Introdução a Robótica
Aula 4

Prof: Uemerson Pinheiro Junior

Sobre a aula

- Potenciômetro
- Controla Led RGB com potenciômetro
- Módulo Buzzer Ativo KY-012
- Controla Led RGB com potenciômetro
- Projeto Dó Ré Mi
- Exercício

Potenciômetro

Trata-se de um dispositivo eletrônico que permite controlar a resistência elétrica de forma variável.

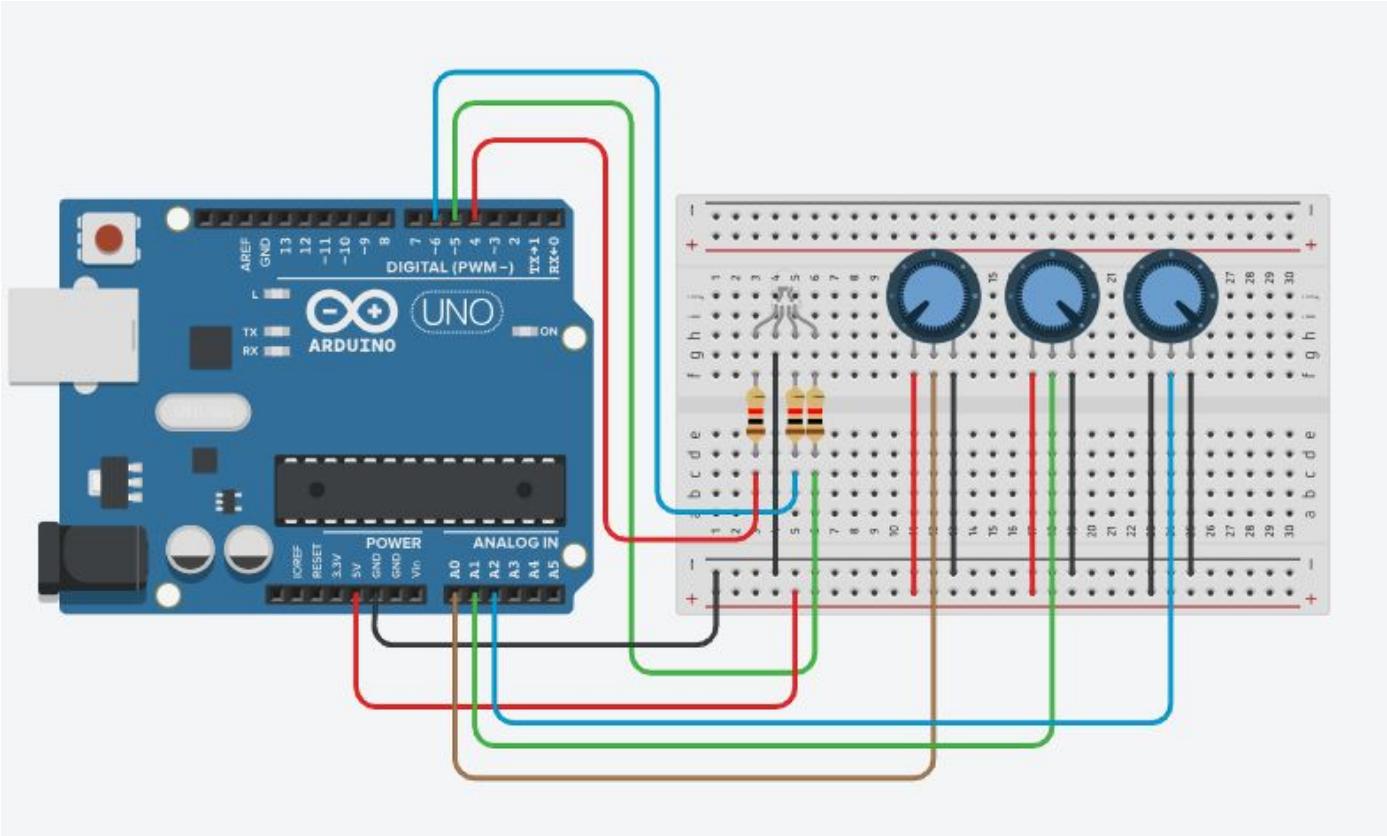
Geralmente, possui três pontos de conexão e um eixo que pode ser girado até 270 graus para ajustar a quantidade de resistência elétrica que ele oferece.

