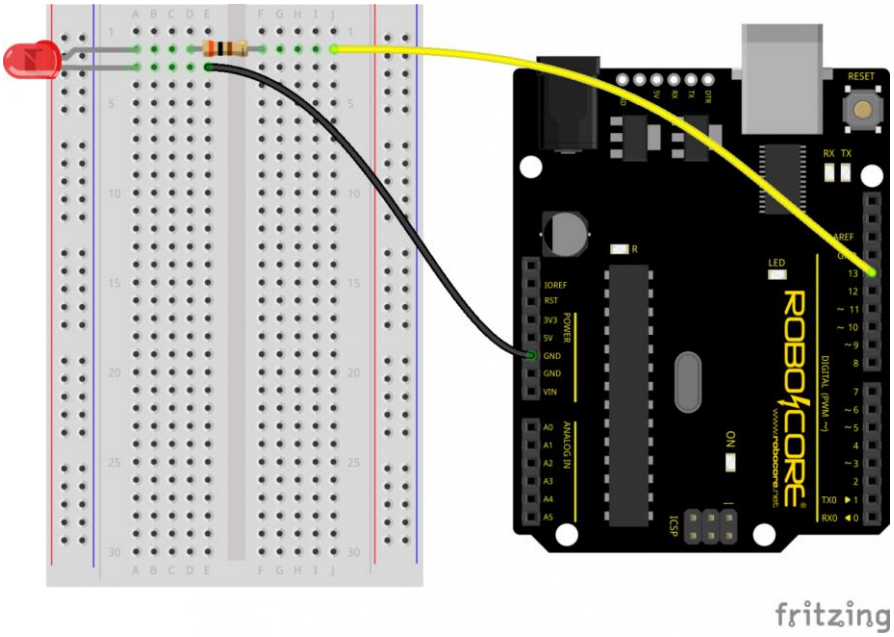
ROBOTTIME
HABILIDADES DO FUTURO

Projeto

Piscando um LED

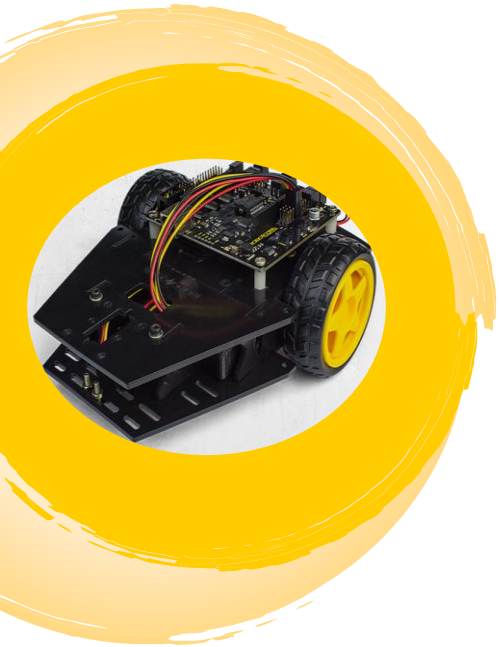
Lista de materiais

- 1x Placa Arduino + Cabo USB
- 1x LED 5mm
- 1x Resistor 300Ω
- 1x Protoboard
- Jumpers



Código

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);  // initialize digital pin LED_BUILTIN as an output.
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```



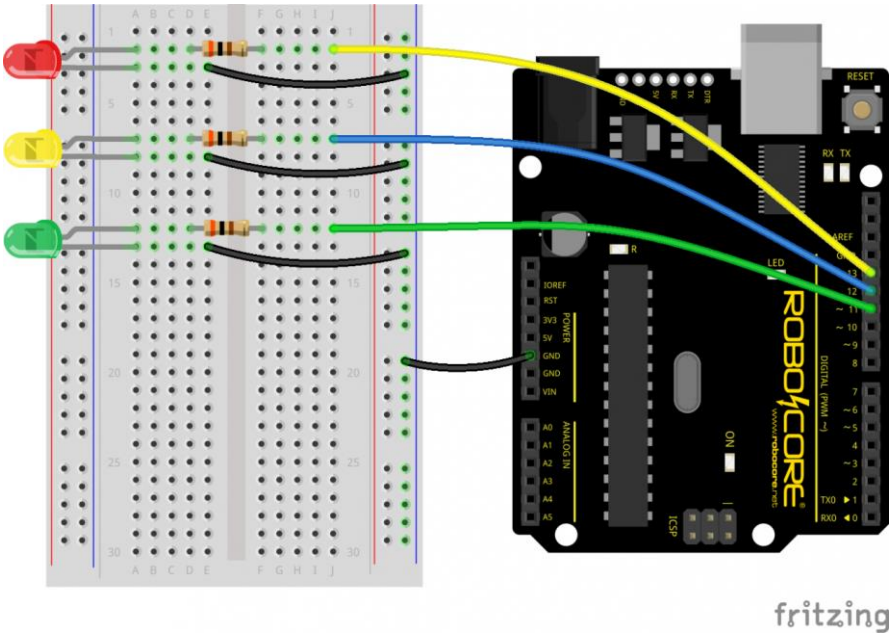
ROBOTTIME
HABILIDADES DO FUTURO

Projeto

Semáforo

Lista de materiais

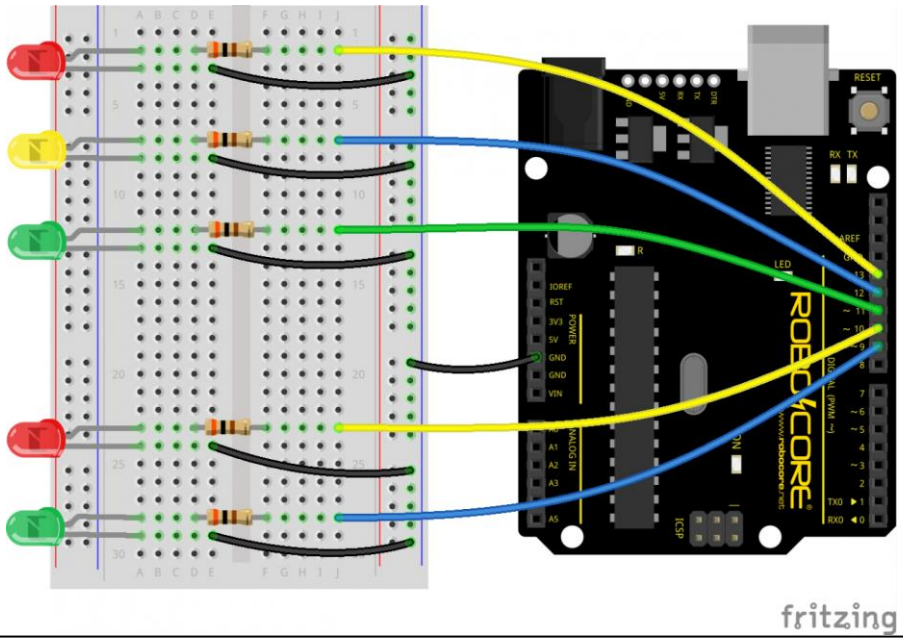
- 1x Placa Arduino com Cabo USB
- 2x LED verde 5mm
- 1x LED amarelo 5mm
- 2x LED vermelho 5mm
- 5x Resistor 300Ω
- 1x Protoboard
- Jumpers




```
void setup(){
  // Configura os pinos dos LEDs como saída
  pinMode(11, OUTPUT); // LED verde
  pinMode(12, OUTPUT); // LED amarelo
  pinMode(13, OUTPUT); // LED vermelho
}

void loop(){
  // Sinal aberto: apaga LED vermelho, acende LED verde
  digitalWrite(11, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
  delay(3000);
```

```
  // Sinal fechado: apaga LED verde, acende LED amarelo
  digitalWrite(11, LOW);
  digitalWrite(12, HIGH);
  digitalWrite(13, LOW);
  delay(2000);

  // Sinal fechado: apaga LED amarelo, acende LED vermelho
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
  digitalWrite(13, HIGH);
  delay(5000);
}
```

