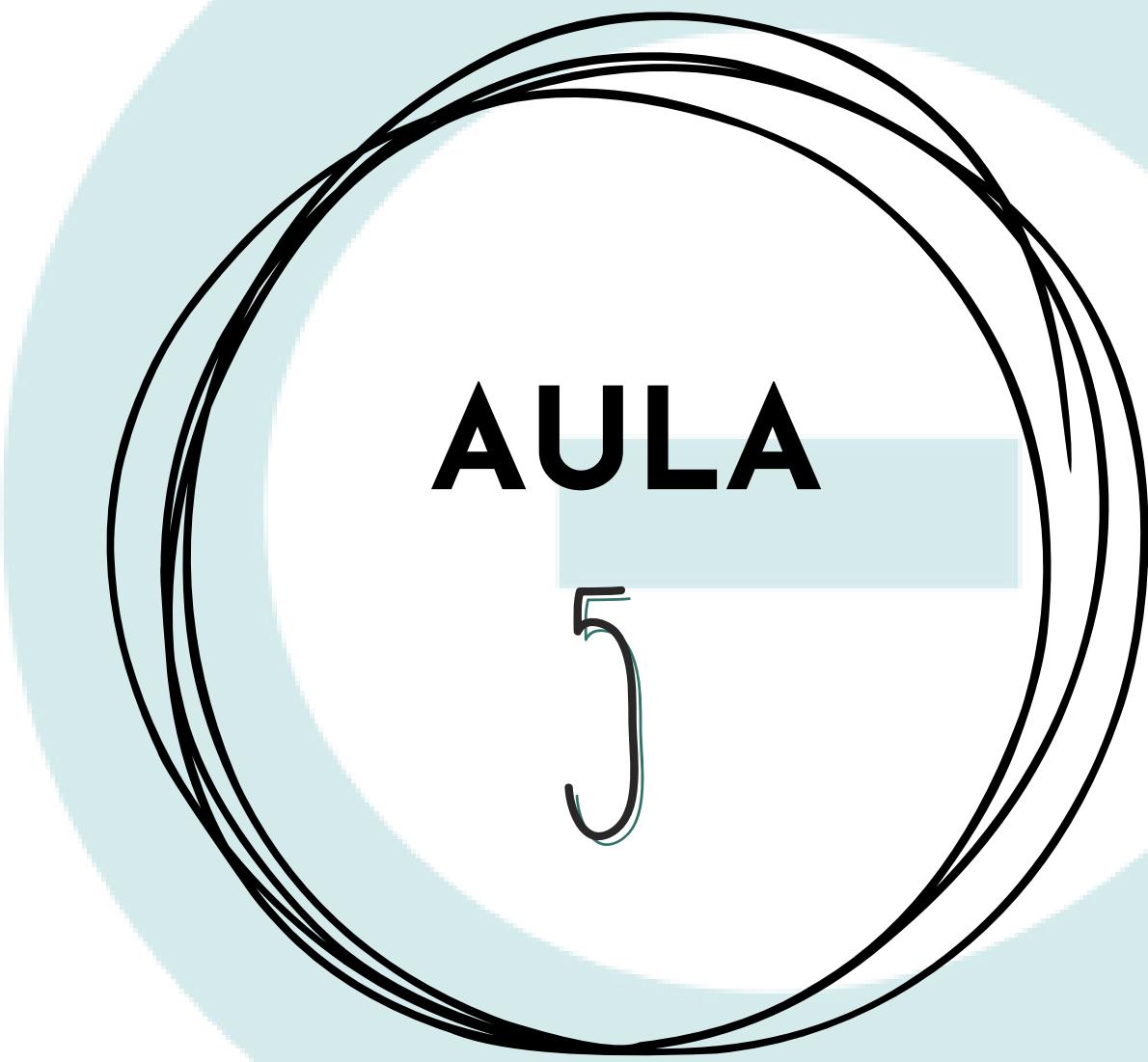
UTP
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CÂMPUS PATO BRANCO



®

®



AULA

5

Potenciômetro e
LED RGB

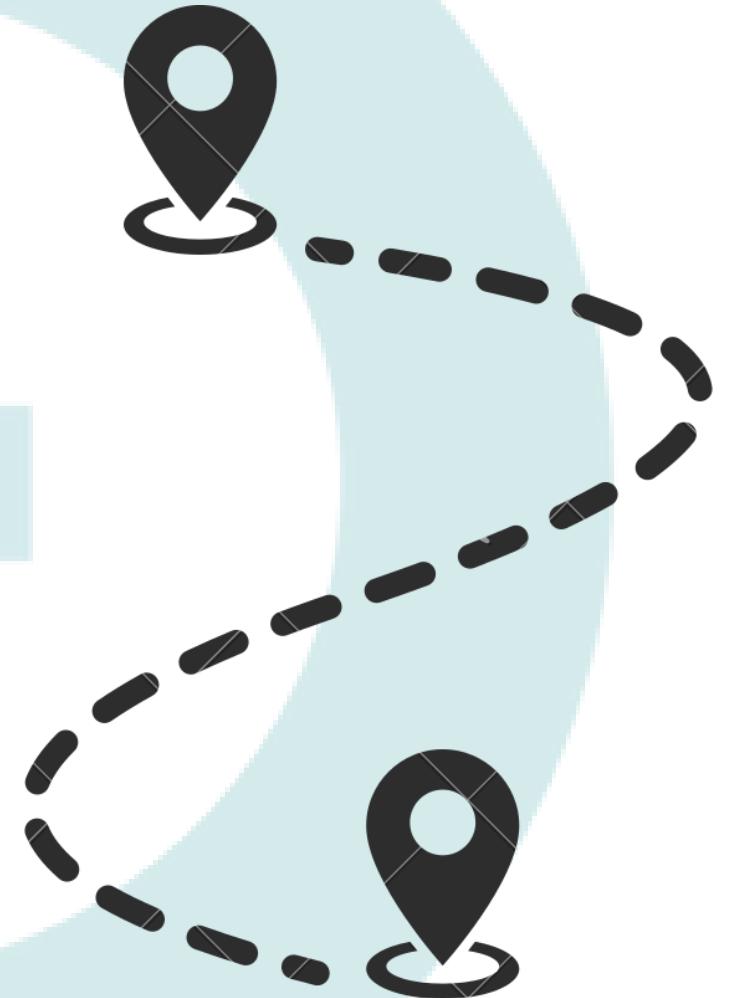
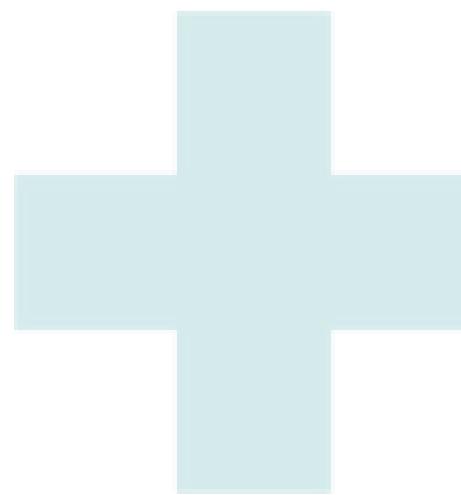
Coordenador
Fábio Favarim

Instrutor
Gabriel S. Folly

ARDUTENO

Roteiro

1. Potenciômetro
2. Função map()
3. LED RGB
4. Laboratórios



ARDUTENO

Potenciômetro - Conceitos e Tipos

- **Definição:** O potenciômetro é um componente eletrônico com resistência ajustável, controlado por um botão giratório.
- **Tipos de Potenciômetros:** Ajuste único (linear) e ajuste duplo (logarítmico), rotativos e lineares.
- **Conexão ao Arduino:** Conectado a pinos analógicos, fornecendo entrada analógica proporcional à sua posição.

ARDUTENO

Aplicações e Utilização do Potenciômetro

- **Aplicações Práticas:** Ajuste de variáveis analógicas como volume, brilho, contraste e temperatura.
- **Conexão e Utilização:** Terminal central (cursor) conectado ao ponto de ajuste, os outros terminais variam a resistência total.
- **Controle Analógico:** Proporciona controle suave e contínuo sobre sinais elétricos, essencial em aplicações que requerem modulação fina.

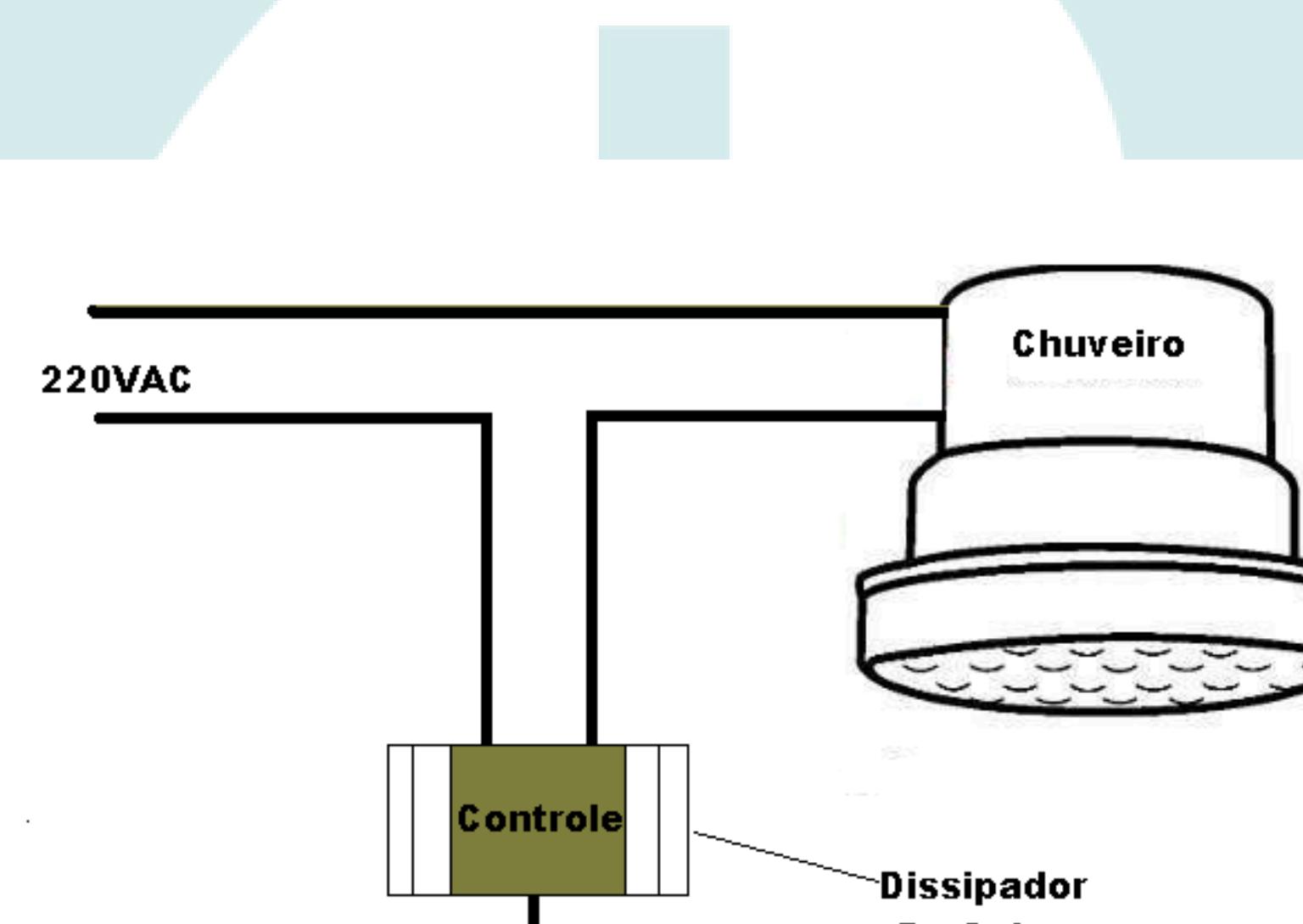
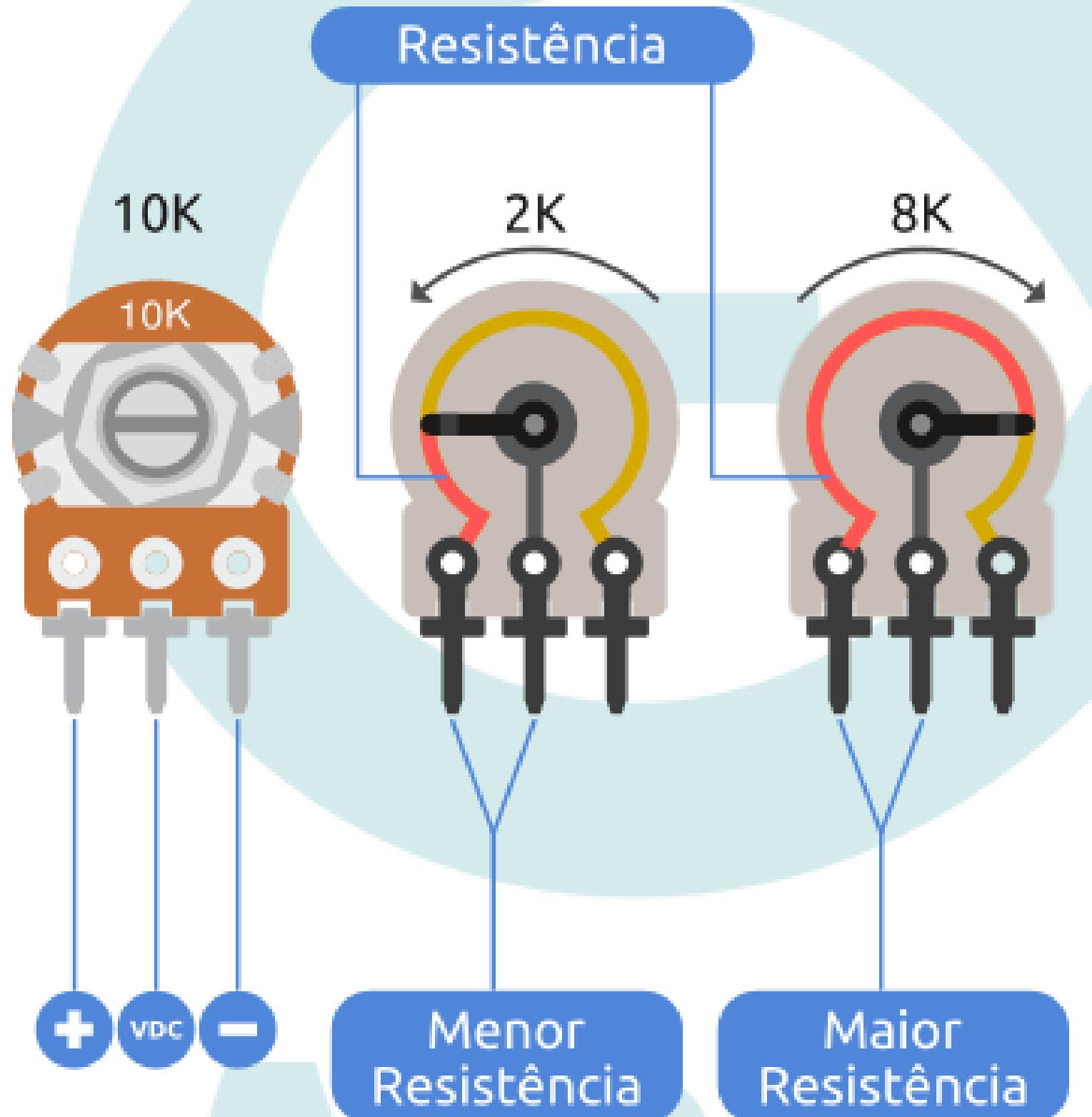
ARDUTINO

Potenciômetro - Papel Crucial em Projetos

- **Papel Crucial:** Desempenha um papel essencial na criação de interfaces ajustáveis.
- **Controle Analógico:** Possibilita controle analógico em dispositivos eletrônicos.
- **Ampla Variedade de Aplicações:** Encontrado em circuitos de áudio, amplificadores, controles de rádio, entre outros.
- **Elemento Essencial:** Torna-se um elemento essencial em projetos de eletrônica, proporcionando controle e ajuste em diversas aplicações.