Quando os três terminais são utilizados, ele atua como um divisor de tensão, com os valores dos resistores em constante alteração conforme o movimento do eixo giratório, variando de zero (ou resistência mínima) até o máximo.

Vamos criar o projeto controla led RGB com potenciômetro no tinkercad

https://www.tinkercad.com

Elabore o esquema do circuito a seguir:



Adicione as instruções a seguir

```
int PotRedPin   = A0;          // Pino do potenciômetro vermelho
int PotGreenPin = A1;          // Pino do potenciômetro verde
int PotBluePin  = A2;          // Pino do potenciômetro azul

int LedRedPin   = 4;           // Pino do Led vermelho
int LedGreenPin = 5;           // Pino do Led verde
int LedBluePin  = 6;           // Pino do Led azul

int Value = 0;
```

Adicione as instruções a seguir

```
void setup() {
    Serial.begin(9600);
    pinMode(PotRedPin, INPUT);
    pinMode(PotGreenPin, INPUT);
    pinMode(PotBluePin, INPUT);
    pinMode(LedRedPin, OUTPUT);
    pinMode(LedGreenPin, OUTPUT);
    pinMode(LedBluePin, OUTPUT);
}
```

Adicione as instruções a seguir

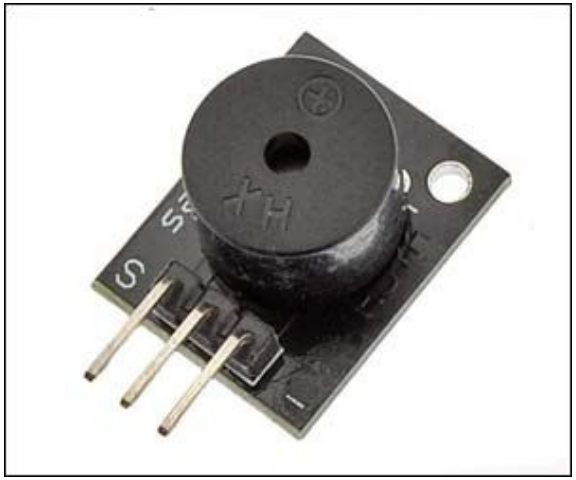
```
void loop() {
    // ---- Vermelho ---- //
    Value = analogRead(PotRedPin);
    analogWrite(LedRedPin, map(Value, 0, 1023, 0, 255));

    // ---- Verde ---- //
    Value = analogRead(PotGreenPin);
    analogWrite(LedGreenPin, map(Value, 0, 1023, 0, 255));

    // ---- Azul ---- //
    Value = analogRead(PotBluePin);
    analogWrite(LedBluePin, map(Value, 0, 1023, 0, 255));

    delay(100);
}
```

Módulo Buzzer Ativo KY-012

O módulo é um dispositivo que produz sons quando uma corrente elétrica é aplicada.

Ele contém um pequeno alto-falante projetado para emitir sinais sonoros quando alimentado com corrente contínua.

Encontra-se amplamente empregado em sistemas de alarme, impressoras, computadores e em projetos de robótica e automação residencial. Sua função principal é alertar o operador sobre eventos ou situações específicas por meio de sinais sonoros.