Circuito 2




```
void setup(){
  // Configura os pinos com os LEDs como saída
  pinMode(9, OUTPUT); // LED verde pedestres
  pinMode(10, OUTPUT); // LED vermelho pedestres
  pinMode(11, OUTPUT); // LED verde carros
  pinMode(12, OUTPUT); // LED amarelo carros
  pinMode(13, OUTPUT); // LED vermelho carros
}

void loop(){
  // Sinal para pedestres fechado: apaga LED verde, acende LED vermelho
  digitalWrite(9, LOW);
  digitalWrite(10, HIGH);
  // Sinal para carros aberto: apaga LED vermelho, acende LED verde
```

```
  digitalWrite(11, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
  delay(3000);

  // Sinal para carros fechando: apaga LED verde, acende LED amarelo
  digitalWrite(11, LOW);
  digitalWrite(12, HIGH);
  digitalWrite(13, LOW);
  delay(2000);

  // Sinal para pedestres aberto: apaga LED vermelho, acende LED verde
  digitalWrite(9, HIGH);
  digitalWrite(10, LOW);
  // Sinal para carros aberto: apaga LED verde, acende LED vermelho
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
  digitalWrite(13, HIGH);
  delay(5000);
}
```

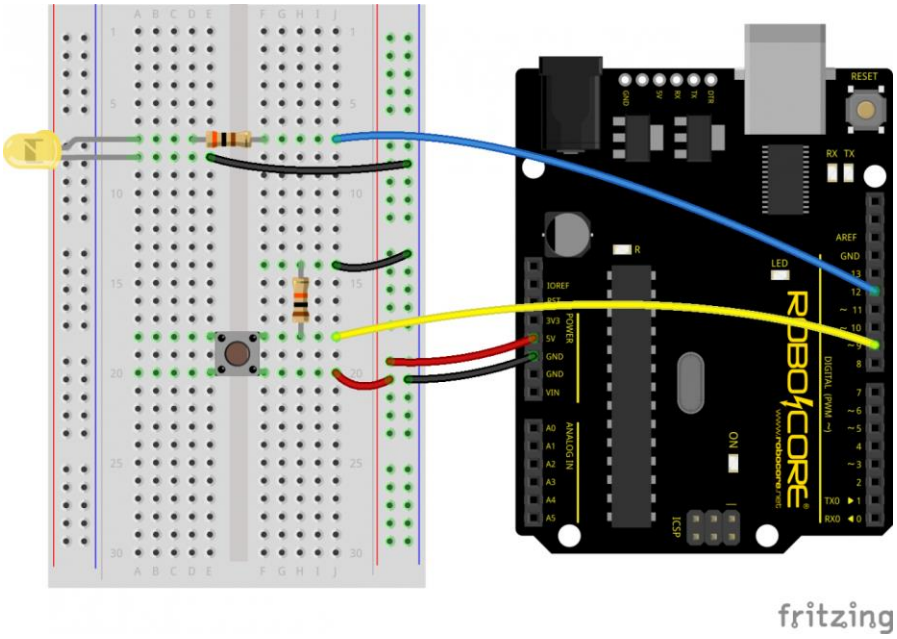
ROBOTTIME
HABILIDADES DO FUTURO

Projeto

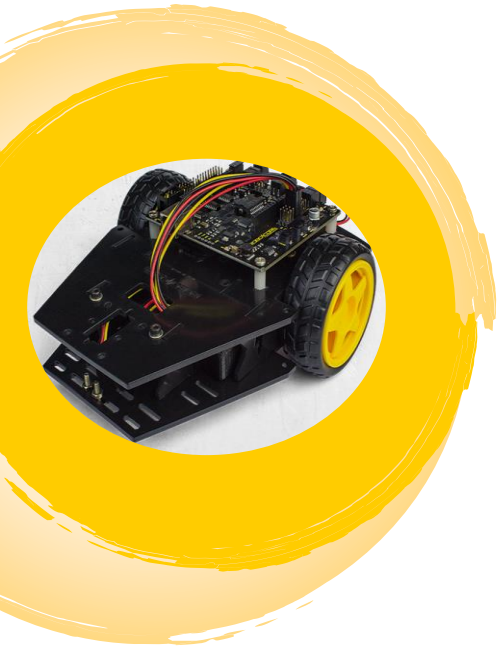
Lendo um botão

Lista de materiais

- 1x Placa Arduino com Cabo USB
- 1x LED 5mm
- 1x Resistor 300Ω
- 1x Chave Momentânea (PushButton)
- 1x Resistor 10kΩ
- 1x Protoboard
- Jumpers




```
void setup(){
  pinMode(9, INPUT); // configura o pino com o botão como entrada
  pinMode(12, OUTPUT); // configura o pino com o LED como saída
}

void loop(){
  if (digitalRead(9) == HIGH){ // se botão estiver pressionado (HIGH)
    digitalWrite(12, HIGH); // acende o LED
  }
  else{ // se não estiver pressionado (LOW)
    digitalWrite(12, LOW); // apaga o LED
  }
}
```



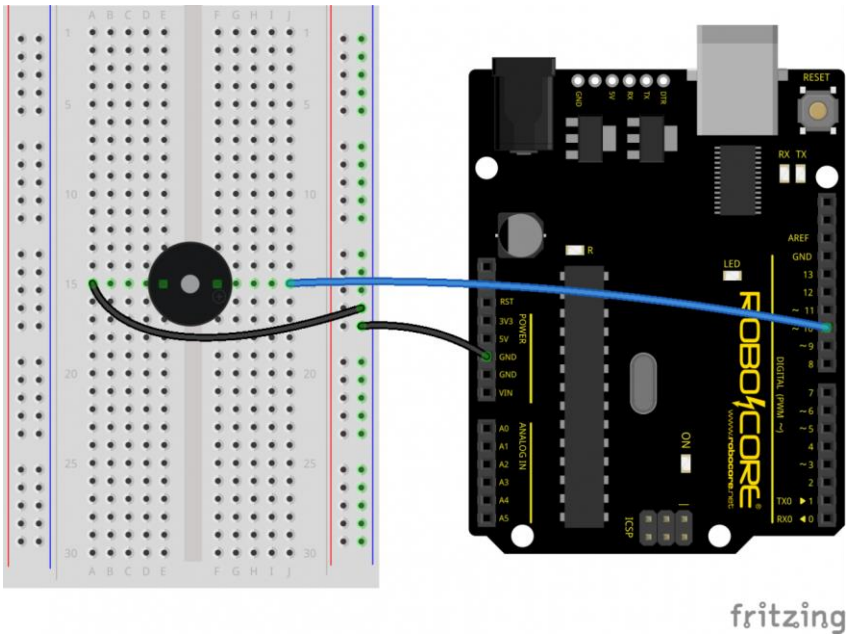
ROBOTTIME
HABILIDADES DO FUTURO

Projeto

Fazendo barulho

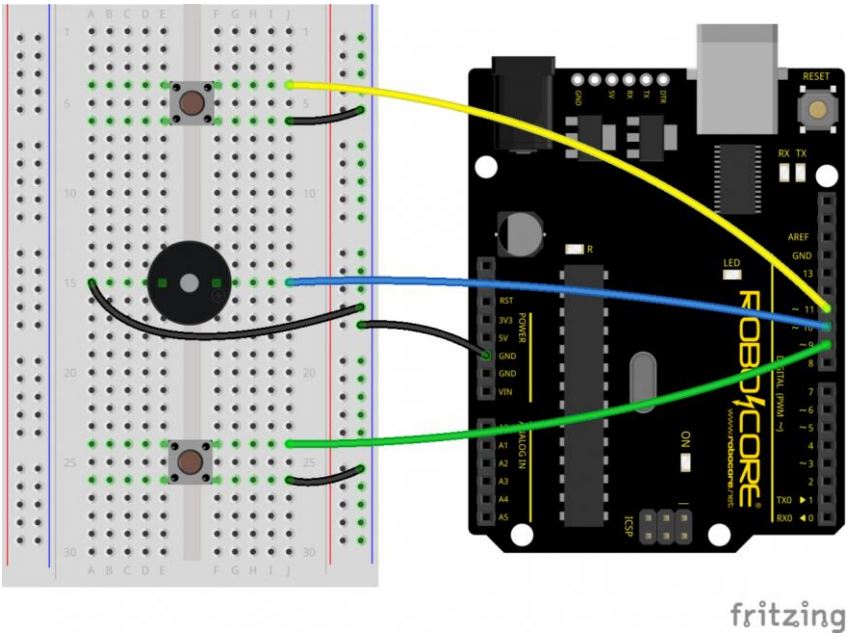
Lista de materiais

- 1x Placa Arduino + Cabo USB
- 1x Buzzer Passivo 5V
- 2x Chave Momentânea (PushButton)
- 1x Protoboard
- Jumpers




```
int frequencia = 2000;

void setup(){
  pinMode(10, OUTPUT); // configura o pino com o buzzer como saída
}

void loop(){
  tone(10, frequencia); // gera frequencia de 2000Hz no buzzer
  delay(1000);
  noTone(10); // para frequencia no buzzer
  delay(1000);
}
```