ARDUINO

R®

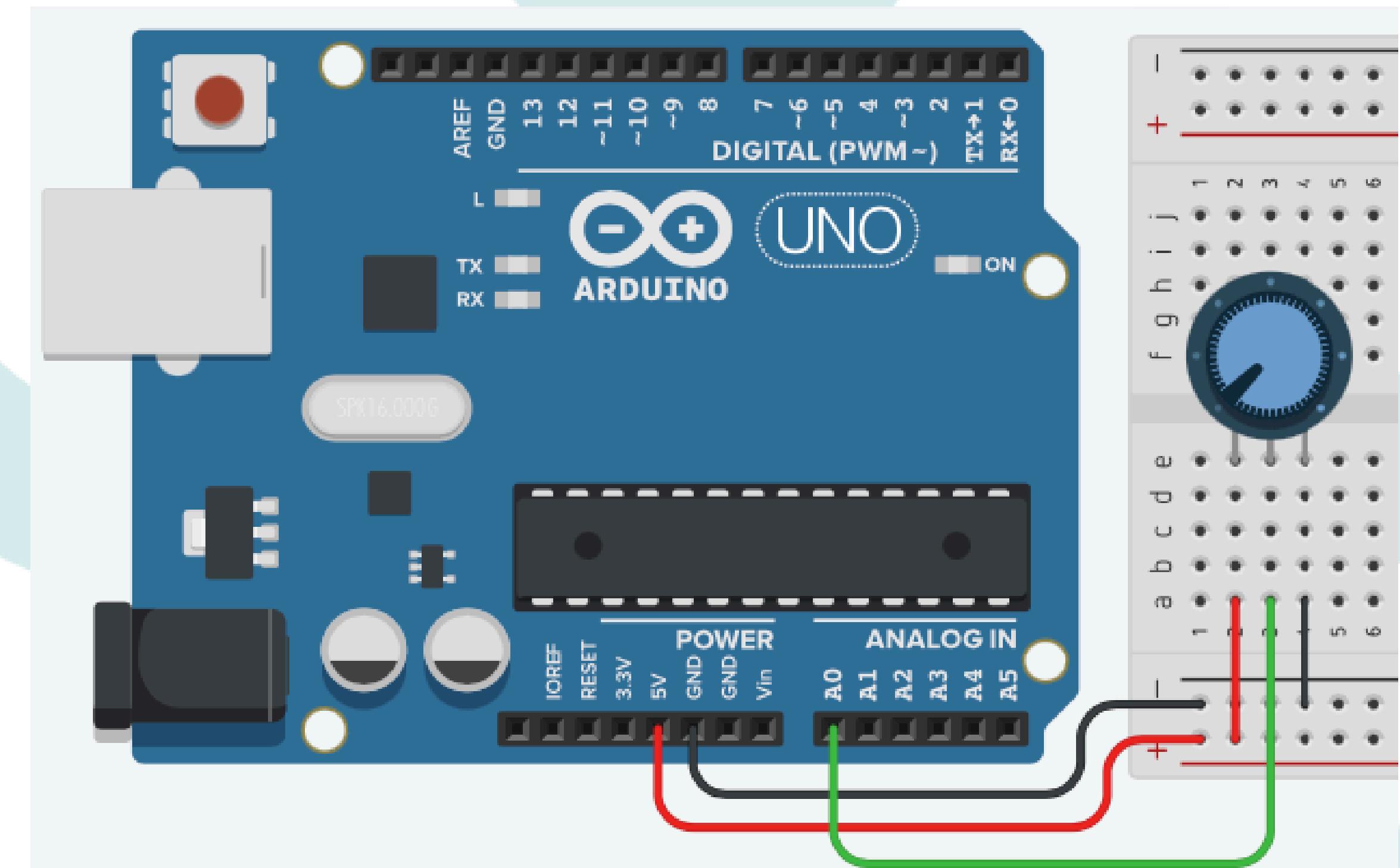


Conexão com Arduino

- Para utilizar o potenciômetro em projetos com Arduino devemos conectá-lo da seguinte maneira.
- **Conexão Física:**
 - Conecte um fio do terminal central do potenciômetro a quaisquer pinos analógicos no Arduino. Sendo estes A0, A1, A2, A3, A4 e A5.
 - Conecte um fio de um dos terminais laterais ao pino GND (terra) no Arduino.
 - Conecte o outro terminal lateral a um dos pinos 5V no Arduino.
- No exemplo a seguir é possível ver uma possível conexão, onde a porta analógica escolhida foi a A0.

ARDUTINO

R



ANALOG INPUT

Função 'map()' no Arduino

- **Sintaxe:** map(valor, deLimitelinferior, deLimiteSuperior, paraLimitelinferior, paraLimiteSuperior)
- **Parâmetros:**
 - **valor:** O valor a ser mapeado.
 - **deLimitelinferior e deLimiteSuperior:** Limites do intervalo original.
 - **paraLimitelinferior e paraLimiteSuperior:** Limites desejados para o mapeamento.
- **Funcionamento:** Transformação linear ajusta o valor de entrada proporcionalmente entre os intervalos especificados.
- **Exemplo Prático:** Mapeamento de leitura analógica (potenciômetro) de 0 a 1023 para controle de intensidade de LED de 0 a 255 (intervalo comum para PWM no Arduino).

```
1 int valorPotenciometro = analogRead(pinoPotenciometro);
2 int intensidadeLED = map(valorPotenciometro, 0, 1023, 0, 255);
3 analogWrite(pinoLED, intensidadeLED);
```

LEDs RGB

- **Estrutura Básica:**

- Composto por LEDs individuais (vermelho, verde, azul).
- Controle separado para cada LED permite mistura de cores.

- **Três Canais de Controle:**

- Controle independente para vermelho, verde e azul.
- Ajuste de intensidade para variedade de cores e tons.

- **Pinos e Conexões:**

- **Quatro pinos:** GND (cátodo comum) e ânodos para R, G, B.
- Controle de intensidade ligando ânodo de cada cor a uma tensão específica.

- **Controle de Cores:**

- Combinação de intensidades nas cores cria milhões de tonalidades.
- Usado em iluminação, displays coloridos e sinalização.

Aplicações e Controle

- **Aplicações Comuns:**

- Iluminação ambiente em projetos eletrônicos.
- Displays informativos coloridos.
- Decoração em ambientes controlados por microcontroladores.

- **Controle por PWM:**

- Utilização frequente de modulação por largura de pulso (PWM).
- Arduino e outras placas microcontroladoras usam PWM para controlar intensidade.

ARDUTENO

Programação e Alimentação

- **Programação e Controle:** Define intensidade para cada canal (R, G, B) usando funções como `analogWrite()` no Arduino.
- **Considerações de Alimentação:** Consumo de energia pode ser maior devido ao controle individual. Dimensionamento adequado da fonte de alimentação é crucial.
- **Conclusão:** LEDs RGB oferecem flexibilidade para efeitos visuais coloridos e dinâmicos. Escolha popular em projetos que demandam iluminação personalizável e controle preciso de cores.

ARDUINO

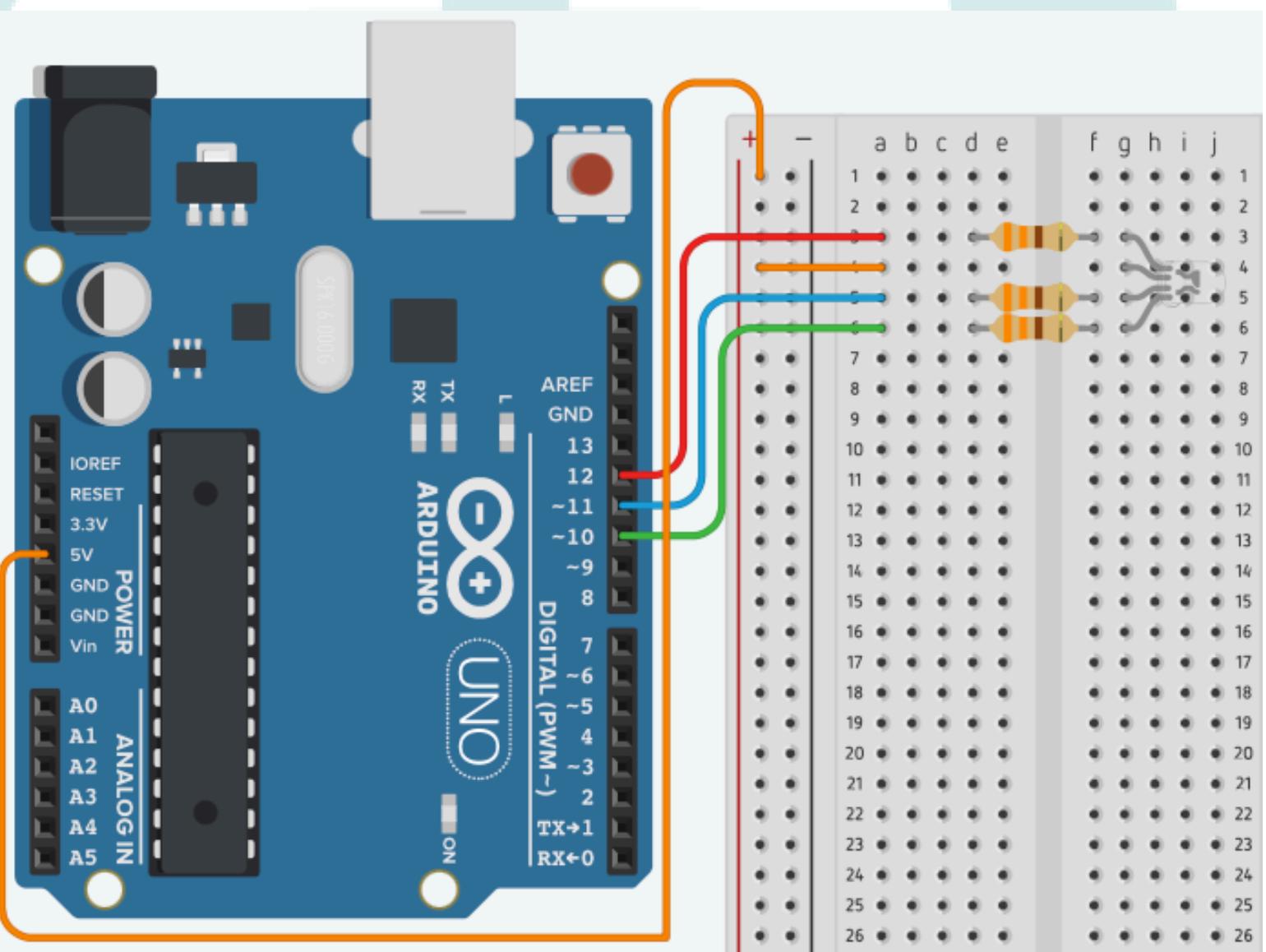
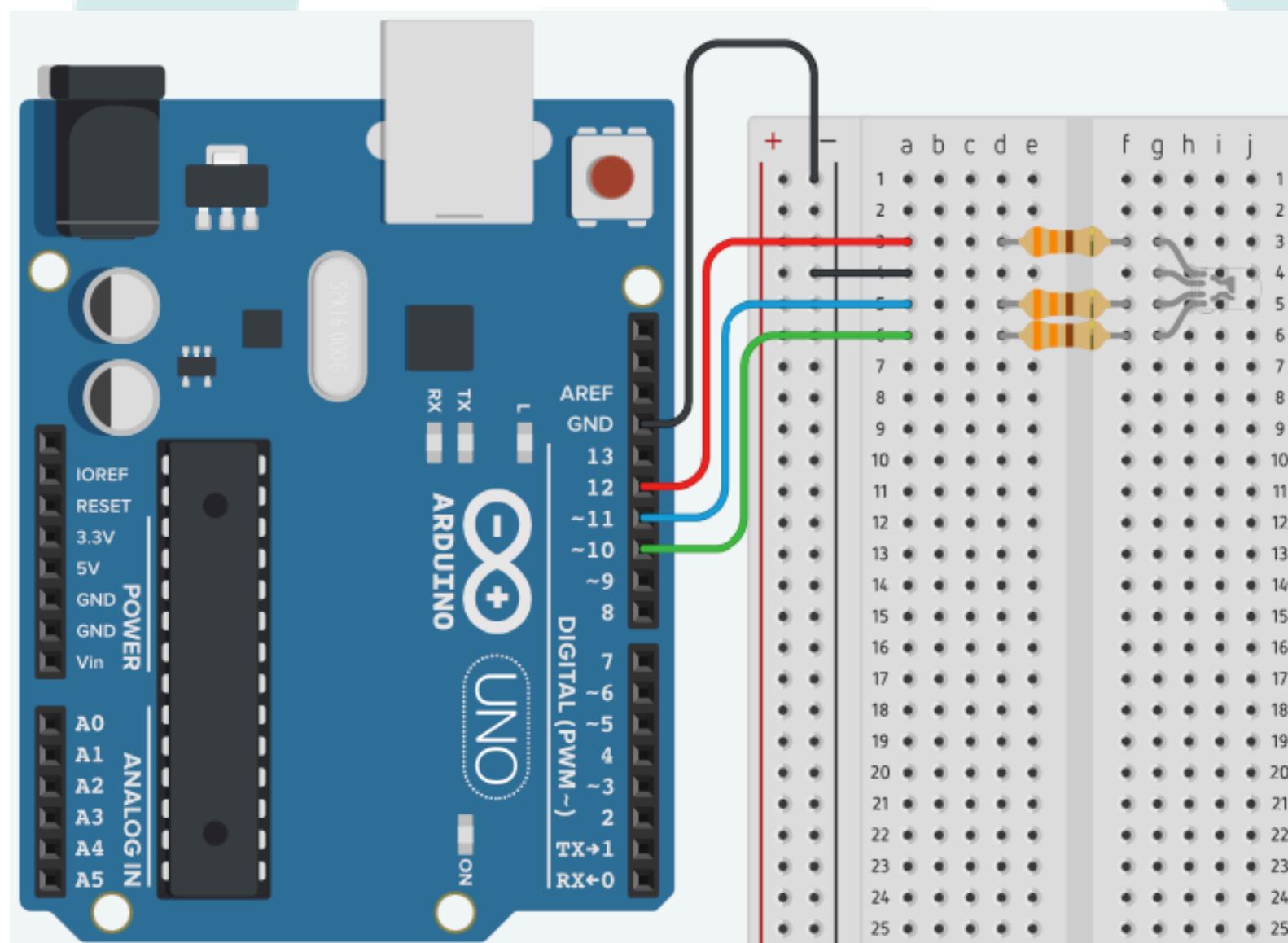
Tabela cores RGB