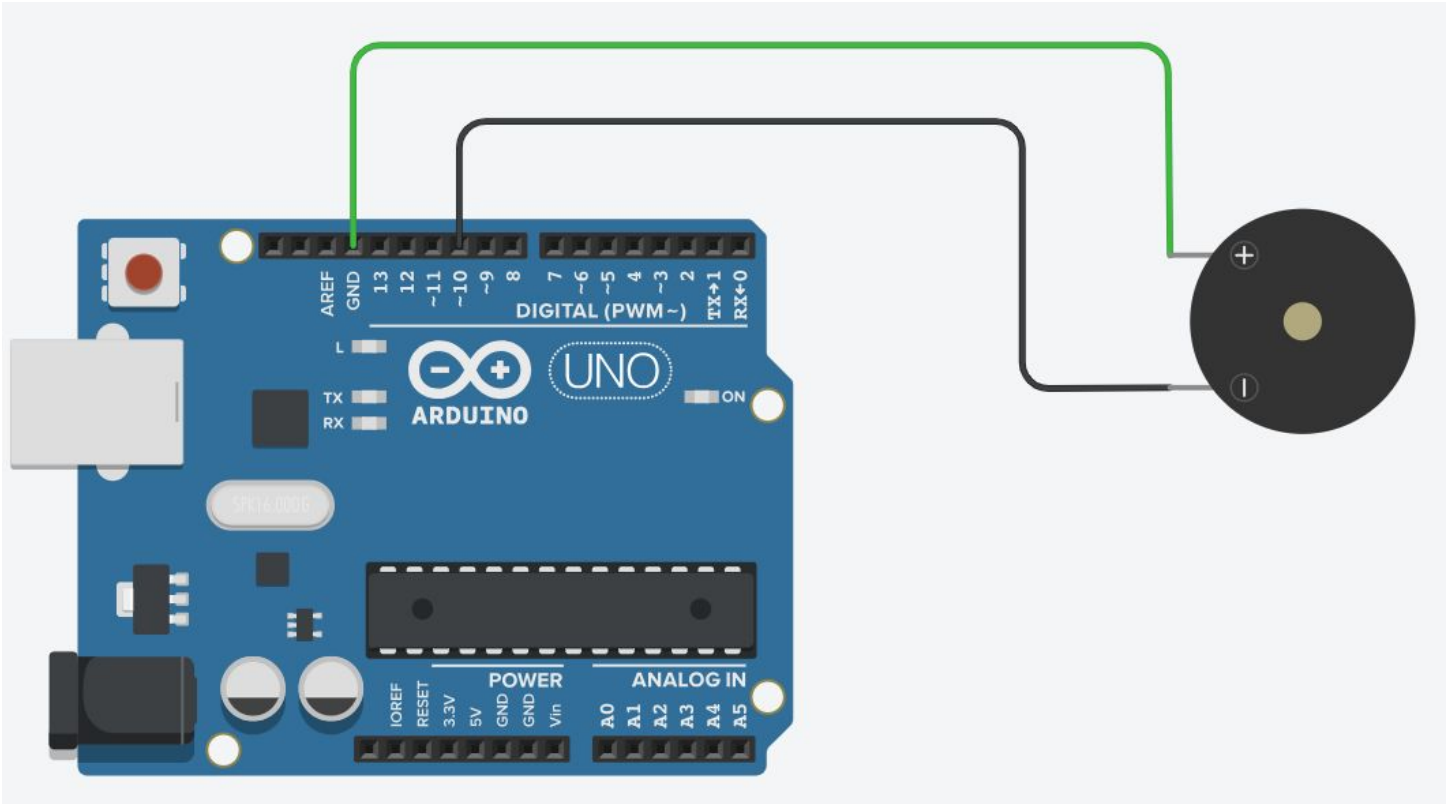
Características

- Tensão de trabalho: 3,5 - 5V
- Corrente: < 42mA
- Saída de Som: > 85DB
- Temperatura de Operação: -20°C ~ +45°C
- Temperatura de armazenamento: -20°C ~ +60°C

Vamos criar o projeto projeto Dó Ré Mi com módulo buzzer ativo KY-012 no tinkercad

https://www.tinkercad.com

Elabore o esquema do circuito a seguir:



Adicione as instruções a seguir

```
int buzzerPin = 10;  // Conectar um buzzer ao pino 10

void setup()  {
    pinMode(buzzerPin, OUTPUT); // Configurar o pino do buzzer como saída
}
```