Circuito2



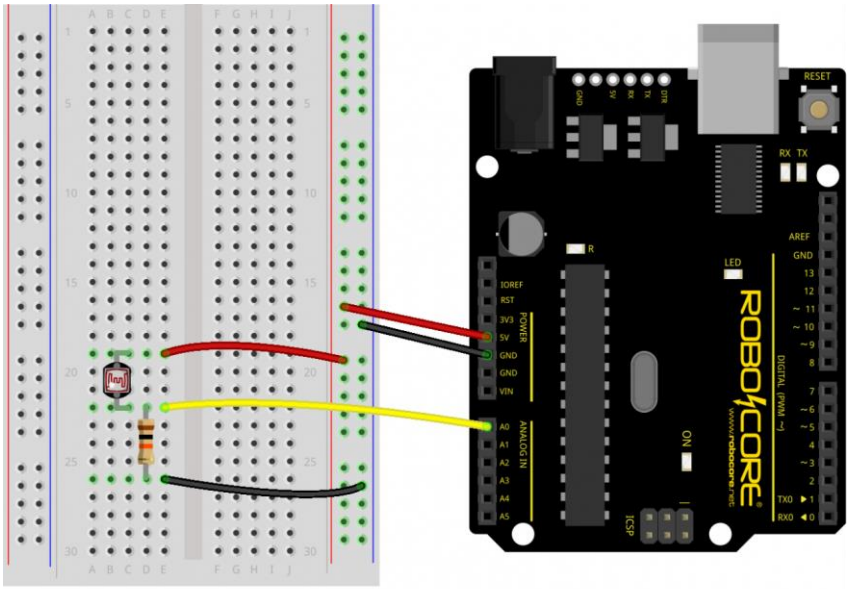

```
int frequencia = 2000; // frequencia inicial do buzzer
const int pinoBuzzer = 10; // pino onde o buzzer esta conectado
const int pinoBotao1 = 9; // pino onde o botao 1 esta conectado
const int pinoBotao2 = 11; // pino onde o botao 2 esta conectado

void setup(){
  pinMode(pinoBuzzer, OUTPUT); // configura o pino com o buzzer como saída
  pinMode(pinoBotao1, INPUT_PULLUP); // configura o pino com o Botão como entrada
  pinMode(pinoBotao2, INPUT_PULLUP); // configura o pino com o Botão como entrada
}

void loop(){

  // Soma 100 à frequencia atual se o botao estiver pressionado
  if (digitalRead(pinoBotao1) == LOW) {
    frequencia = frequencia + 100;
  }

  // Subtrai 100 da frequencia atual se o botao estiver pressionado
  if (digitalRead(pinoBotao2) == LOW) {
    frequencia = frequencia - 100;
  }

  tone(pinoBuzzer, frequencia); // gera frequencia no buzzer
  delay(1000);
  noTone(pinoBuzzer); // para frequencia no buzzer
  delay(1000);
}
```

Projeto

Sensor de Luz

Lista de materiais

- 1x Placa Arduino + Cabo USB
- 1x Protoboard
- 1x Sensor de Luminosidade LDR 5mm
- 1x Resistor 10kΩ
- Jumpers

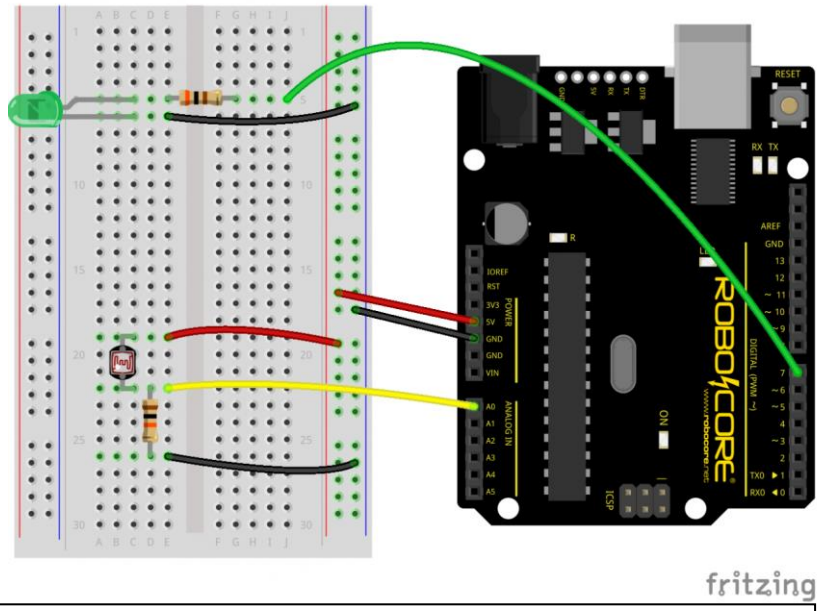



```
const int pinoLDR = A0; // pino onde o LDR está conectado
int leitura = 0; // variável para armazenar o valor lido pelo ADC
float tensao = 0.0; // variável para armazenar a tensão

void setup() {
  // Inicia e configura a Serial
  Serial.begin(9600); // 9600bps

  // configura o pino com LDR como entrada
  pinMode(pinoLDR, INPUT); // pino A0
}

void loop() {
  // le o valor de tensão no pino do LDR
```

```
  leitura = analogRead(pinoLDR);

  // imprime valor lido pelo arduino (0 a 1023)
  Serial.print("Leitura: ");
  Serial.print(leitura);
  Serial.print("\t"); // tabulacao

  // converte e imprime o valor em tensão elétrica
  tensao = leitura * 5.0 / 1023.0;
  Serial.print("Tensão: ");
  Serial.print(tensao);
  Serial.print("V");

  Serial.println(); // nova linha

  delay(1000); // aguarda 1 segundo para uma nova leitura
}
```

Lista de materiais - Circuito 2

- 1x Placa Arduino + Cabo USB
- 1x Protoboard
- 1x Sensor de Luminosidade LDR 5mm
- 1x Resistor 10kΩ
- 1x LED VERDE 5mm
- 1x Resistor 300Ω
- Jumpers




```
const int pinoLDR = A0; // pino onde o LRD está conectado
const int pinoLED = 7; // pino onde o LED está conectado
int leitura = 0; // variável para armazenar o valor lido pelo ADC

void setup() {
  // configura os pinos do LDR e LED
```

```
const int pinoLDR = A0; // pino onde o LRD está conectado
const int pinoLED = 7;
  pinMode(pinoLDR, INPUT);
  pinMode(pinoLED, OUTPUT);
}

void loop() {
  // le o valor de ADC no pino do LDR
  leitura = analogRead(pinoLDR);

  // verifica luminosidade do ambiente
  if(leitura < 40){ // se a luminosidade for menor que 40
    digitalWrite(pinoLED,HIGH); // acende o LED
  }
  else { // se não
    digitalWrite(pinoLED,LOW); // apaga o LED
  }
  delay(100); // aguarda 100 milissegundos para uma nova leitura
}
```