Cor	RGB	LED RGB catodo comum	LED RGB ânodo comum
Vermelho	(255, 0, 0)	(255, 0, 0)	(0, 255, 255)
Verde	(0, 255, 0)	(0, 255, 0)	(255, 0, 255)
Azul	(0, 0, 255)	(0, 0, 255)	(255, 255, 0)
Amarelo	(255, 255, 0)	(255, 255, 0)	(0, 0, 255)
Azul meia-noite	(25, 25, 112)	(25, 25, 112)	(230, 230, 127)
Roxo	(128, 0, 128)	(128, 0, 128)	(127, 255, 127)
Verde mar médio	(60, 179, 113)	(60, 179, 113)	(195, 76, 142)

ARDUTINO

Cátodo Comum e Ânodo Comum

- Cátodo Comum



ARDUINO

Laboratório 01

Semáforo de pedestres com um LED RGB

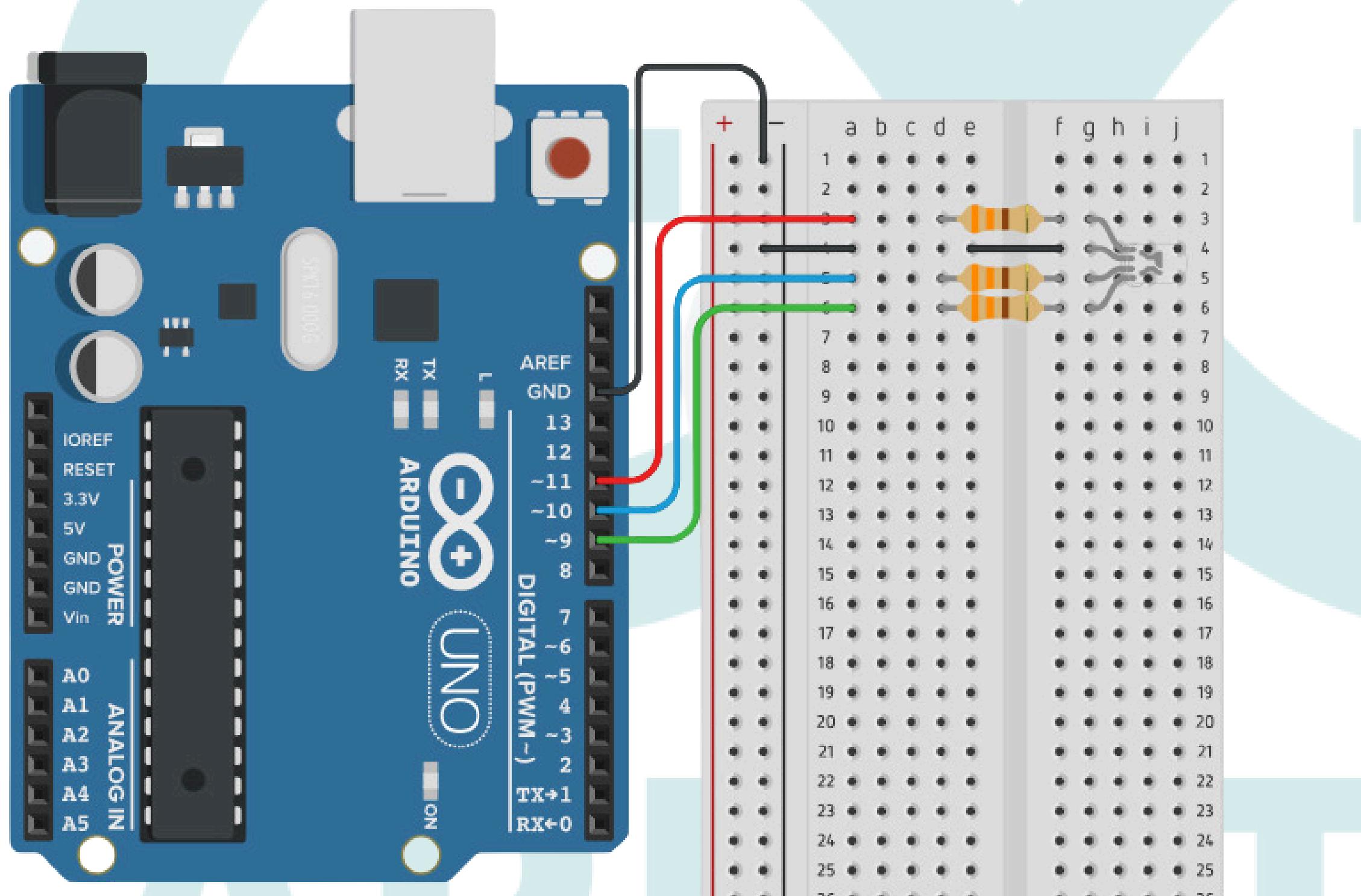
Neste guia, você aprenderá a montar um circuito usando um Arduino para que um LED RGB funcione como um semáforo para pedestres.

Materiais Necessários:

- Arduino Uno, Protoboard e Jumpers.
- 3 resistor de 330 ohms.
- 1 LED RGB.

ARDUINO

Solução Laboratório 01



```
1 int R = 11; // Pino Red - Vermelho
2 int B = 10; // Pino Blue - Azul
3 int G = 9; // Pino Green - Verde
4
5 void setup() {
6     pinMode(R, OUTPUT);
7     pinMode(G, OUTPUT);
8     pinMode(B, OUTPUT);
9 }
10
11 void loop() {
12
13     // VERMELHO
14     digitalWrite(R, HIGH);
15     digitalWrite(G, LOW);
16     digitalWrite(B, LOW);
17     delay(5000);
18
19     // VERDE
20     digitalWrite(R, LOW);
21     digitalWrite(G, HIGH);
22     digitalWrite(B, LOW);
23     delay(5000);
24
25 }
26
```

Laboratório 02

Semáforo de veículos com um LED RGB

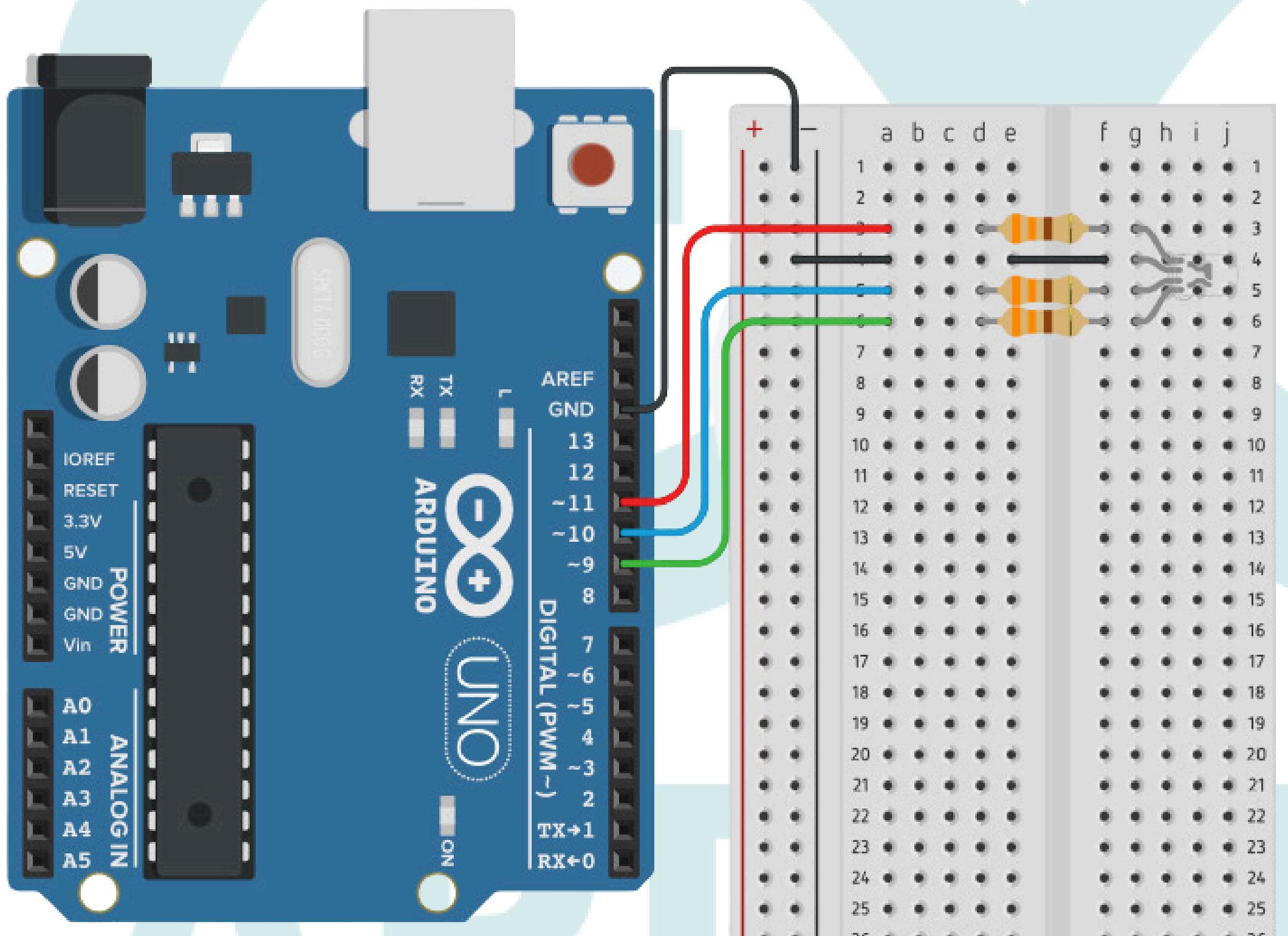
Neste guia, você aprenderá a montar um circuito utilizando um LED RGB para que funcione como um semáforo de veículos.

Materiais Necessários:

- Arduino Uno, Protoboard e Jumpers.
- 1 LED RGB.
- 3 resistor de 330 ohms.

ARDUTENO

Solução Laboratório 02



```
1 int R = 11; // Pino Red - Vermelho
2 int B = 10; // Pino Blue - Azul
3 int G = 9; // Pino Green - Verde
4
5 void setup() {
6     pinMode(R, OUTPUT);
7     pinMode(G, OUTPUT);
8     pinMode(B, OUTPUT);
9 }
0
10 void loop() {
11
12     // VERMELHO
13     digitalWrite(R, HIGH);
14     digitalWrite(G, LOW);
15     digitalWrite(B, LOW);
16     delay(5000);
17
18     // AMARELO
19     digitalWrite(R, HIGH);
20     digitalWrite(G, HIGH);
21     digitalWrite(B, LOW);
22     delay(5000);
23
24     // VERDE
25     digitalWrite(R, LOW);
26     digitalWrite(G, HIGH);
27     digitalWrite(B, LOW);
28     delay(5000);
29 }
30
```

Laboratório 03

Mudança gradual de cor com LED RGB

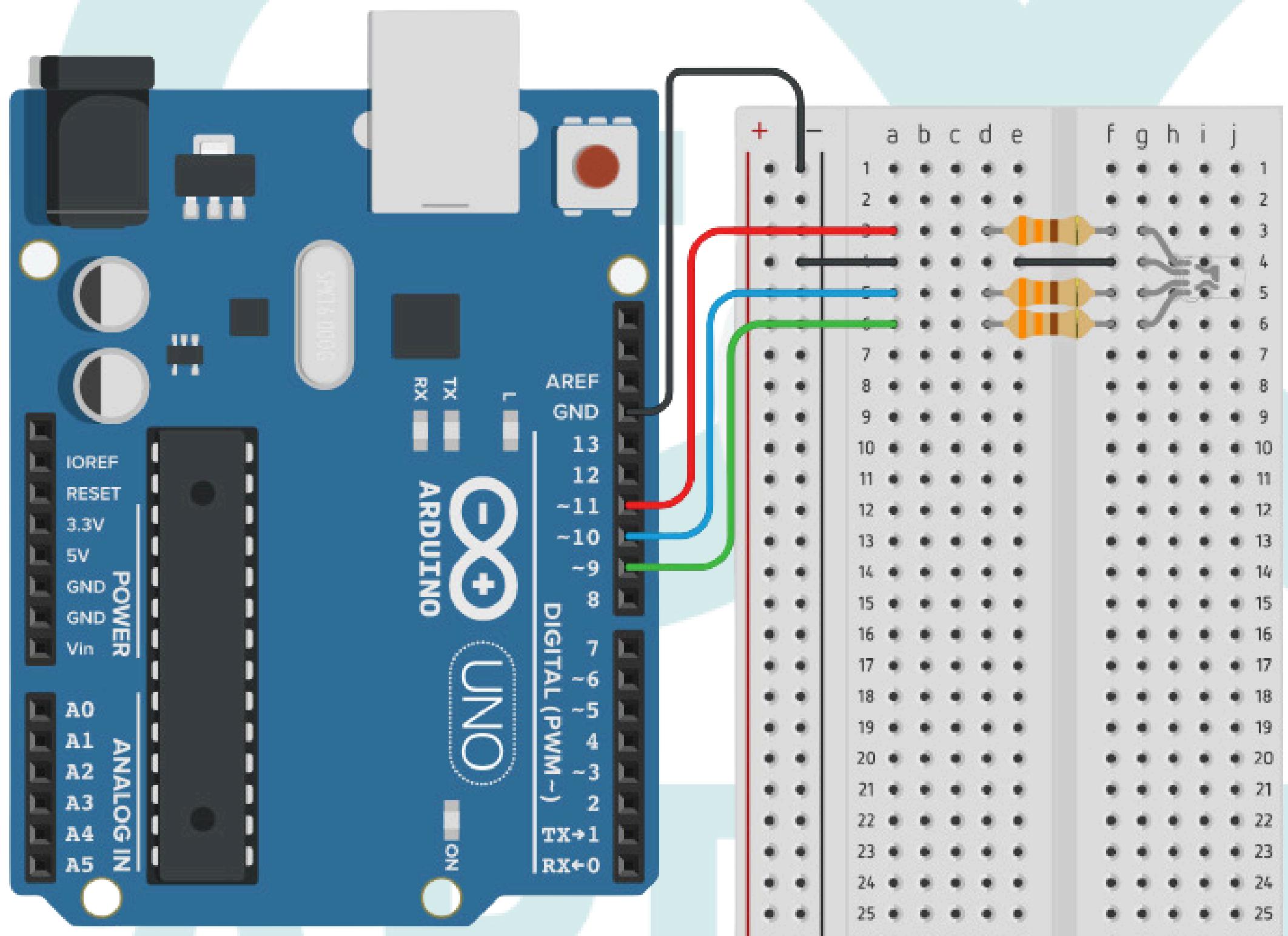
Neste guia, você aprenderá a montar um circuito usando um Arduino para que um LED RGB mude sua cor gradualmente.