Adicione as instruções a seguir

```
void loop()  {

    // Função tone: tone(pino, frequência, duração)
    // pino: pino conectado ao Arduino
    // frequência: definida em hertz
    // duração: definida em milissegundos

    delay(2000);
    tone(buzzerPin, 262, 200); // DO
    delay(200);
    tone(buzzerPin, 294, 300); // RE
    delay(200);
    tone(buzzerPin, 330, 300); // MI
    delay(200);
    tone(buzzerPin, 349, 300); // FA
    delay(300);
```

Adicione as instruções a seguir

```
tone(buzzerPin, 349, 300); // FA
delay(300);
tone(buzzerPin, 349, 300); // FA
delay(300);
tone(buzzerPin, 262, 100); // DO
delay(200);
tone(buzzerPin, 294, 300); // RE
delay(200);
tone(buzzerPin, 262, 100); // DO
delay(200);
tone(buzzerPin, 294, 300); // RE
delay(300);
tone(buzzerPin, 294, 300); // RE
delay(300);
tone(buzzerPin, 294, 300); // RE
delay(300);
```

Adicione as instruções a seguir

```
tone(buzzerPin, 262, 200); // DO
delay(200);
tone(buzzerPin, 392, 200); // SOL
delay(200);
tone(buzzerPin, 349, 200); // FA
delay(200);
tone(buzzerPin, 330, 300); // MI
delay(300);
tone(buzzerPin, 330, 300); // MI
delay(300);
tone(buzzerPin, 330, 300); // MI
delay(300);
tone(buzzerPin, 262, 200); // DO
delay(200);
tone(buzzerPin, 294, 300); // RE
delay(200);
```

Adicione as instruções a seguir

```
    tone(buzzerPin, 330, 300); // MI
    delay(200);
    tone(buzzerPin, 349, 300); // FA
    delay(300);
    tone(buzzerPin, 349, 300); // FA
    delay(300);
}
```

Exercício:

1. Utilizando o esquema anterior e os códigos a seguir para reproduzir a música de Beethoven - Für Elise

```
// Melodia de "Für Elise" de Beethoven

// Definição das frequências das notas
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
```

```
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
```

```
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
```



```
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
```



```
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
```

```
// Melodia de "Für Elise" de Beethoven
int melody[] = {
  NOTE_E5, NOTE_DS5, NOTE_E5, NOTE_DS5, NOTE_E5, NOTE_B4, NOTE_D5,
NOTE_C5, NOTE_A4, NOTE_C4, NOTE_E4, NOTE_A4, NOTE_B4,
  NOTE_E4, NOTE_GS4, NOTE_B4, NOTE_C5, NOTE_E4, NOTE_E5, NOTE_DS5,
NOTE_E5, NOTE_DS5, NOTE_E5, NOTE_B4, NOTE_D5, NOTE_C5,
  NOTE_A4, NOTE_C4, NOTE_E4, NOTE_A4, NOTE_B4, NOTE_E4, NOTE_C5,
NOTE_B4, NOTE_A4, NOTE_E4, NOTE_E5, NOTE_DS5, NOTE_E5,
  NOTE_DS5, NOTE_E5, NOTE_B4, NOTE_D5, NOTE_C5, NOTE_A4, NOTE_C4,
NOTE_E4, NOTE_A4, NOTE_B4, NOTE_E4, NOTE_GS4, NOTE_B4,
  NOTE_C5, NOTE_E4
};

// Duração das notas na melodia
int noteDurations[] = {
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4,
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4,
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4,
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4,
  8
};
```

```
void setup() {
  // Configura o pino do buzzer como saída
  pinMode(10, OUTPUT);
}

void loop() {
  // Toca a melodia
  for (int i = 0; i < sizeof(melody) / sizeof(melody[0]); i++) {
    int noteDuration = 1000 / noteDurations[i];
    tone(10, melody[i], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(10);
  }
  // Espera um pouco antes de repetir a melodia
  delay(2000);
}
```

Referências

Potenciômetro. Disponível em: <http://www.um.pro.br/arduino/index.php?c=potenciometro>

Módulo Buzzer Ativo KY-012. Disponível em: <http://www.um.pro.br/arduino/index.php?c=Modulo_Buzzer>