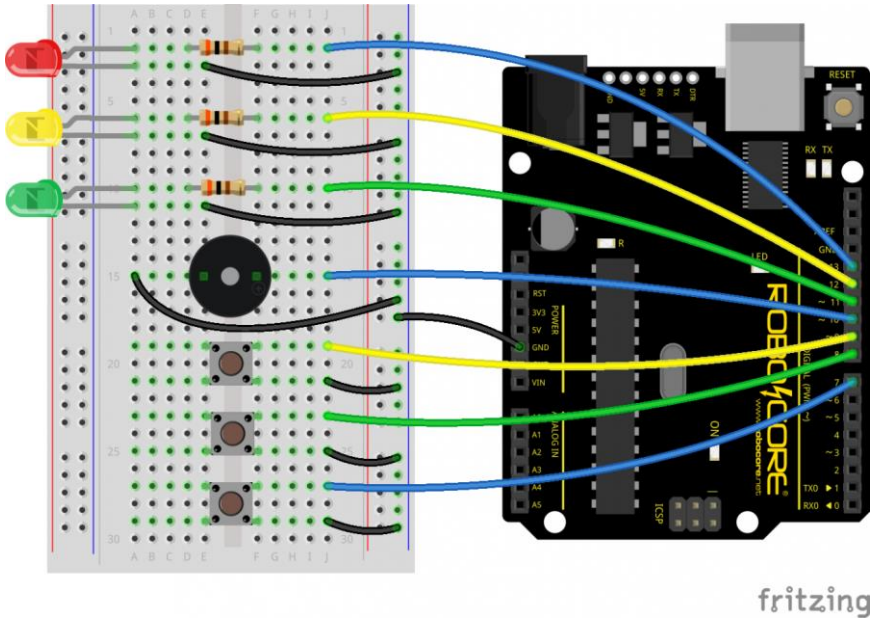
ROBOTTIME
HABILIDADES DO FUTURO

Projeto

Piano

Lista de materiais

- 1x Placa Arduino + Cabo USB
- 1x Buzzer Passivo 5V
- 1x LED verde 5mm
- 1x LED amarelo 5mm
- 1x LED vermelho 5mm
- 3x Resistor 300Ω
- 3x Chave Momentânea (PushButton)
- 1x Protoboard
- Jumpers

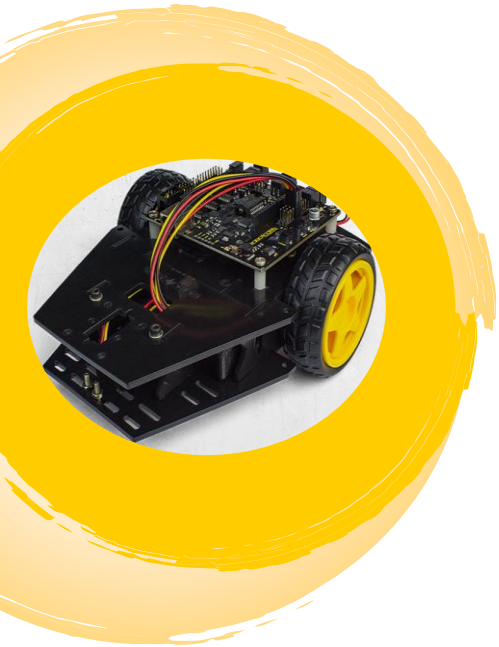


```
const int pino_buzzer = 10; // pino onde o buzzer está conectado

// Frequencias de cada nota musical
const int c = 261; // Dó
const int d = 293; // Ré
const int e = 329; // Mi
const int f = 349; // Fá
const int g = 391; // Sol
const int a = 440; // Lá
const int b = 493; // Si

void setup(){
  pinMode(pino_buzzer, OUTPUT); // configura o pino com o buzzer como saída
}

void loop(){
  // Toca a nota Dó por 1 segundo
  tone(pino_buzzer, c);
  delay(1000);

  // Toca a nota Ré por 1 segundo
  tone(pino_buzzer, d);
  delay(1000);

  // Toca a nota Mi por 1 segundo
  tone(pino_buzzer, e);
  delay(1000);

  // Toca a nota Fá por 1 segundo
  tone(pino_buzzer, f);
  delay(1000);

  // Toca a nota Sol por 1 segundo
  tone(pino_buzzer, g);
  delay(1000);

  // Toca a nota Lá por 1 segundo
  tone(pino_buzzer, a);
  delay(1000);

  // Toca a nota Si por 1 segundo
  tone(pino_buzzer, b);
  delay(1000);

  // Desliga o Buzzer por 2 segundos
  noTone(pino_buzzer);
  delay(2000);
}
```

Código 2

```
const int pino_botao1 = 7;
const int pino_botao2 = 8;
const int pino_botao3 = 9;
const int pino_buzzer = 10;
const int pino_led1 = 11;
const int pino_led2 = 12;
const int pino_led3 = 13;

// Frequencias de cada nota musical
const int c = 261; // Dó
const int d = 293; // Ré
const int e = 329; // Mi
const int f = 349; // Fá
const int g = 391; // Sol
const int a = 440; // Lá
const int b = 493; // Si

void setup(){
  // Configura os pinos
  pinMode(pino_botao1, INPUT_PULLUP);
  pinMode(pino_botao2, INPUT_PULLUP);
  pinMode(pino_botao3, INPUT_PULLUP);
  pinMode(pino_buzzer, OUTPUT);
  pinMode(pino_led1, OUTPUT);
  pinMode(pino_led2, OUTPUT);
  pinMode(pino_led3, OUTPUT);
}

void loop(){
  // Salva o estado de cada botão
  bool estado_botao1 = digitalRead(pino_botao1);
  bool estado_botao2 = digitalRead(pino_botao2);
  bool estado_botao3 = digitalRead(pino_botao3);

  if (estado_botao1 == LOW) {
    tone(pino_buzzer, c); // Dó
    digitalWrite(pino_led1, HIGH);
    digitalWrite(pino_led2, LOW);
    digitalWrite(pino_led3, LOW);
  }
  else if (estado_botao2 == LOW) {
    tone(pino_buzzer, d); // Ré
    digitalWrite(pino_led1, LOW);
    digitalWrite(pino_led2, HIGH);
    digitalWrite(pino_led3, LOW);
  }
```

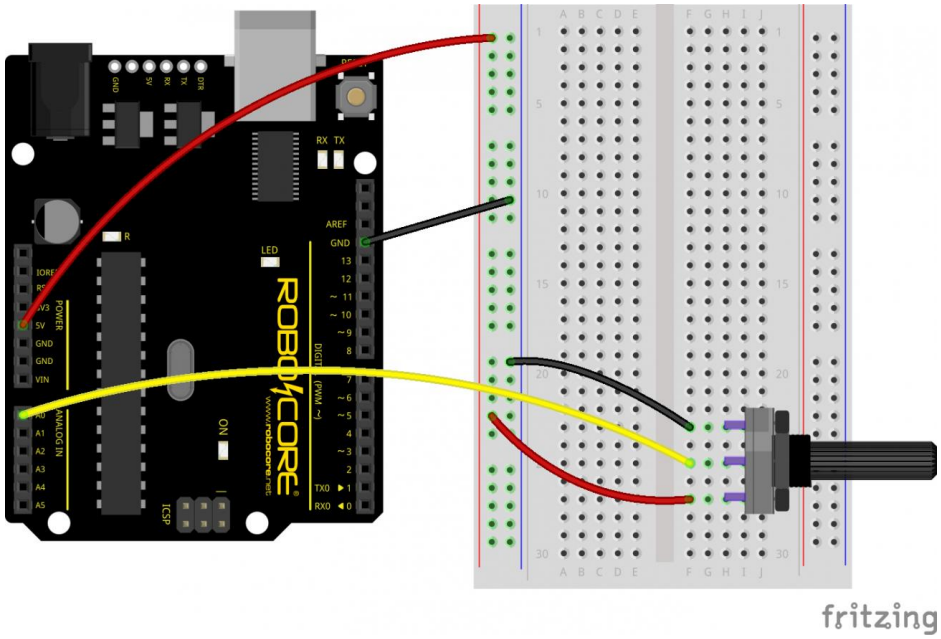
```
  else if (estado_botao3 == LOW) {
    tone(pino_buzzer, e); // Mi
    digitalWrite(pino_led1, LOW);
    digitalWrite(pino_led2, LOW);
    digitalWrite(pino_led3, HIGH);
  }
  else {
    noTone(pino_buzzer); // desliga buzzer
    digitalWrite(pino_led1, LOW);
    digitalWrite(pino_led2, LOW);
    digitalWrite(pino_led3, LOW);
  }
}
```