Materiais Necessários:

- Arduino Uno, Protoboard e Jumpers.
- 1 LED RGB.
- 3 resistor de 330 ohms.

ARDUTENO

Solução Laboratório 03



```

1 int R = 11; // Pino Red - Vermelho
2 int B = 10; // Pino Blue - Azul
3 int G = 9; // Pino Green - Verde
4
5 void setup() {
6     pinMode(R, OUTPUT);
7     pinMode(G, OUTPUT);
8     pinMode(B, OUTPUT);
9 }
10
11 void loop() {
12
13     for (int i = 0; i <= 255; i++) {
14         analogWrite(R, i);
15         delay(10);
16     }
17     for (int i = 255; i >= 0; i--) {
18         analogWrite(R, i);
19         delay(10);
20     }
21     for (int i = 0; i <= 255; i++) {
22         analogWrite(B, i);
23         delay(10);
24     }
25     for (int i = 255; i >= 0; i--) {
26         analogWrite(B, i);
27         delay(10);
28     }
29     for (int i = 0; i <= 255; i++) {
30         analogWrite(G, i);
31         delay(10);
32     }
33     for (int i = 255; i >= 0; i--) {
34         analogWrite(G, i);
35         delay(10);
36     }
37 }
```

Laboratório 04

Mudança de cor de um LED RGB com a utilização de um potenciômetro

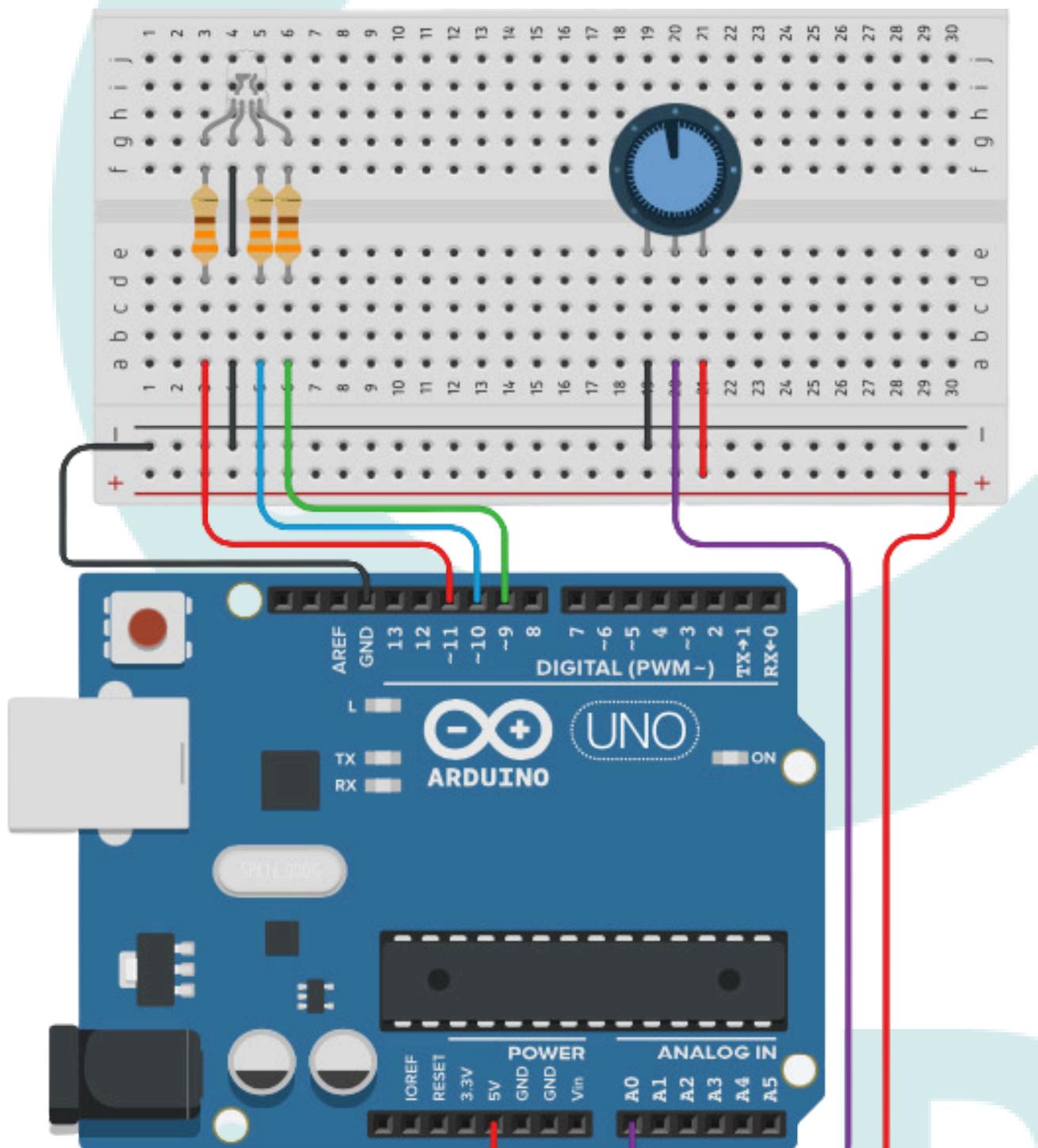
Neste guia, você aprenderá a montar um circuito utilizando um LED RGB que muda sua cor com a utilização de um potenciômetro.

Materiais Necessários:

- Arduino Uno, Protoboard e Jumpers.
- 1 LED RGB.
- 3 resistor de 330 ohms.
- 1 Potenciômetro.

ARDUTENO

Solução Laboratório 04



```

1 int R = 11; // Pino Red - Vermelho
2 int B = 10; // Pino Blue - Azul
3 int G = 9; // Pino Green - Verde
4 int potenciometro = A0;
5 int valor_pot;

6
7 void setup() {
8     pinMode(R, OUTPUT);
9     pinMode(G, OUTPUT);
10    pinMode(B, OUTPUT);
11 }

12
13 void loop() {
14     valor_pot = analogRead(potenciometro); // Valor pode variar entre 0 e 1023
15
16     if(valor_pot >= 0 && valor_pot <= 256) // Se o valor lido for >= 0 e menor igual a 256, o LED permanece apagado
17     {
18         digitalWrite(R, LOW);
19         digitalWrite(G, LOW);
20         digitalWrite(B, LOW);
21     }
22     if(valor_pot > 256 && valor_pot <= 512) // Se o valor lido for maior que 256 e menor igual a 512, o LED terá a cor vermelha
23     {
24         digitalWrite(R, HIGH);
25         digitalWrite(G, LOW);
26         digitalWrite(B, LOW);
27     }
28     if(valor_pot > 512 && valor_pot <= 768) // Se o valor lido for maior que 512 e menor igual a 768, o LED terá a cor verde
29     {
30         digitalWrite(R, LOW);
31         digitalWrite(G, HIGH);
32         digitalWrite(B, LOW);
33     }
34     if(valor_pot > 768 && valor_pot <= 1023) // Se o valor lido for maior que 768 e menor igual a 1023, o LED terá a cor azul
35
36         digitalWrite(R, LOW);
37         digitalWrite(G, LOW);
38         digitalWrite(B, HIGH);
39     }
40 }
```

Laboratório 05

Mudança de cor de um LED RGB com a utilização de um potenciômetro e acendendo um LED azul

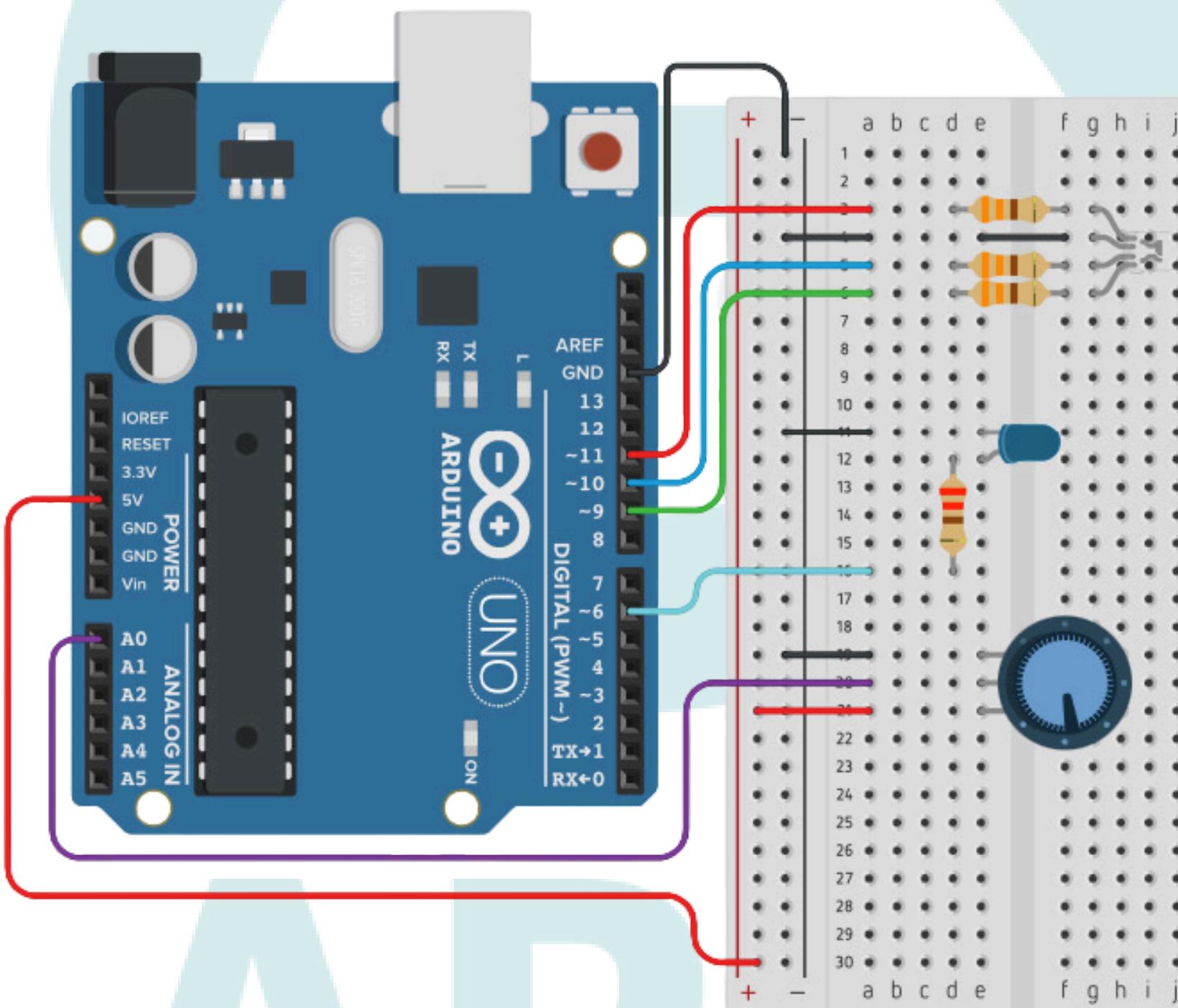
Neste guia, você aprenderá a montar um circuito utilizando um LED RGB que muda sua cor com a utilização de um potenciômetro e toda vez que o LED RGB estiver com a cor azul um LED azul irá acender.

Materiais Necessários:

- Arduino Uno, Protoboard e Jumpers.
- 1 LED azul.
- 1 resistor de 220 ohms (para o LED azul).
- 3 resistor de 330 ohms.
- 1 LED RGB.
- 1 Potenciômetro.

ARDUTINO

Solução Laboratório 05



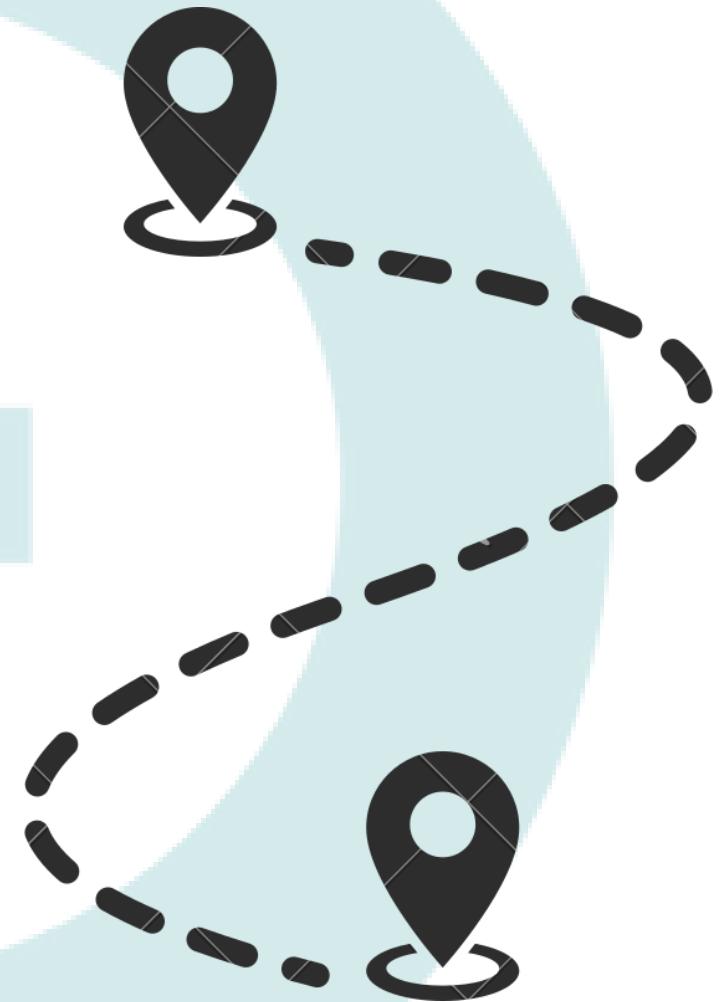
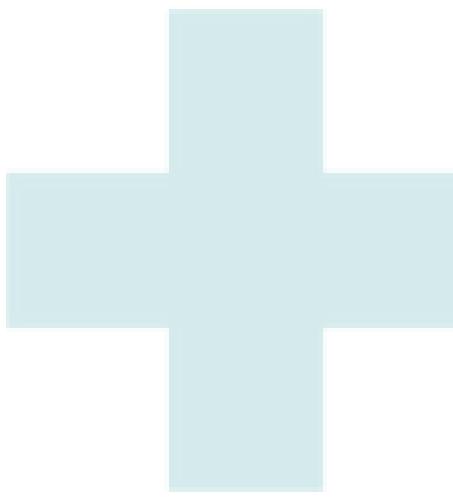
```

1 int R = 11; // Pino Red - Vermelho
2 int B = 10; // Pino Blue - Azul
3 int G = 9; // Pino Green - Verde
4 int potenciometro = A0;
5 int led_azul = 6;
6 int valor_pot;
7
8 void setup() {
9     pinMode(R, OUTPUT);
10    pinMode(G, OUTPUT);
11    pinMode(B, OUTPUT);
12    pinMode(led_azul, OUTPUT);
13 }
14
15 void loop() {
16     valor_pot = analogRead(potenciometro); // Valor pode variar entre 0 e 1023
17
18     if(valor_pot >= 0 && valor_pot <= 256) // Se o valor lido for >= 0 e menor igual a 256, o LED permanece apagado
19     {
20         digitalWrite(R, LOW);
21         digitalWrite(G, LOW);
22         digitalWrite(B, LOW);
23     }
24     if(valor_pot > 256 && valor_pot <= 512) // Se o valor lido for maior que 256 e menor igual a 512, o LED tera a cor vermelha
25     {
26         digitalWrite(R, HIGH);
27         digitalWrite(G, LOW);
28         digitalWrite(B, LOW);
29     }
30     if(valor_pot > 512 && valor_pot <= 768) // Se o valor lido for maior que 512 e menor igual a 768, o LED tera a cor verde
31     {
32         digitalWrite(R, LOW);
33         digitalWrite(G, HIGH);
34         digitalWrite(B, LOW);
35     }
36     if(valor_pot > 768 && valor_pot <= 1023) // Se o valor lido for maior que 768 e menor igual a 1023, o LED tera a cor azul
37     {
38         digitalWrite(R, LOW);
39         digitalWrite(G, LOW);
40         digitalWrite(B, HIGH);
41         digitalWrite(led_azul, HIGH); //Acende e Desliga o LED azul
42         digitalWrite(led_azul, LOW);
43     }
44 }

```

Roteiro

1. Comunicação Serial
2. Portas Tx e Rx
3. Funções
4. Display de 7 Segmentos
 - Cátodo Comum
 - Ânodo Comum
5. Buzzer
 - Buzzer Ativo
 - Buzzer Passivo
6. Laboratórios

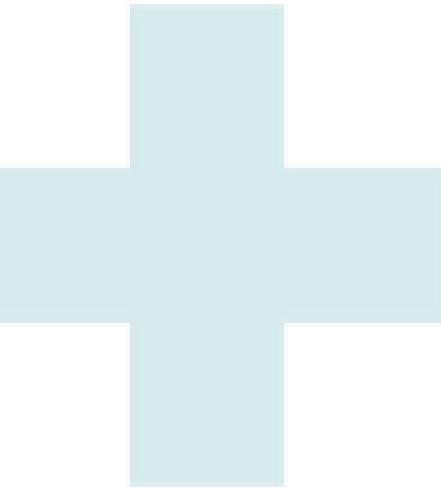


ARDUTENO

Comunicação Serial

- **Definição:**

- Troca sequencial de dados entre dispositivos.
- Utilização de **UART**, SPI e I2C no Arduino.



- **Aplicações Cotidianas:**

- USB para teclados, mouses.
- IoT: termômetros, leitores RFID.

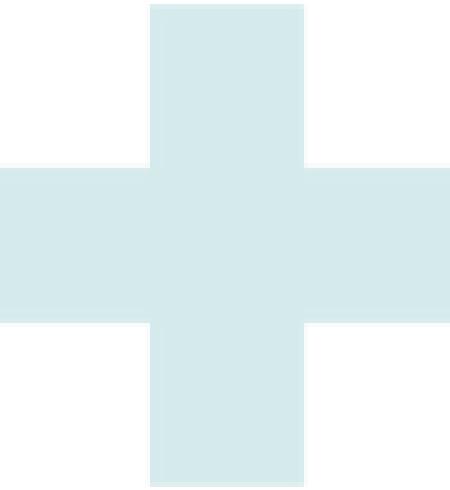
- **Importância:**

- Facilita interconexão, aplicações práticas.

ARDUTENO

Configuração e Baud Rate

- **Baud Rate:**
 - Número de bits por segundo.
 - Configurado com `Serial.begin()`.
- **Configuração Exemplo:**
 - `Serial.begin(9600)`.
- **Facilita Comunicação:**
 - Essencial para comunicação serial no Arduino.



ARDUINO

Funções Básicas

Para realizar os primeiros programas usando a serial, vamos utilizar os seguintes comandos print e read:

- **Função Serial.print e Serial.println():** Função Serial.print é usada para envio de informações do Arduino para o PC e a sintaxe é dada por:
 - Serial.print(Argumento);
 - Argumento pode ser um texto (sempre enviado entre "") ou uma variável.
 - A função Serial.println tem a mesma proposta da anterior, o seu diferencial é que após o argumento será acrescentado uma quebra de linha.
- **Função Serial.read:** É utilizada para o envio de informações do PC para o Arduino.
 - O valor enviado deve ser armazenado em uma variável.
- **Exemplo:** Enviando uma variável do PC para o Arduino

Exemplos print e read

```
1 void setup()
2 {
3     Serial.begin(9600); // inicializando comunicacao serial com Baud rate de 9600
4 }
5
6 void loop() {
7 {
8     Serial.print("Aula sobre comunicacao serial :)");
9 }
```

 Monitor serial

Aula sobre comunicacao serial :)Aula sobre comunicacao serial :)Aula sobre comunicacao serial :)Aula sobre comunicacao serial :)Aula sobre

```
1 void setup()
2 {
3     Serial.begin(9600); // inicializando comunicacao serial com Baud rate de 9600
4 }
5
6 void loop()
7 {
8     int x;
9     Serial.print(x);
10 }
```

```
1 void setup()
2 {
3     Serial.begin(9600); // inicializando comunicacao serial com Baud rate de 9600
4 }
5
6 void loop()
7 {
8     Serial.println("Aula sobre comunicacao serial :)");
9 }
```

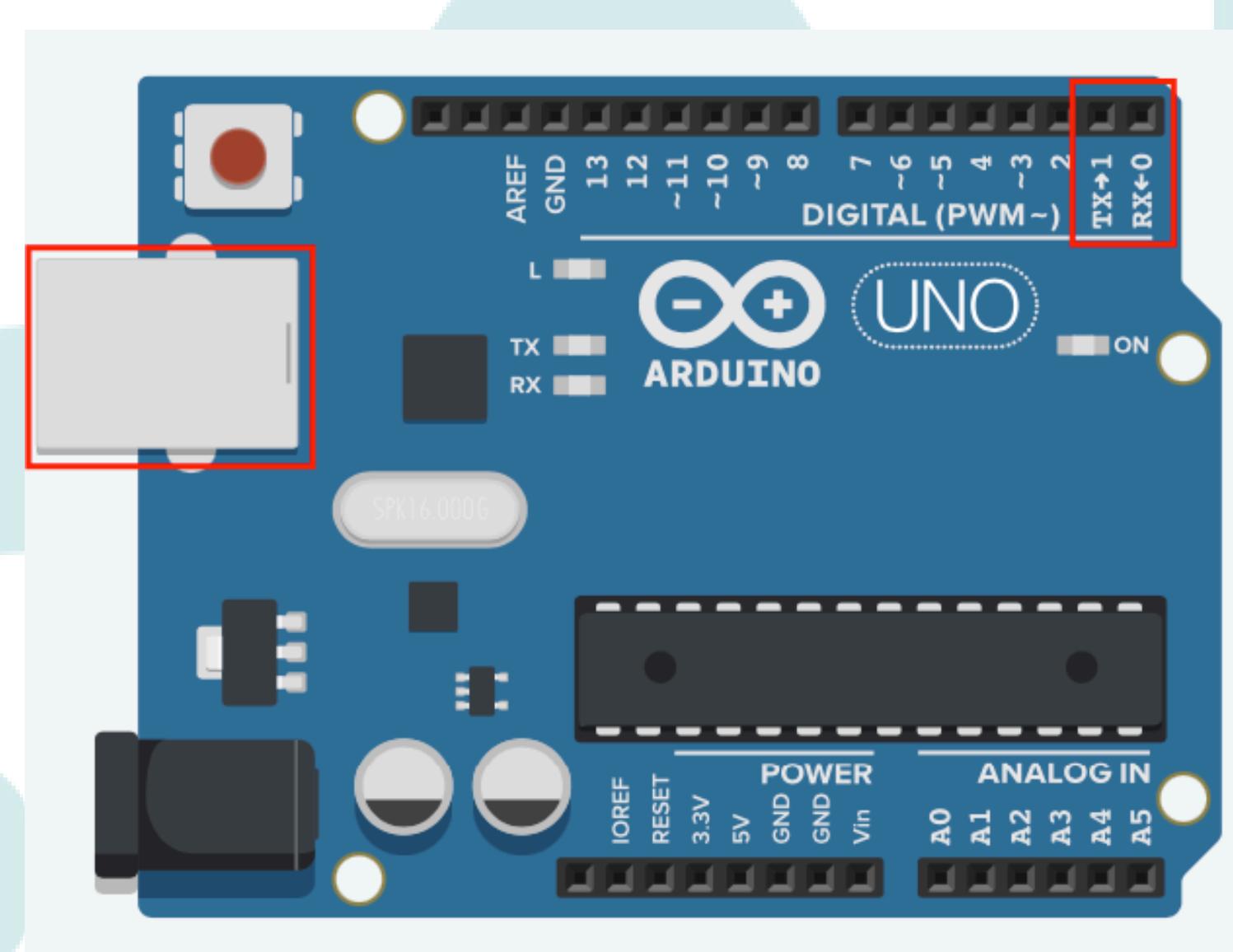
Portas TX e RX na Comunicação Serial

- **TX (Transmissão) e RX (Recepção):** Papéis fundamentais na comunicação serial.
 - **TX:** Transmite dados de um dispositivo.
 - **RX:** Recebe dados em outro dispositivo.
- **Arduino e Comunicação Serial:** Utiliza TX e RX para enviar e receber informações.
 - **Exemplo:** Conectar sensor ao Arduino.
- **Comunicação Bidirecional:**
 - Permite troca eficiente de dados entre dispositivos.
- **Importância em Dispositivos Eletrônicos:**
 - Módulos Bluetooth, GPS, sensores.
 - Essencial para o funcionamento adequado.

Portas TX e RX na Comunicação Serial

- **Desenvolvimento de Projetos Eletrônicos:**

- Compreender o papel das portas TX e RX é vital.
- Fundamentais para projetos envolvendo comunicação serial.



ARDUINO

Importância das Funções na Programação

- **Definição:**

- Blocos de código reutilizáveis e modularizados.
- Desempenham tarefas específicas em um programa.

- **Organização e Estruturação:**

- Essenciais para organizar e estruturar o código.
- Promovem clareza e eficiência no desenvolvimento de software.

- **Linguagens de Programação:**

- Presentes em C, C++, Python, JavaScript.

- **Parâmetros e Resultados:**

- Recebem parâmetros como entrada.
- Processam internamente e podem retornar um resultado.

ARDUINO

- **Facilitam a Escrita de Código:**

- Código mais conciso e legível.
- Tarefas complexas divididas em partes gerenciáveis.

```
int nomeFuncao(){...}
float nomeFuncao(){...}
char nomeFuncao(){...}
void nomeFuncao(){...}
```

ARDUINO

Benefícios da Utilização de Funções

- **Reutilização de Código:**

- Criação de funções permite reutilizar código.
- Evita redundâncias em diferentes partes do programa.

- **Atualização e Correção:**

- Facilita a atualização ou correção de funcionalidades específicas.
- Alterações em uma função refletem em todas as suas chamadas.

- **Contribuição para Modularidade:**

- Torna o código mais fácil de entender e manter.
- Cada função representa uma unidade modular de funcionalidade.

- **Essenciais no Desenvolvimento:**

- Fundamentais para o desenvolvimento eficiente e estruturado de programas.

Descrição

- A função `int` é utilizada para manipulação de números inteiros, retornando um valor inteiro.
- A função `float` é utilizada para a manipulação de números decimais, retornando um valor ponto flutuante.
- A função `char` é utilizada para manipulação de caracteres, retornando um valor do tipo caractere.
- A função `void` não retorna nenhum valor. Ela é utilizada para realização de uma tarefa específica ou executar um conjunto de instruções, como por exemplo salvar as configurações para acender LEDs que serão acesos em um display. Será de bastante utilidade para trabalharmos com o display de 7 Segmentos.

ARDUTINO

Display de 7 Segmentos

- **Definição:**

- Componente eletrônico para exibir números, letras e caracteres alfanuméricos.
- Consiste em sete LEDs dispostos para formar dígitos.

- **Configuração:**

- Cada LED representa um segmento.
- Permite exibição de números de 0 a 9 e algumas letras.

- **Conexão com Arduino:**

- Atribuição de pinos específicos do display a pinos digitais da placa.
- Uso de resistores limitadores de corrente para cada segmento e um para o pino comum.

- **Comunicação via Programação:**

- Controle dos LEDs acesos para representar o caractere desejado.

Aplicações do Display de 7 Segmentos

- **Desenvolvimento de Projetos:**

- Comum em projetos Arduino para exibir valores numéricos.