Projeto

Trabalhando com gráficos

Lista de materiais



- 1x Placa Arduino com Cabo USB
- 1x Potenciômetro 10kΩ
- 1x Protoboard
- Jumpers

```
const int pinoPotenciometro = A0; // pino onde o potenciometro está conectado
int leitura = 0; // variável para armazenar o valor lido pelo ADC
float tensao = 0.0; // variável para armazenar a tensão

void setup() {
  // Inicia e configura a Serial
  Serial.begin(9600); // 9600 bps

  // configura o pino com potenciometro como entrada
  pinMode(pinoPotenciometro, INPUT); // pino A0
}

void loop() {
  // le o valor de tensão no pino do potenciometro
  leitura = analogRead(pinoPotenciometro);
```

```
  // converte e imprime o valor em tensão elétrica
  tensao = leitura * 5.0 / 1023.0;

  // imprime valor no plotter serial
  Serial.println(tensao);

  delay(100); // aguarda 100 milissegundos para uma nova leitura
}
```

Código 2

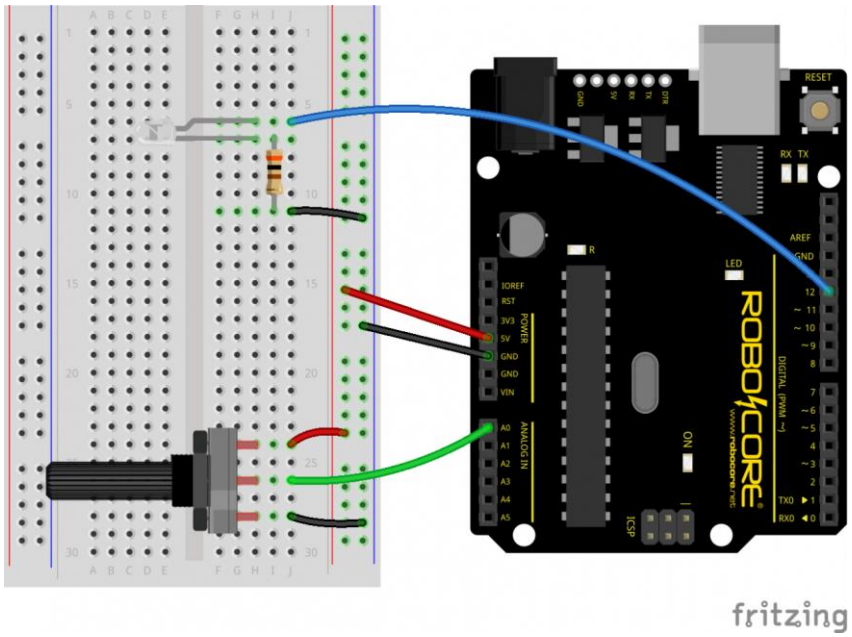
```
const int pinoPotenciometro = A0; // pino onde o potenciometro está conectado
int leitura = 0; // variável para armazenar o valor lido pelo ADC
float tensao = 0.0; // variável para armazenar a tensão

void setup() {
  // Inicia e configura a Serial
  Serial.begin(9600); // 9600 bps

  // configura o pino com potenciometro como entrada
  pinMode(pinoPotenciometro, INPUT); // pino A0
}

void loop() {
  // le o valor de tensão no pino do potenciometro
  leitura = analogRead(pinoPotenciometro);

  // converte e imprime o valor em tensão elétrica
  tensao = leitura * 5.0 / 1023.0;

  // imprime valor no plotter serial
  Serial.print(leitura);
  Serial.print(" "); // separa as váriaveis
  Serial.print(tensao*100); //exibe o valor de tensão vezes 100.
  Serial.print(" "); // separa as váriaveis
  Serial.print(tensao*90); //exibe o valor de tensão vezes 90.
  Serial.print(" "); // separa as váriaveis
  Serial.print(leitura/2); //exibe o valor de tensão dividido por 2.
  Serial.print(" "); // separa as váriaveis
  Serial.println(tensao*80); //exibe o valor de tensão vezes 80.

  delay(100); // aguarda 100 milissegundos para uma nova leitura
}
```

Projeto

Dimmer LED

Lista de materiais

- 1x Placa Arduino + Cabo USB
- 1x Potenciômetro 10kΩ
- 1x LED Branco de Alto Brilho 5mm
- 1x Resistor 300Ω
- 1x Protoboard
- Jumpers

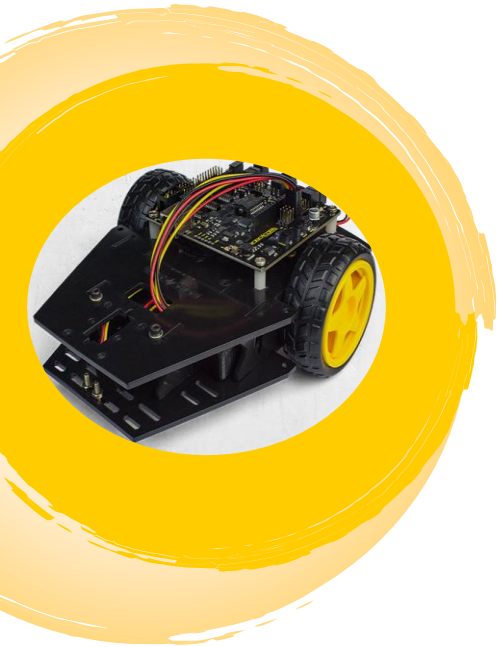



```
const int pinoPotenciometro = A0; // pino onde o pontenciômetro está conectado
const int pinoLED = 12; // pino onde o LED está conectado
const int periodo = 1023;
int tempo_ligado, tempo_desligado ; // variável para armazenar o valor lido pelo ADC

void setup() {
  pinMode(pinoPotenciometro, INPUT); // configura o pino com potenciômetro como entrada
  pinMode(pinoLED, OUTPUT); // configura o pino com o LED como saída
  Serial.begin(9600);
}

void loop() {
  tempo_ligado = analogRead(pinoPotenciometro);// le o valor de tensão no pino do potenciômetro
  tempo_desligado = periodo - tempo_ligado; // atribui a variável tempo_desligado, quanto tempo o LED
deverá permanecer desligado.
```

```
  digitalWrite(pinoLED, HIGH); // aciona o LED
  delay(tempo_ligado); // aguarda o valor lido no pino do potenciômetro em milissegundos
  digitalWrite(pinoLED, LOW); // apaga o LED
  delay(tempo_desligado); // aguarda o valor lido no pino do potenciômetro em milissegundos
  Serial.print(tempo_ligado);
  Serial.print("\t");
  Serial.println(tempo_desligado);

}
```

Código 2

```
const int pinoPotenciometro = A0; // pino onde o pontenciômetro está conectado
const int pinoLED = 11; // pino onde o LED está conectado
int leitura = 0; // variável para armazenar o valor lido pelo ADC
int pwm = 0; // variável para armazenar o valor da razão cíclica (duty cycle)

void setup() {
  pinMode(pinoPotenciometro, INPUT); // configura o pino com potenciômetro como entrada
  pinMode(pinoLED, OUTPUT); // configura o pino com o LED como saída
}

void loop() {
  // le o valor de tensão no pino do potenciômetro
  leitura = analogRead(pinoPotenciometro);

  // converte o valor do potenciômetro, de 0 a 1023, em um valor de 0 a 255
  pwm = map(leitura, 0, 1023, 0, 255);

  // atualiza a razão cíclica do pino 11, variando a intensidade do LED
  analogWrite(pinoLED, pwm);

}
```

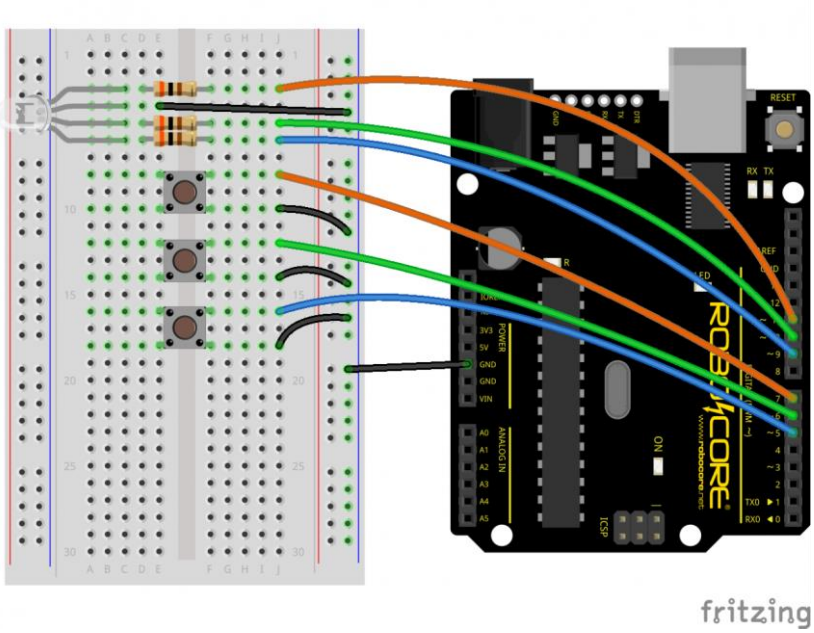


Projeto

RGB (RED GREEN BLUE)