- **Utilização Prática:**

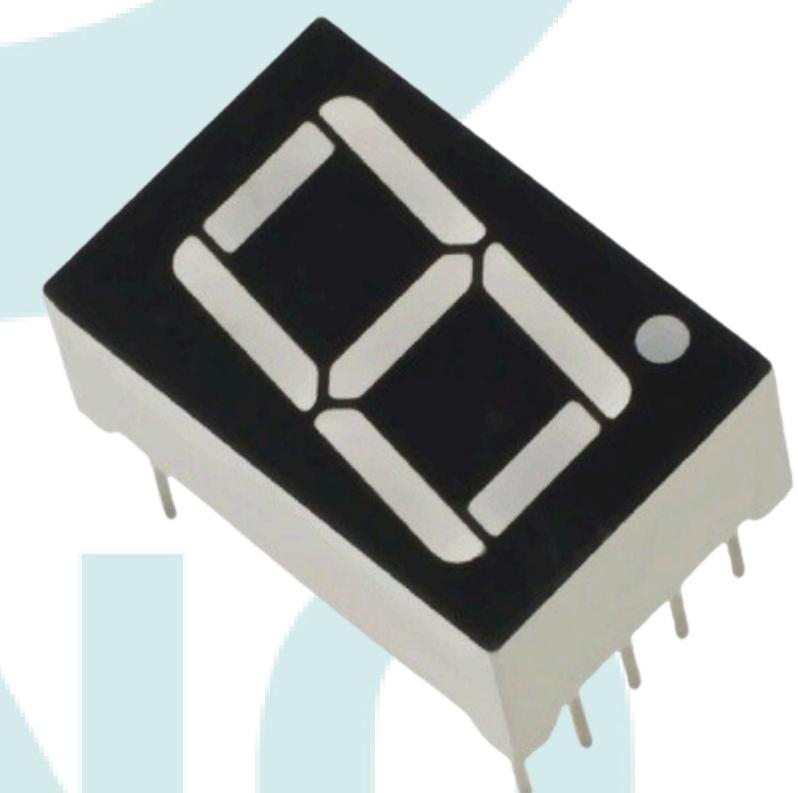
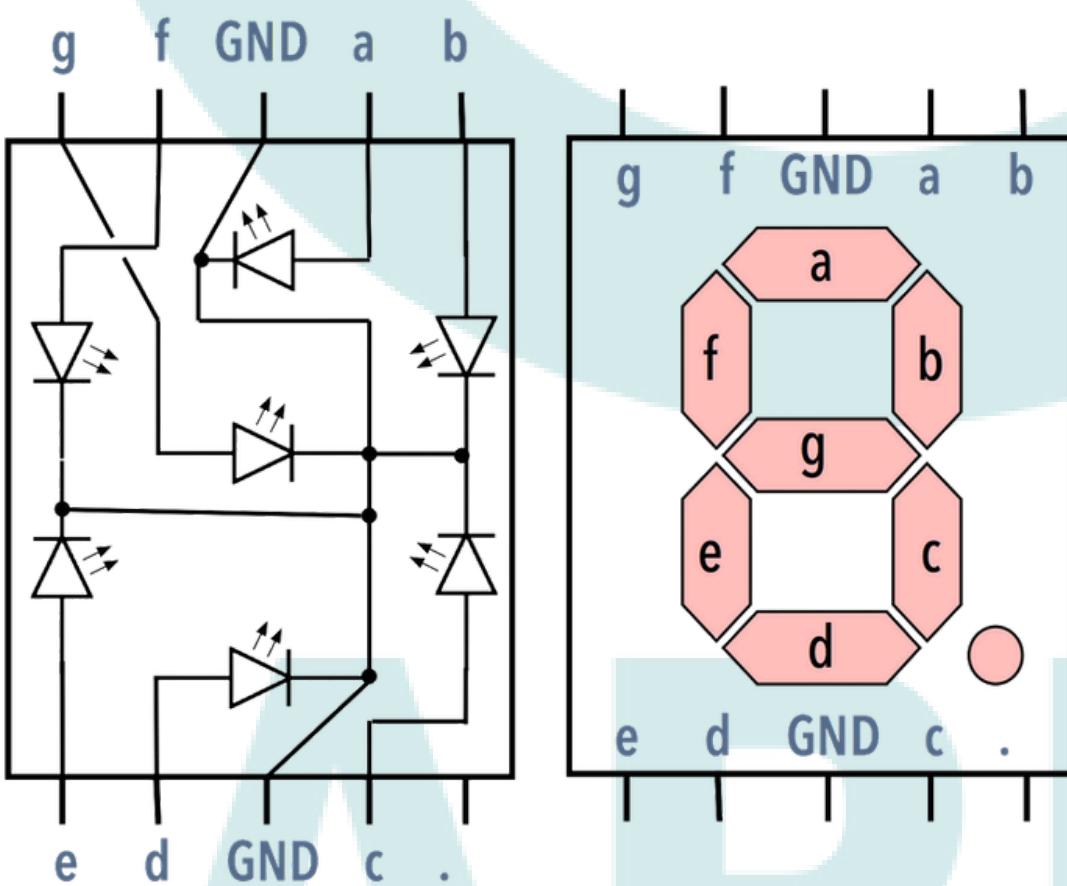
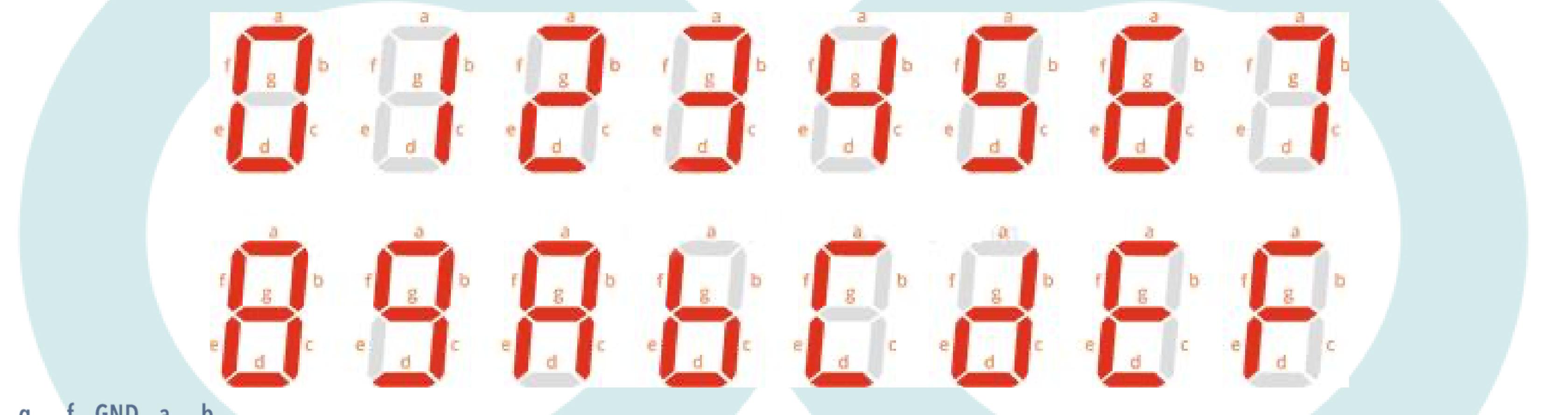
- Exibição da leitura de sensores.
- Contadores e medições numéricas.

- **Simplicidade e Versatilidade:**

- Fácil integração e programação.
- Escolha comum em aplicações que demandam exibição de informações numéricas.

ARDUTINO

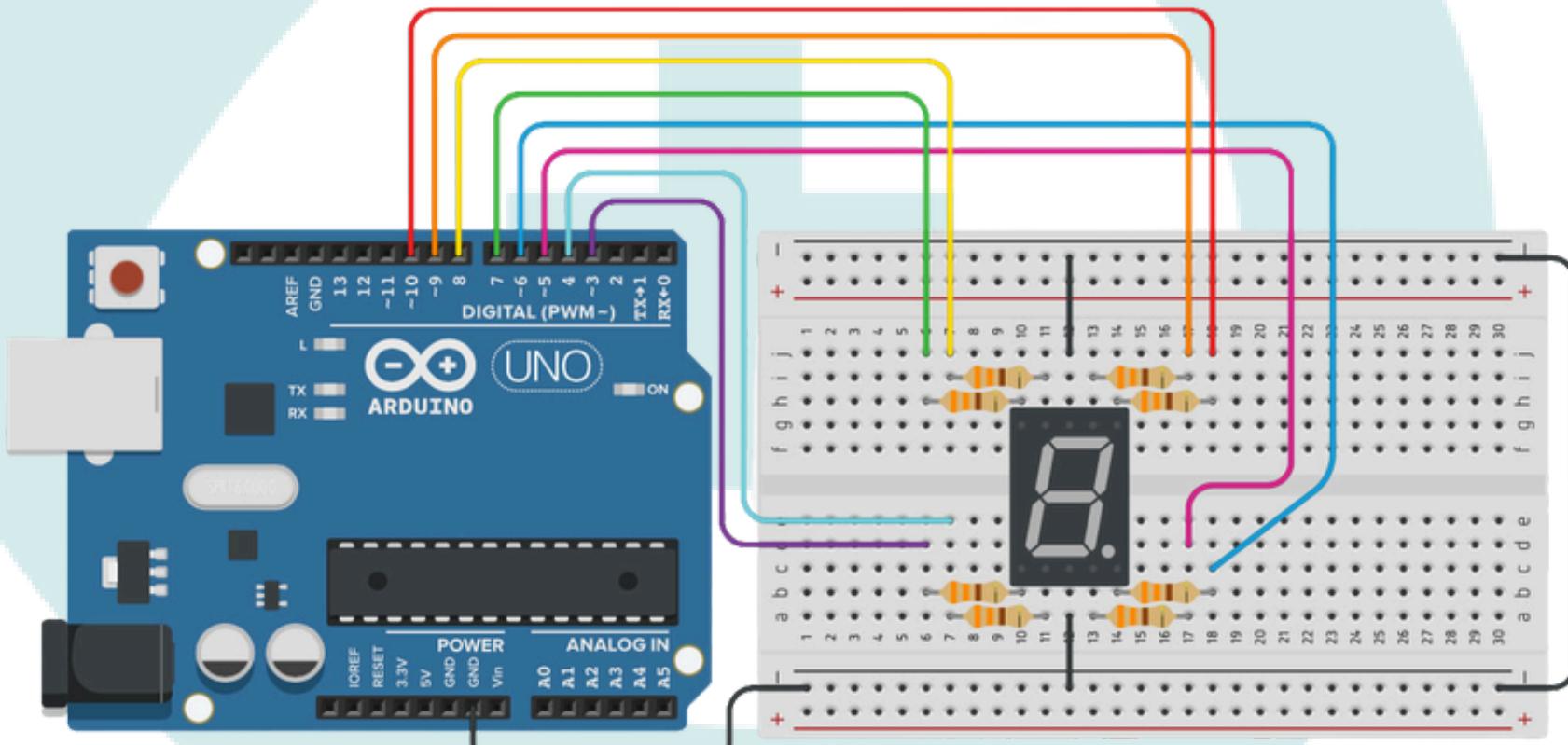
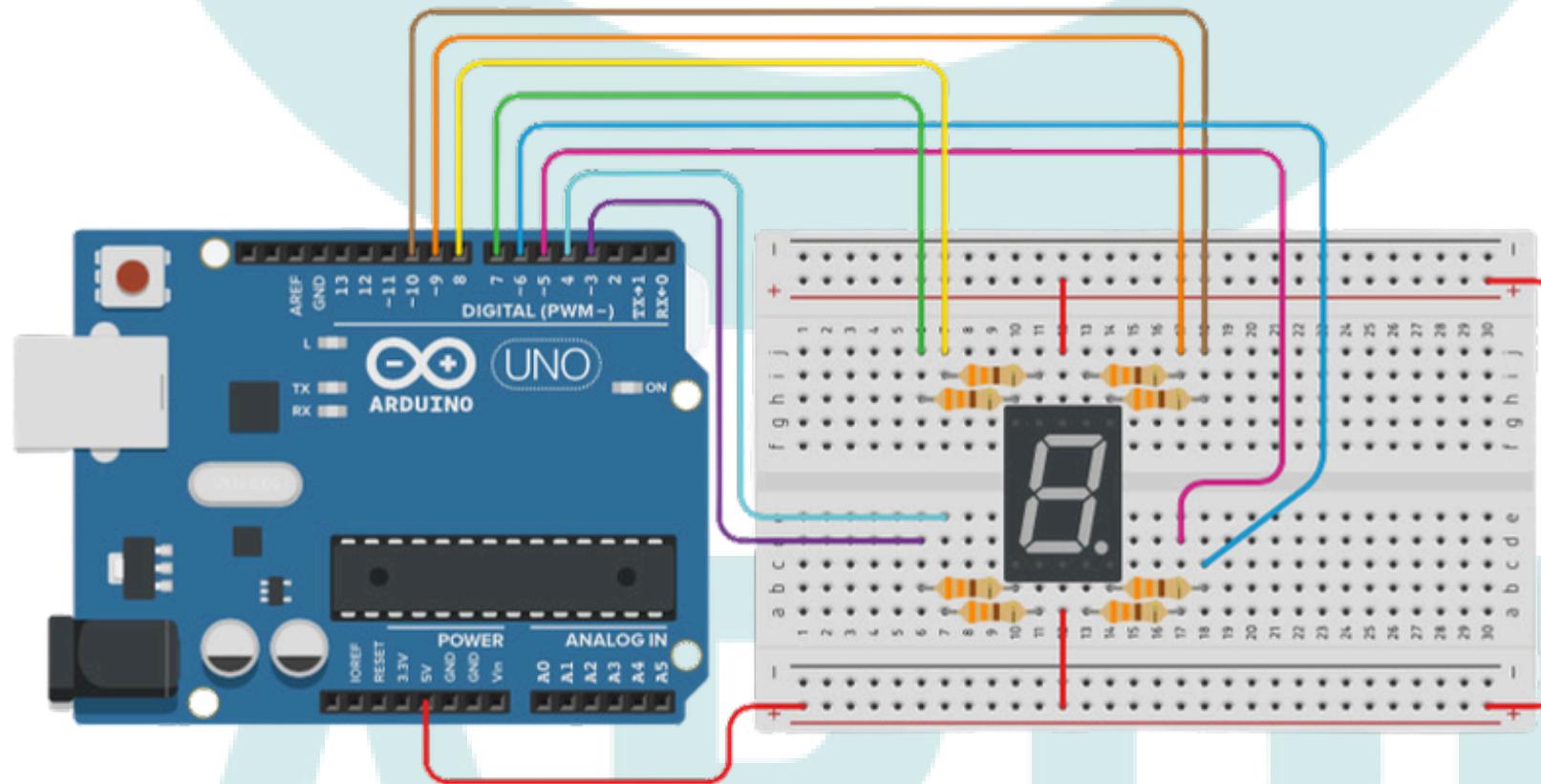
®



ARDUINO

Configurações

- Cátodo Comum



- Ânodo Comum

Exemplo Prático

- **Objetivo do Exemplo:**
 - Exibir os números 1 e 7 no display de 7 segmentos, seguido pelo desligamento de todos os LEDs (estado LOW).
 - **Configuração Específica:**
 - Utilização da configuração cátodo comum.
 - **Implementação com Funções:**
 - Definir funções para exibir o número 1, número 7 e desligar o display.

```
1 // declaracao dos pinos do display de 7 segmentos      36 // ====== FUNCOES
2 int A = 9;                                         37 void um()
3 int B = 10;                                         38 {
4 int C = 5;                                         39   digitalWrite(A, LOW);
5 int D = 4;                                         40   digitalWrite(B, HIGH);
6 int E = 3;                                         41   digitalWrite(C, HIGH);
7 int F = 8;                                         42   digitalWrite(D, LOW);
8 int G = 7;                                         43   digitalWrite(E, LOW);
9                                         44   digitalWrite(F, LOW);
10 void setup()                                         45   digitalWrite(G, LOW);
11 {                                                 46 }
12   pinMode(A, OUTPUT);                                47 void sete()
13   pinMode(B, OUTPUT);                                48 {
14   pinMode(C, OUTPUT);                                49   digitalWrite(A, HIGH);
15   pinMode(D, OUTPUT);                                50   digitalWrite(B, HIGH);
16   pinMode(E, OUTPUT);                                51   digitalWrite(C, HIGH);
17   pinMode(F, OUTPUT);                                52   digitalWrite(D, LOW);
18   pinMode(G, OUTPUT);                                53   digitalWrite(E, LOW);
19 }                                                 54   digitalWrite(F, LOW);
20                                         55   digitalWrite(G, LOW);
21 void loop()                                         56 }
22 {                                                 57 void desliga()
23   // chamando a funcao correspondente ao numero 1    58 {
24   um();                                         59   digitalWrite(A, LOW);
25   delay(1000);                                     60   digitalWrite(B, LOW);
26   // chamando a funcao correspondente ao numero 7    61   digitalWrite(C, LOW);
27   sete();                                         62   digitalWrite(D, LOW);
28   delay(1000);|                                     63   digitalWrite(E, LOW);
29                                         64   digitalWrite(F, LOW);
30   while(true){                                     65   digitalWrite(G, LOW);
31     // uma das opcoes para que o display permaneca desligado
32     desliga();                                     66 }
```

Buzzer

- **Definição:**

- Componente eletrônico que converte sinais elétricos em ondas sonoras audíveis.

- **Tipos de Buzzers:**

- **Passivo:** Requer onda quadrada para produzir som.
 - Composto por diafragma móvel e bobina.
 - **Ativo:** Contém circuito interno para gerar onda sonora.
 - Pode ser controlado por sinal lógico.

- **Aplicações Comuns:**

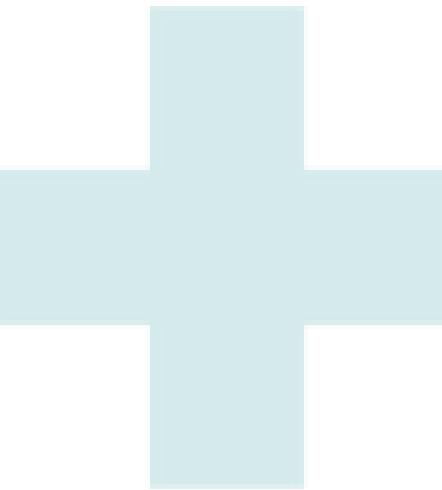
- Projetos Arduino para feedback sonoro.
 - Alarmes, indicadores sonoros, jogos, etc.

ARDUINO

Funcionamento e Acionamento

- **Buzzer Passivo:**

- Requer sinal de onda quadrada.
- Vibração do diafragma e bobina geram som.



- **Buzzer Ativo:**

- Contém circuito interno para gerar som.
- Pode ser controlado diretamente por sinal lógico.

- **Acionamento no Arduino:**

- Ondas quadradas pelos pinos digitais.
- Comando `tone()`: `tone(pino, frequência em Hz, duração em ms)`.

ARDUTINO

Frequências das Notas Musicais

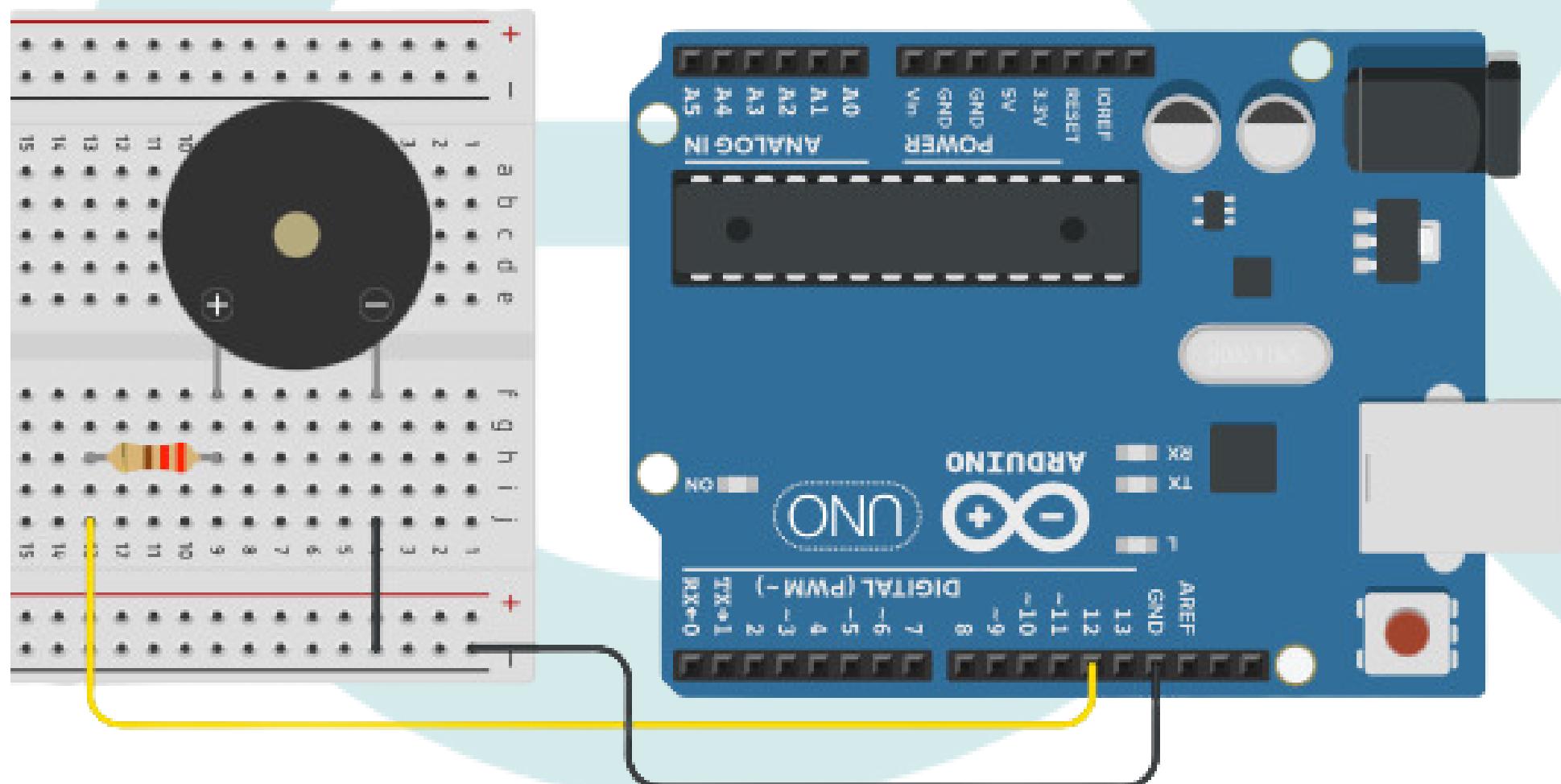
- **Notas Musicais e Frequências:**

- Cada nota musical possui uma frequência específica.
 - Dó: 262 Hz
 - Ré: 294 Hz
 - Mi: 330 Hz
 - Fá: 349 Hz
 - Sol: 392 Hz
 - Lá: 440 Hz
 - Si: 494 Hz
 - #Dó: 528 Hz

- **Programação no Arduino:**

- Utilização do comando `tone()` para gerar frequências desejadas.

Exemplo Prático



```
1 int buzzer = 12;
2 void setup() {
3     pinMode(buzzer, OUTPUT);
4 }
5
6 void loop() {
7     // aciona o buzzer com a nota Ré
8     tone(buzzer, 294, 500);
9     delay(1000);
10    // função que desativa o buzzer
11    noTone(buzzer);
12    // nota Lá
13    tone(buzzer, 440, 600);
14    delay(1000);
15 }
```

Laboratório 01

Mostrando os dígitos de 0 a 9 no Display de 7 segmentos

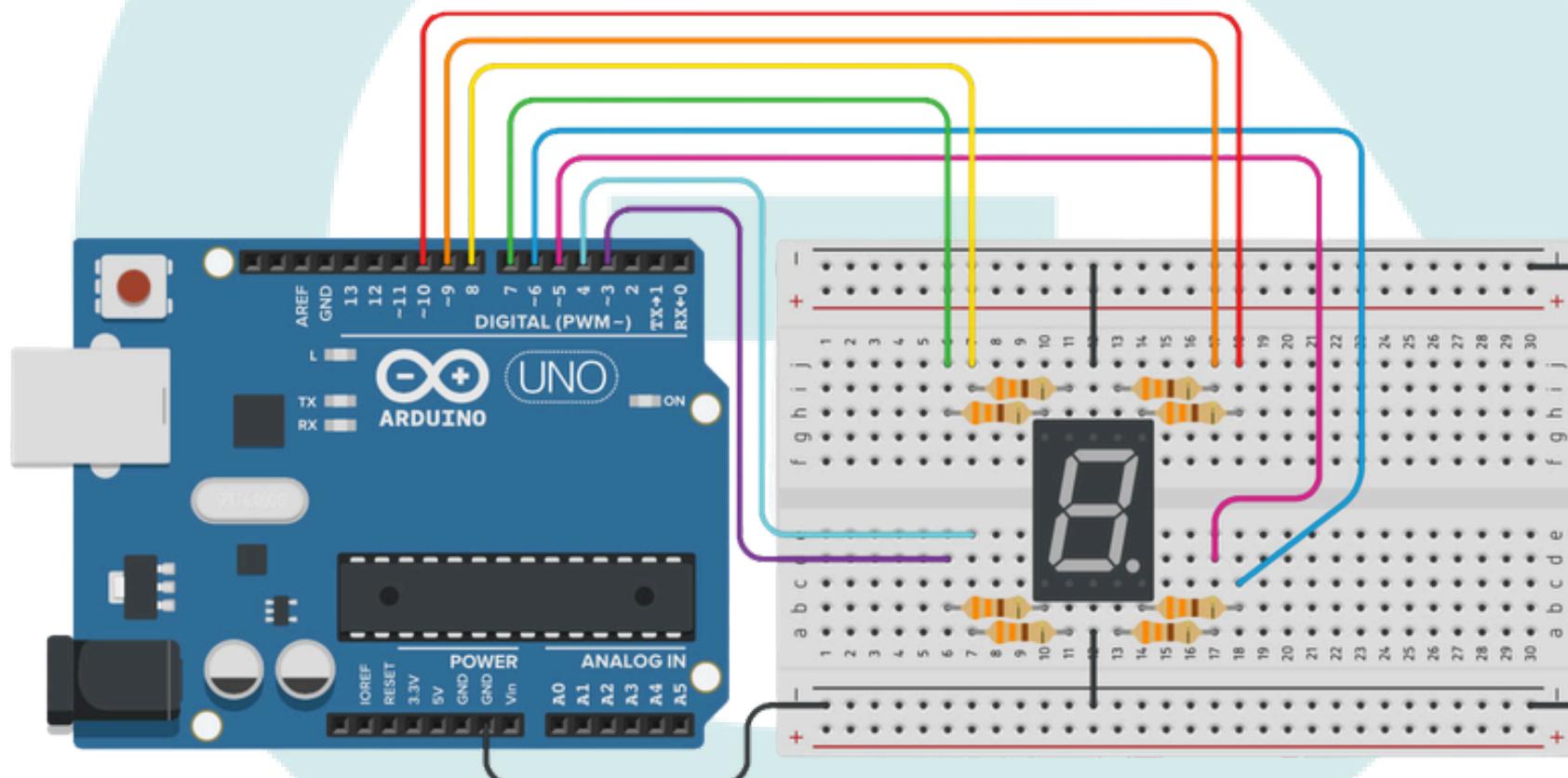
Neste guia, você aprenderá a montar um circuito utilizando um display de 7 segmentos para mostrar os dígitos de 0 a 9.

Materiais Necessários:

- Arduino UNO, Protoboard e Jumpers.
- 8 Resistores de 220 ohms.
- 1 Display de 7 Segmentos.

ARDUTINO

Solução Laboratório 01



```

124 void nove()
125 {
126     digitalWrite(A, HIGH);
127     digitalWrite(B, HIGH);
128     digitalWrite(C, HIGH);
129     digitalWrite(D, HIGH);
130     digitalWrite(E, LOW);
131     digitalWrite(F, HIGH);
132     digitalWrite(G, HIGH);
133 }
134 void desliga()
135 {
136     digitalWrite(A, LOW);
137     digitalWrite(B, LOW);
138     digitalWrite(C, LOW);
139     digitalWrite(D, LOW);
140     digitalWrite(E, LOW);
141     digitalWrite(F, LOW);
142     digitalWrite(G, LOW);
143 }

18 void loop()
19 {
20     um();
21     delay(1000);
22     dois();
23     delay(1000);
24     tres();
25     delay(1000);
26     quatro();
27     delay(1000);
28     cinco();
29     delay(1000);
30     seis();
31     delay(1000);
32     sete();
33     delay(1000);
34     oito();
35     delay(1000);
36     nove();
37     delay(1000);

39 while(true){
40     //uma das opcoes para que o display permaneca desligado
41     desliga();
42 }
43

```

```

1 int A = 9;
2 int B = 10;
3 int C = 5;
4 int D = 4;
5 int E = 3;
6 int F = 8;
7 int G = 7;
8 void setup()
9 {
10     pinMode(A, OUTPUT);
11     pinMode(B, OUTPUT);
12     pinMode(C, OUTPUT);
13     pinMode(D, OUTPUT);
14     pinMode(E, OUTPUT);
15     pinMode(F, OUTPUT);
16     pinMode(G, OUTPUT);
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

```

```

44 void um()
45 {
46     digitalWrite(A, LOW);
47     digitalWrite(B, HIGH);
48     digitalWrite(C, HIGH);
49     digitalWrite(D, LOW);
50     digitalWrite(E, LOW);
51     digitalWrite(F, LOW);
52     digitalWrite(G, LOW);
53 }
54 void dois()
55 {
56     digitalWrite(A, HIGH);
57     digitalWrite(B, LOW);
58     digitalWrite(C, HIGH);
59     digitalWrite(D, HIGH);
60     digitalWrite(E, HIGH);
61     digitalWrite(F, LOW);
62     digitalWrite(G, HIGH);
63 }
64 void tres()
65 {
66     digitalWrite(A, HIGH);
67     digitalWrite(B, HIGH);
68     digitalWrite(C, HIGH);
69     digitalWrite(D, HIGH);
70     digitalWrite(E, LOW);
71     digitalWrite(F, LOW);
72     digitalWrite(G, HIGH);
73 }
74 void quatro()
75 {
76     digitalWrite(A, LOW);
77     digitalWrite(B, HIGH);
78     digitalWrite(C, HIGH);
79     digitalWrite(D, LOW);
80     digitalWrite(E, LOW);
81     digitalWrite(F, HIGH);
82     digitalWrite(G, HIGH);
83 }

84 void cinco()
85 {
86     digitalWrite(A, HIGH);
87     digitalWrite(B, LOW);
88     digitalWrite(C, HIGH);
89     digitalWrite(D, HIGH);
90     digitalWrite(E, LOW);
91     digitalWrite(F, HIGH);
92     digitalWrite(G, HIGH);
93 }
94 void seis()
95 {
96     digitalWrite(A, HIGH);
97     digitalWrite(B, LOW);
98     digitalWrite(C, HIGH);
99     digitalWrite(D, HIGH);
100    digitalWrite(E, HIGH);
101    digitalWrite(F, HIGH);
102    digitalWrite(G, HIGH);
103 }
104 void sete()
105 {
106     digitalWrite(A, HIGH);
107     digitalWrite(B, HIGH);
108     digitalWrite(C, HIGH);
109     digitalWrite(D, LOW);
110     digitalWrite(E, LOW);
111     digitalWrite(F, LOW);
112     digitalWrite(G, LOW);
113 }
114 void oito()
115 {
116     digitalWrite(A, LOW);
117     digitalWrite(B, HIGH);
118     digitalWrite(C, HIGH);
119     digitalWrite(D, HIGH);
120     digitalWrite(E, HIGH);
121     digitalWrite(F, HIGH);
122     digitalWrite(G, HIGH);
123 }