Lista de materiais

- 1x Placa Arduino com Cabo USB
- 1x LED RGB
- 3x Resistor 300Ω
- 3x Chave Momentânea (PushButton)
- 1x Protoboard
- Jumpers




```
// Declaração dos pinos
const int pinoLEDR = 11;
const int pinoLEDG = 10;
const int pinoLEDB = 9;
const int pinoBotaoR = 7;
const int pinoBotaoG = 6;
const int pinoBotaoB = 5;

// Variáveis para armazenar o estado de cada botão
int leituraBotaoR;
int leituraBotaoG;
int leituraBotaoB;

void setup() {
  // Configura os pinos
  pinMode(pinoLEDR,OUTPUT);
```

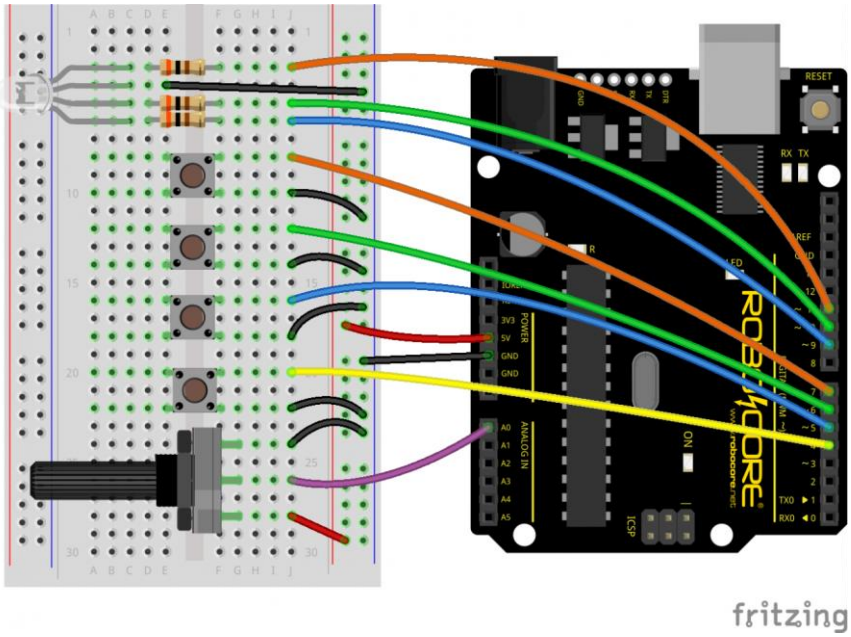
```
  pinMode(pinoLEDG,OUTPUT);
  pinMode(pinoLEDB,OUTPUT);
  pinMode(pinoBotaoR,INPUT_PULLUP);
  pinMode(pinoBotaoG,INPUT_PULLUP);
  pinMode(pinoBotaoB,INPUT_PULLUP);
}

void loop() {
  // le estado dos botões
  leituraBotaoR = digitalRead(pinoBotaoR);
  leituraBotaoG = digitalRead(pinoBotaoG);
  leituraBotaoB = digitalRead(pinoBotaoB);

  if(leituraBotaoR == LOW) { // verifica se o botão foi pressionado
    analogWrite(pinoLEDR, 255); // aciona a cor com brilho máximo
  } else { // senão
    analogWrite(pinoLEDR, 0); // apaga o LED
  }

  if(leituraBotaoG == LOW) {  // verifica se o botão foi pressionado
    analogWrite(pinoLEDG, 255); // aciona a cor com brilho máximo
  } else { // senão
    analogWrite(pinoLEDG, 0); // apaga o LED
  }

  if(leituraBotaoB == LOW) {  // verifica se o botão foi pressionado
    analogWrite(pinoLEDB, 255); // aciona a cor com brilho máximo
  } else { // senão
    analogWrite(pinoLEDB, 0); // apaga o LED
  }

  delay(100); // aguarda 100 milissegundos para uma nova leitura
}
```

Projeto

Pixel

Lista de materiais

- 1x Placa Arduino com cabo USB
- 1x LED RGB Alto Brilho
- 1x Potenciômetro 10kΩ
- 3x Resistor 300Ω
- 4x Chave Momentânea (PushButton)
- 1x Protoboard
- Jumpers




```
// Declaração dos pinos
const int pinoLEDR = 11;
const int pinoLEDG = 10;
const int pinoLEDB = 9;
const int pinoBotaoR = 7;
const int pinoBotaoG = 6;
const int pinoBotaoB = 5;
const int pinoBotaoA = 4; // botao de ajuste
const int pinoPotenciometro = A0; // potenciometro de intensidade

// Variáveis para armazenar o estado de cada botão
int leituraBotaoR;
int leituraBotaoG;
int leituraBotaoB;
int leituraBotaoA;
```

```
int leituraPotenciometro;
int pwmR = 0;
int pwmG = 0;
int pwmB = 0;

void setup() {
  // Configura os pinos
  pinMode(pinoLEDR,OUTPUT);
  pinMode(pinoLEDG,OUTPUT);
  pinMode(pinoLEDB,OUTPUT);
  pinMode(pinoBotaoR,INPUT_PULLUP);
  pinMode(pinoBotaoG,INPUT_PULLUP);
  pinMode(pinoBotaoB,INPUT_PULLUP);
  pinMode(pinoBotaoA,INPUT_PULLUP);
  pinMode(pinoPotenciometro,INPUT);
}

void loop() {
  // le estado dos botões e do potenciometro
  leituraBotaoR = digitalRead(pinoBotaoR);
  leituraBotaoG = digitalRead(pinoBotaoG);
  leituraBotaoB = digitalRead(pinoBotaoB);
  leituraBotaoA = digitalRead(pinoBotaoA);
  leituraPotenciometro = analogRead(pinoPotenciometro);

  if(leituraBotaoR == LOW) { // verifica se o botão foi pressionado
    // Se o botao de ajuste estiver pressionado
    if (leituraBotaoA == LOW) {
      // ajusta a intensidade da cor de acordo com o potenciometro
      pwmR = map(leituraPotenciometro, 0, 1023, 0, 255);
    }
    // aciona a cor com a intensidade configurada
    analogWrite(pinoLEDR, pwmR);
  }else { // senão
    analogWrite(pinoLEDR, 0); // apaga o LED
  }

  if(leituraBotaoG == LOW) {  // verifica se o botão foi pressionado
    // Se o botao de ajuste estiver pressionado
    if (leituraBotaoA == LOW) {
      // ajusta a intensidade da cor de acordo com o potenciometro
      pwmG = map(leituraPotenciometro, 0, 1023, 0, 255);
    }
    // aciona a cor com a intensidade configurada
    analogWrite(pinoLEDG, pwmG);
  }else { // senão
    analogWrite(pinoLEDG, 0); // apaga o LED
  }
```

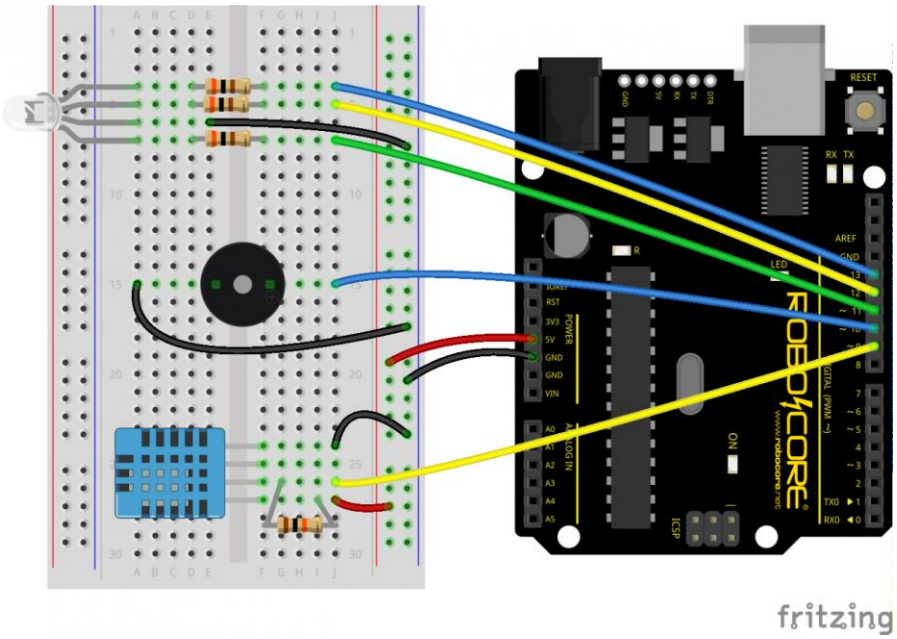
```
  if(leituraBotaoB == LOW) {  // verifica se o botão foi pressionado
    // Se o botao de ajuste estiver pressionado
    if (leituraBotaoA == LOW) {
      // ajusta a intensidade da cor de acordo com o potenciometro
      pwmB = map(leituraPotenciometro, 0, 1023, 0, 255);
    }
    // aciona a cor com a intensidade configurada
    analogWrite(pinoLEDB, pwmB);
  }else { // senão
    analogWrite(pinoLEDB, 0); // apaga o LED
  }

  delay(100); // aguarda 100 milissegundos para uma nova leitura
}
```

Projeto

Alarme

Lista de materiais

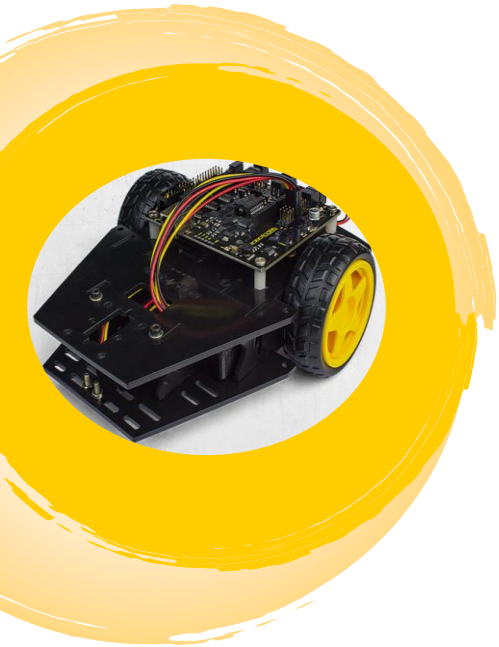
- 1x Placa Arduino + Cabo USB
- 1x Sensor de Temperatura e Umidade DHT11
- 1x Buzzer
- 1x LED RGB 5 mm
- 3x Resistor 300 Ω
- 1x Resistor 10 kΩ
- 1x Protoboard
- Jumpers




```
// Inclui a biblioteca DHT que possui as funções dos sensores do tipo DHT
#include "DHT.h"

// Define os pinos de cada componente conectado
const int pino_dht = 9; // DHT
const int pinoLEDR = 11; // LED RGB (vermelho)
const int pinoLEDG = 12; // LED RGB (verde)
const int pinoLEDB = 13; // LED RGB (azul)
const int pinoBuzzer = 10; // buzzer

// variável para armazenar o valor de temperatura e umidade
float temperatura;
float umidade;
```

```
DHT dht(pino_dht, DHT11); // define o pino e o tipo de DHT

void setup() {
  dht.begin(); // inicializa o sensor DHT

  // Configura os pinos dos LEDs e buzzer
  pinMode(pinoLEDB, OUTPUT);
  pinMode(pinoLEDG, OUTPUT);
  pinMode(pinoLEDR, OUTPUT);
  pinMode(pinoBuzzer, OUTPUT);
}

void loop() {
  // Aguarda alguns segundos entre uma leitura e outra
  delay(2000); // 2 segundos (Datasheet)

  // A leitura da temperatura ou umidade pode levar 250ms
  // O atraso do sensor pode chegar a 2 segundos
  temperatura = dht.readTemperature(); // lê a temperatura em Celsius
  umidade = dht.readHumidity(); // lê a umidade

  // Se ocorreu alguma falha durante a leitura
  if (isnan(umidade) || isnan(temperatura)) {
    // Acende branco
    digitalWrite(pinoLEDR, HIGH);
    digitalWrite(pinoLEDG, HIGH);
    digitalWrite(pinoLEDB, HIGH);
  }
  else { // Se não

    if (temperatura > 35 || temperatura < 15) {
      // Acende vermelho e aciona buzzer a 2000Hz
      digitalWrite(pinoLEDR, HIGH);
      digitalWrite(pinoLEDG, LOW);
      digitalWrite(pinoLEDB, LOW);
      tone(pinoBuzzer, 2000);
      delay(1000);
    }
    else if (temperatura > 30 || temperatura < 20) {
      // Acende azul e aciona buzzer a 1000Hz
      digitalWrite(pinoLEDR, LOW);
      digitalWrite(pinoLEDG, LOW);
      digitalWrite(pinoLEDB, HIGH);
      tone(pinoBuzzer, 1000);
      delay(500);
    }
    else {
      // Acende verde
      digitalWrite(pinoLEDR, LOW);
```

```
      digitalWrite(pinoLEDG, HIGH);
      digitalWrite(pinoLEDB, LOW);
    }

    // Desliga o buzzer
    noTone(pinoBuzzer);
  }
}
```