```

Laboratório 02

Contagem regressiva do 5 ao 0 com a emissão de alerta sonoro

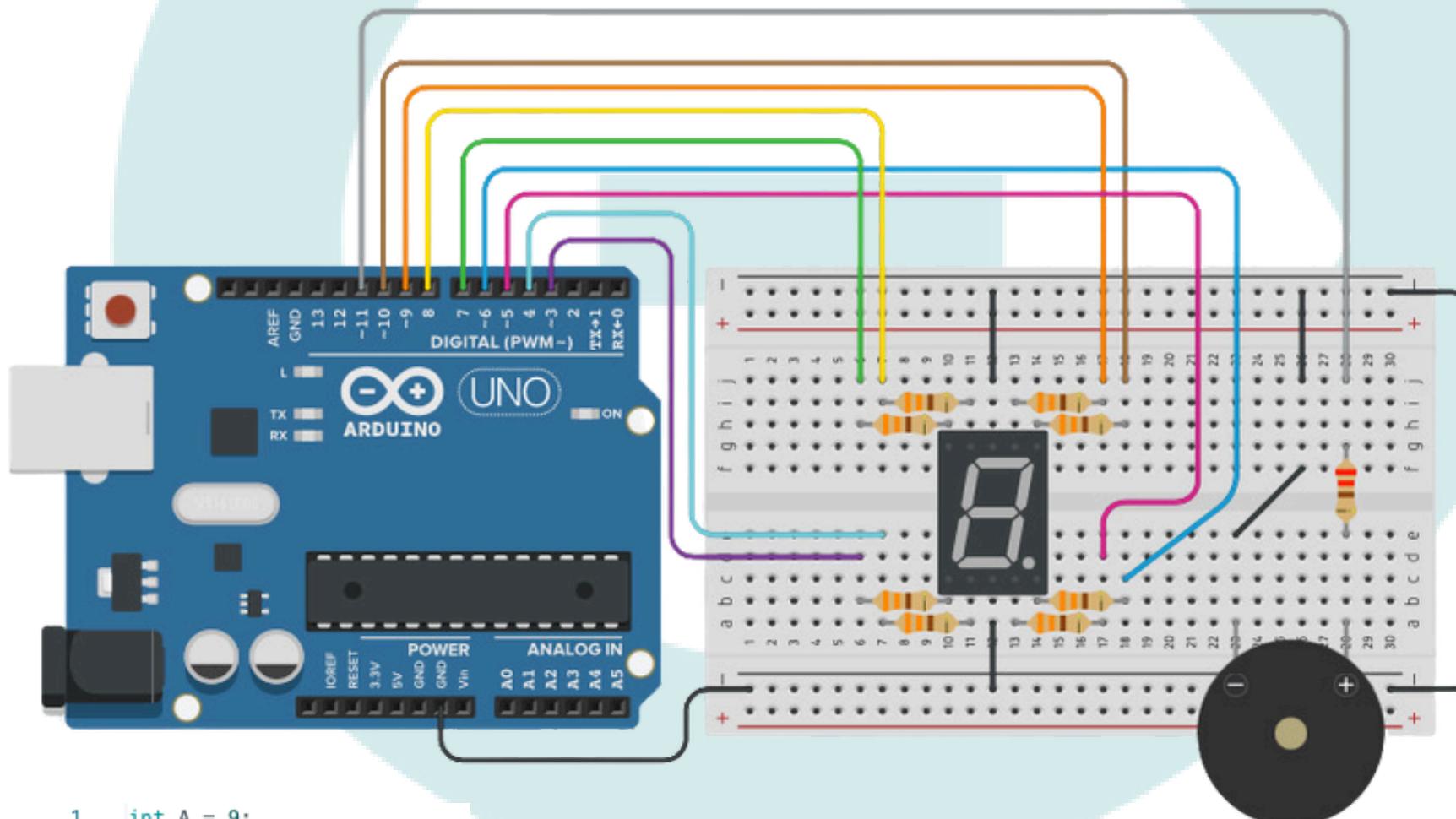
Neste guia, você aprenderá a montar um circuito que faz contagem regressiva com display de 7 segmentos e emite um alerta sonoro.

Materiais Necessários:

- Arduino UNO, Protoboard e Jumpers.
- 8 Resistores de 330 ohms.
- 1 Resistor de 220 ohms.
- 1 Display de 7 Segmentos.

ARDUTENO

Solução Laboratório 02



```

1 int A = 9;
2 int B = 10;
3 int C = 5;
4 int D = 4;
5 int E = 3;
6 int F = 8;
7 int G = 7;
8 int buzzer = 11;
9 void setup()
10 {
11   pinMode(A, OUTPUT);
12   pinMode(B, OUTPUT);
13   pinMode(C, OUTPUT);
14   pinMode(D, OUTPUT);
15   pinMode(E, OUTPUT);
16   pinMode(F, OUTPUT);
17   pinMode(G, OUTPUT);
18   pinMode(buzzer, OUTPUT);
19 }
20 void loop()
21 {
22   //chamando as funções de 5 ao 0
23   cinco();
24   delay(500);
25   quatro();
26   delay(500);
27   tres();
28   delay(500);
29   dois();
30   delay(500);
31   um();
32   delay(500);
33   zero();
34   delay(500);
35   //chamando a função para simular um alarme com o buzzer
36   alarme();
37   delay(500);
38
39   while(true)
40   {
41     //uma das opções para que o display e o buzzer permanecem desligados
42     desliga();
43     noTone(buzzer);
44   }
}

```

```

45 void um()
46 {
47   digitalWrite(A, LOW);
48   digitalWrite(B, HIGH);
49   digitalWrite(C, HIGH);
50   digitalWrite(D, LOW);
51   digitalWrite(E, LOW);
52   digitalWrite(F, LOW);
53   digitalWrite(G, LOW);
54 }
55 void dois()
56 {
57   digitalWrite(A, HIGH);
58   digitalWrite(B, HIGH);
59   digitalWrite(C, LOW);
60   digitalWrite(D, HIGH);
61   digitalWrite(E, HIGH);
62   digitalWrite(F, LOW);
63   digitalWrite(G, HIGH);
64 }
65 void tres()
66 {
67   digitalWrite(A, HIGH);
68   digitalWrite(B, HIGH);
69   digitalWrite(C, HIGH);
70   digitalWrite(D, HIGH);
71   digitalWrite(E, LOW);
72   digitalWrite(F, LOW);
73   digitalWrite(G, LOW);
74 }
75 void quatro()
76 {
77   digitalWrite(A, LOW);
78   digitalWrite(B, HIGH);
79   digitalWrite(C, HIGH);
80   digitalWrite(D, LOW);
81   digitalWrite(E, LOW);
82   digitalWrite(F, HIGH);
83   digitalWrite(G, HIGH);
84 }
85 void cinco()
86 {
87   digitalWrite(A, HIGH);
88   digitalWrite(B, LOW);
89   digitalWrite(C, HIGH);
90   digitalWrite(D, HIGH);
91   digitalWrite(E, LOW);
92   digitalWrite(F, HIGH);
93   digitalWrite(G, HIGH);
94 }
95 void zero()
96 {
97   digitalWrite(A, HIGH);
98   digitalWrite(B, HIGH);
99   digitalWrite(C, HIGH);
100  digitalWrite(D, HIGH);
101  digitalWrite(E, HIGH);
102  digitalWrite(F, HIGH);
103  digitalWrite(G, HIGH);
104 }
105 void desliga()
106 {
107   digitalWrite(A, LOW);
108   digitalWrite(B, LOW);
109   digitalWrite(C, LOW);
110   digitalWrite(D, LOW);
111   digitalWrite(E, LOW);
112   digitalWrite(F, LOW);
113   digitalWrite(G, LOW);
114 }
115 void alarme()
116 {
117   int cont = 0;
118   // função para que seja emitido 5 vezes o som pelo buzzer, simulando um alarme
119   while(cont < 5)
120   {
121     // a frequência escolhida foi 494, mas é possível utilizar qualquer outra frequência
122     tone(buzzer, 494, 100);
123     delay(200);
124     //incrementando o contador
125     cont++;
126   }
127 }

```

Laboratório 03

Contagem regressiva em display de 7 segmentos com acionamento por botão

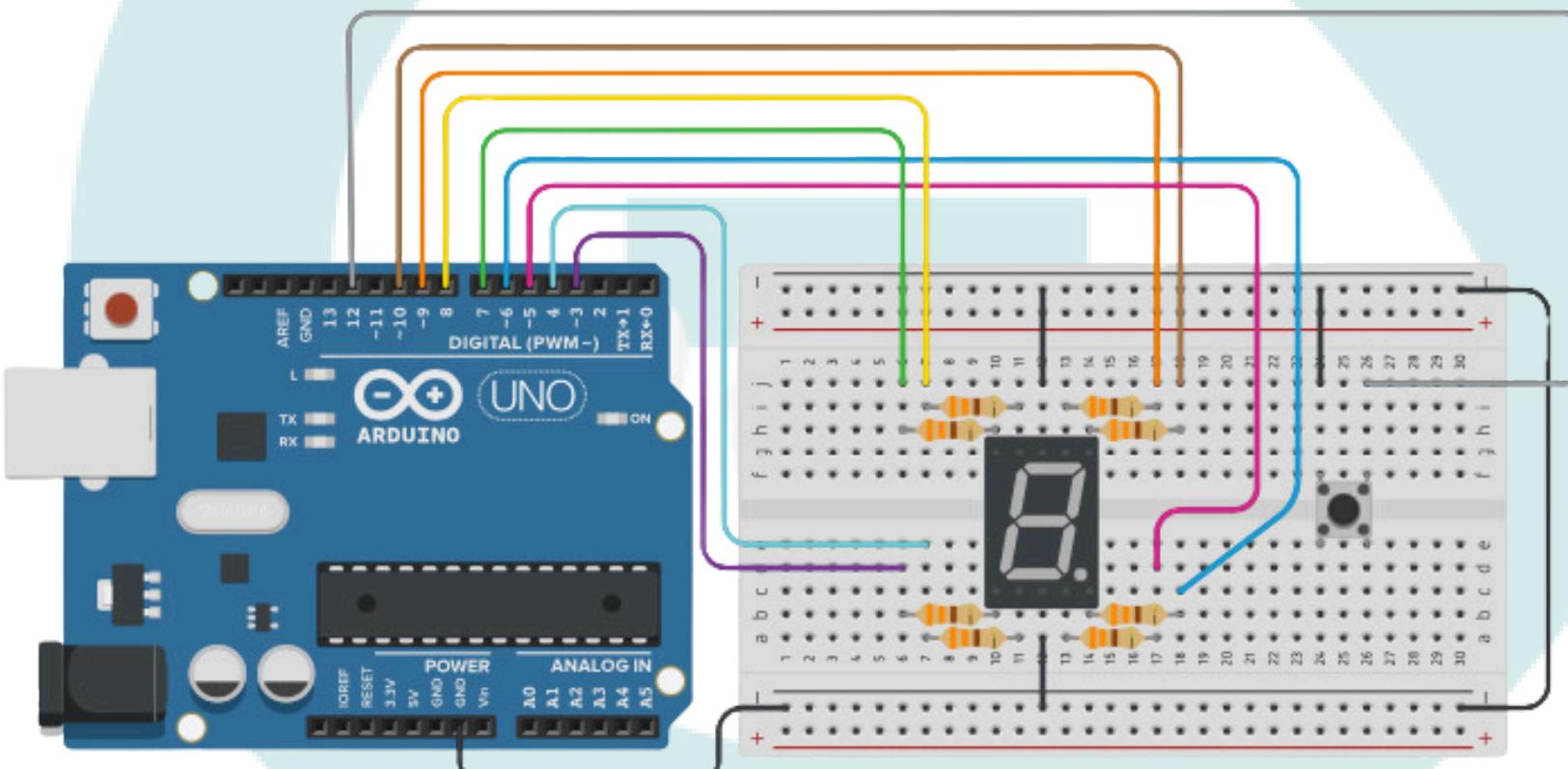
Neste guia, você aprenderá a montar um circuito que faz contagem regressiva com display de 7 segmentos e emite um alerta sonoro.

Materiais Necessários:

- Arduino UNO, Protoboard e Jumpers.
- 8 Resistores de 220 ohms.
- 1 Display de 7 Segmentos.
- 1 Botão de pressão.

ARDUTINO

Solução Laboratório 03



```

66 void zero()
67 {
68     digitalWrite(A, HIGH);
69     digitalWrite(B, HIGH);
70     digitalWrite(C, HIGH);
71     digitalWrite(D, HIGH);
72     digitalWrite(E, HIGH);
73     digitalWrite(F, HIGH);
74     digitalWrite(G, LOW);
75 }
76 void desliga()
77 {
78     digitalWrite(A, LOW);
79     digitalWrite(B, LOW);
80     digitalWrite(C, LOW);
81     digitalWrite(D, LOW);
82     digitalWrite(E, LOW);
83     digitalWrite(F, LOW);
84     digitalWrite(G, LOW);
85 }

```

```

1 int A = 9;
2 int B = 10;
3 int C = 5;
4 int D = 4;
5 int E = 3;
6 int F = 8;
7 int G = 7;
8 int botao = 12;
9 void setup()
10 {
11     pinMode(A, OUTPUT);
12     pinMode(B, OUTPUT);
13     pinMode(C, OUTPUT);
14     pinMode(D, OUTPUT);
15     pinMode(E, OUTPUT);
16     pinMode(F, OUTPUT);
17     pinMode(G, OUTPUT);
18     pinMode(botao, INPUT_PULLUP);
19 }
20 void loop()
21 {
22     // se o botao for pressionado inicia contagem regressiva
23     if(digitalRead(botao) == LOW){
24         tres();
25         delay(500);
26         dois();
27         delay(500);
28         um();
29         delay(500);
30         zero();
31         delay(500);
32         desliga();
33     }
34 }

```

```

36 void um()
37 {
38     digitalWrite(A, LOW);
39     digitalWrite(B, HIGH);
40     digitalWrite(C, HIGH);
41     digitalWrite(D, LOW);
42     digitalWrite(E, LOW);
43     digitalWrite(F, LOW);
44     digitalWrite(G, LOW);
45 }
46 void dois()
47 {
48     digitalWrite(A, HIGH);
49     digitalWrite(B, HIGH);
50     digitalWrite(C, LOW);
51     digitalWrite(D, HIGH);
52     digitalWrite(E, HIGH);
53     digitalWrite(F, LOW);
54     digitalWrite(G, HIGH);
55 }
56 void tres()
57 {
58     digitalWrite(A, HIGH);
59     digitalWrite(B, HIGH);
60     digitalWrite(C, HIGH);
61     digitalWrite(D, HIGH);
62     digitalWrite(E, LOW);
63     digitalWrite(F, LOW);
64     digitalWrite(G, HIGH);
65 }

```