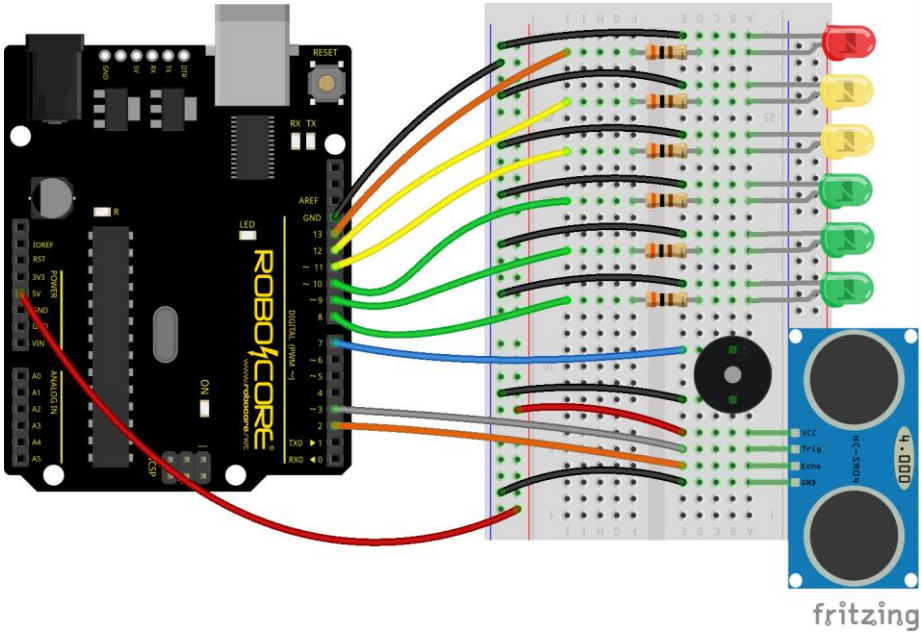


Projeto

Sensor de ré

Lista de materiais

- 1x Placa Arduino + Cabo USB
- 1x Protoboard
- 1x Sensor Ultrassônico - HC-SR04
- 1x Buzzer Passivo 5 V
- 6x Resistor 300 Ω
- 1x LED Vermelho 5 mm
- 2x LED Amarelo 5 mm
- 3x LED Verde 5 mm
- Jumpers




```
const int TRIG = 3, ECHO = 2, buzzer = 7;
unsigned int intervalo, distancia;

void setup() {
  Serial.begin(9600);
  pinMode(TRIG,OUTPUT);
  pinMode(ECHO,INPUT);
  pinMode(buzzer,OUTPUT);
```

```
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}

void loop() {
  distancia = sensor_morcego(TRIG,ECHO); // Chamada da função de leitura.
  Serial.println(distancia);
  if (distancia <= 10) { // Condicional para leituras inferiores a 10cm.
    digitalWrite(13,HIGH);
    digitalWrite(12,HIGH);
    digitalWrite(11,HIGH);
    digitalWrite(10,HIGH);
    digitalWrite(9,HIGH);
    digitalWrite(8,HIGH);
    tone(buzzer, 1750);
  }
  else if (distancia > 60) { // Condicional para leituras superiores a 60cm.
      digitalWrite(13,LOW);
      digitalWrite(12,LOW);
      digitalWrite(11,LOW);
      digitalWrite(10,LOW);
      digitalWrite(9,LOW);
      digitalWrite(8,LOW);
      noTone(buzzer);
  }
  else { // Condicional para leitura intermediarias.
    if (distancia <= 20) {
      digitalWrite(13,LOW);
      digitalWrite(12,HIGH);
      digitalWrite(11,HIGH);
      digitalWrite(10,HIGH);
      digitalWrite(9,HIGH);
      digitalWrite(8,HIGH);
      intervalo = 100;
    }
    else if (distancia <= 30) {
      digitalWrite(13,LOW);
      digitalWrite(12,LOW);
      digitalWrite(11,HIGH);
      digitalWrite(10,HIGH);
      digitalWrite(9,HIGH);
      digitalWrite(8,HIGH);
      intervalo = 150;
    }
```

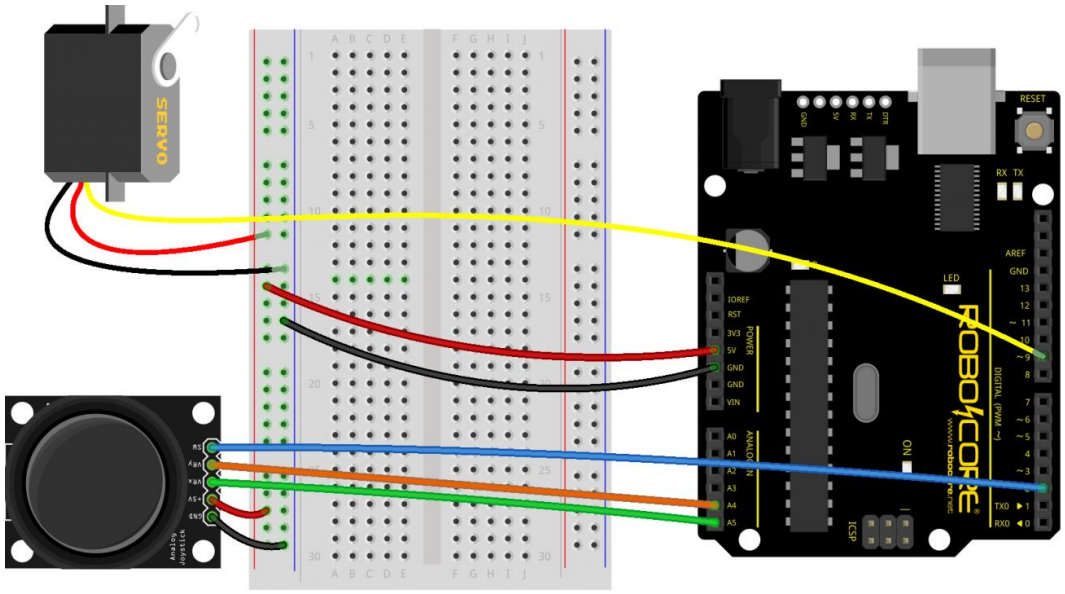
```
    else if (distancia <= 40) {
      digitalWrite(13,LOW);
      digitalWrite(12,LOW);
      digitalWrite(11,LOW);
      digitalWrite(10,HIGH);
      digitalWrite(9,HIGH);
      digitalWrite(8,HIGH);
      intervalo = 200;
    }
    else if (distancia <= 50) {
      digitalWrite(13,LOW);
      digitalWrite(12,LOW);
      digitalWrite(11,LOW);
      digitalWrite(10,LOW);
      digitalWrite(9,HIGH);
      digitalWrite(8,HIGH);
      intervalo = 250;
    }
    else if (distancia <= 60) {
      digitalWrite(13,LOW);
      digitalWrite(12,LOW);
      digitalWrite(11,LOW);
      digitalWrite(10,LOW);
      digitalWrite(9,LOW);
      digitalWrite(8,HIGH);
      intervalo = 300;
    }
    tone(buzzer,1750);
    delay(intervalo);
    noTone(buzzer);
    delay(intervalo);
  }
}

int sensor_morcego(int pinotrig,int pinoecho){ // Função para leitura do sensor
  digitalWrite(pinotrig,LOW);
  delayMicroseconds(2);
  digitalWrite(pinotrig,HIGH);
  delayMicroseconds(10);
  digitalWrite(pinotrig,LOW);

  return pulseIn(pinoecho,HIGH)/59;
}
```

Projeto

Garra Robótica

Lista de materiais

- 1x Placa Arduino com cabo USB
- 1x Protoboard
- 1x Módulo - Joystick
- 1x Garra Robótica Ant
- 1x Micro Servo 9g
- Jumpers



```
#include <Servo.h> //Inclui biblioteca para servo motor

const int SW = 2; // Pino para o botão do joystick
const int PIN_SERVO = 9; // Pino de comandos para o servo motor
const int VRX = A4; // Pino para leitura do eixo X
const int VRY = A5; // Pino para leitura do eixo Y

const int CLOSE = 170; // Limite de fechamento do servo
const int OPEN = 90; // Limite de abertura do servo

int leitura_vrx; // Variável de armazenamento do Eixo X
int leitura_vry; // Variável de armazenamento do Eixo Y
int leitura_sw; // Variável de armazenamento da leitura do botão
int posi_atual = 520; // Variável de referência
boolean abrir = 0; // Variável de sentido.
boolean fechar = 0; // Variável de sentido.

Servo servo_motor;

void setup() {
  pinMode(VRX, INPUT);
  pinMode(VRY, INPUT);
  pinMode(SW, INPUT_PULLUP);
  servo_motor.attach(PIN_SERVO);
}

void loop() {

  leitura_vry = analogRead(VRY);
  leitura_vrx = analogRead(VRX);
  leitura_vrx = map(leitura_vrx, 0, 1023, CLOSE, OPEN);  // Converte valor do potenciometro
  servo_motor.write(leitura_vrx); // Aciona servo motor, com base na leitura do eixo x
  delay(10);
  //Verificação da posição do joytick
  if (leitura_vry >= 550) {
    abrir = 1; // Seta a flag caso o movimento do joystick indicar uma tendencia de abertura
  }
  else if (leitura_vry <= 490) {
    fechar = 1; // Seta a flag caso o movimento do joystick indicar uma tendencia de fechamento
  }
  while (abrir == 1) { // Executa os comando que estiverem dentro das chaves
    leitura_vry = analogRead(VRY);
    leitura_sw = digitalRead(SW);

    if (leitura_sw == HIGH) { // Verifica o botão
```

```
    // Verifica a posição da garra
    if (leitura_vry > posi_atual) { // Se a posição da garra for maior que a posição anterior
      posi_atual = leitura_vry; // Atualiza a varivel de posição
      leitura_vry = map(leitura_vry, 0, 1023, CLOSE, OPEN);
      servo_motor.write(leitura_vry);
    }
   }
   // Atualiza a flag e a posiçao da garra se o botao for pressionado
   else {
    abrir = 0;
    posi_atual = 520;
   }
  }

  while (fechar == 1) { // Executa os comando que estiverem dentro das chaves
   leitura_vry = analogRead(VRY);
   leitura_sw = digitalRead(SW);

   if (leitura_sw == HIGH) { // Verifica o botão
    // Verifica a posição da garra
    if (leitura_vry < posi_atual) { // Se a posição da garra for maior que a posição anterior
      posi_atual = leitura_vry; // Atualiza a varivel de posição
      leitura_vry = map(leitura_vry, 0, 1023, CLOSE, OPEN);
      servo_motor.write(leitura_vry);
    }
   }
   // Atualiza a flag e a posiçao da garra se o botao for pressionado
   else {
    fechar = 0;
    posi_atual = 520;
   }
  }
}
```