

# ROBOT CLASS



Este guia é destinado a entusiastas da tecnologia e profissionais de engenharia, desenvolvedores e corporações no ramo da tecnologia que buscam implementar uma solução de organização de estacionamentos e de evitar acidentes na cidade.

O objetivo do Sistema de Monitoramento de Ponto Cego é criar um Sistema onde é bastante utilizado em carros, para detectar e ajudar ao estacionar o carro. Utilizaremos o Arduino como a ferramenta principal para construir este sistema, aproveitando sua simplicidade e flexibilidade para desenvolvedores de todos os níveis.

Neste guia, você aprenderá como montar um Sistema de Sensor, simples e prático, onde acenderá luzes e terá um barulho para evitar acidentes, e ter uma organização em estacionamentos ou em rodovias da cidade.

No Robot Class, começaremos com os conceitos básicos e, gradualmente, avançar para projetos mais complexos. Este guia é projetado para ser interativo e prático, permitindo que você aplique os conhecimentos de robótica e programação que você adquiriu em um contexto real e significativo.



# Sumário

## 1.0 Introdução

1.1 Guia didático de um Sistema de Monitoramento de Ponto Cego .....	4
1.2 Placa arduino .....	5
1.3 Software de programação .....	6
1.4 Conhecendo os componentes .....	7
1.5 Conhecendo os componentes .....	8
1.6 Funcionamento de Resistores.....	9

## 2.0 Passo a passo

2.1 Montagem do Projeto ; Seleção de Equipamentos .....	10
2.2 Montagem do Projeto : Conexão de Leds e Buzzers .....	11
2.3 Montagem do Projeto ; Conexões de Jumpers .....	12
2.4 Montagem do Projeto ; Conexão de fonte de energia .....	13
2.5 Montagem do Projeto ; Conclusão .....	14
2.6 Códigos; Definições e cálculo de som para centímetros .	15
2.7 Código; Configurando INPUT/OUTPUT e um loop .....	16
2.8 Código; Configurando as distâncias .....	17
2.9 Código; Configurando cálculo de medição .....	18
2.10 Código; Configurando sensor ultrassônico .....	19

## 3.0 Referências

3.1 Referências do projeto .....	20
----------------------------------	----

# Introdução

PROJETO

## 1.1 GUIA DIDÁTICO DE UM SISTEMA DE MONITORAMENTO DE PONTO CEGO

O projeto “Sistema de Monitoramento de Ponto Cego” utilizando *Arduino Uno* é uma oportunidade valiosa para explorar entre a teoria e prática na robótica. Ao fornecer uma aplicação dos conceitos fundamentais, os participantes entram em um ambiente de aprendizado dinâmico, onde podem desenvolver habilidades técnicas, experimentar, aprender com os erros e buscar soluções inovadoras.

Este projeto não só capacita os participantes a aplicarem conhecimentos de eletrônica, programação e mecânica de forma prática, mas também os desafia a pensarem de forma crítica e criativa para resolver problemas complexos. Através da montagem e programação do sistema de monitoramento.

# Introdução

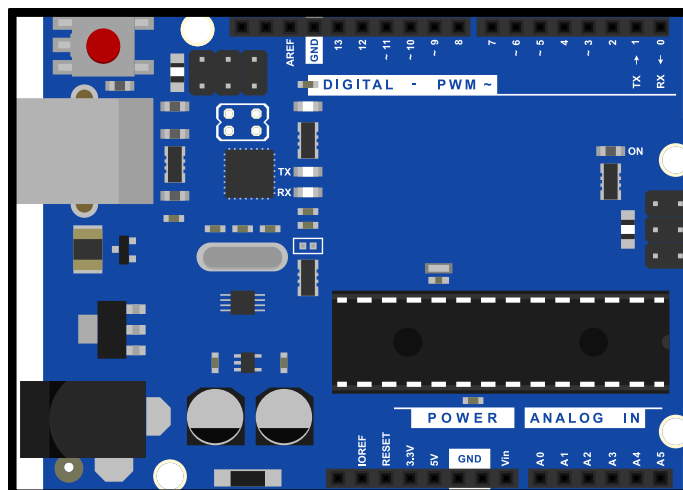
ROBÓTICA

## 1.2 PLACA ARDUINO

O *Arduino* é uma plataforma de eletrônica simples e acessível, perfeita para quem quer experimentar com eletrônica. É usado por estudantes, pesquisadores e pessoas de várias áreas. Podendo conectar coisas como sensores, motores, *LEDs* e monitores, usando uma linguagem de programação que se comunica com o *Arduino*.

A placa *Arduino Uno* tem 14 pinos que podem receber e enviar informações, e uma conexão *USB* que pode ser usado para passar energia para a placa ou se comunicar com ela por meio de um cabo.

Placa *Arduino Uno*.



Fonte: Canvas, 2024

# Introdução

ROBÓTICA

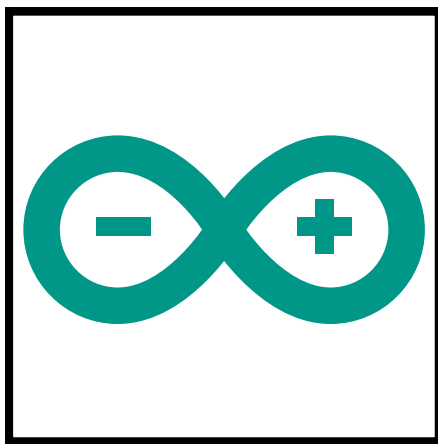
## 1.3 SOFTWARE DE PROGRAMAÇÃO

O *software* de programação do *Arduíno*, são chamados de *IDE*. É como um editor de texto especial para escrever programas que podem ser enviados para a placa *Arduíno*. Para instalar o *software*, é necessário acessar o site oficial e seguir as instruções.

Se não tiver uma placa *Arduíno* física, pode-se usar o *Tinkercad*, um programa online gratuito que permite criar e testar circuitos eletrônicos na internet.

A linguagem de programação do *Arduino* é semelhante ao inglês e ao português, ela é baseada em C/C ++. Basicamente, escrever comandos separados por ponto e vírgula e agrupados em blocos com chaves. Os comentários são escritos com duas barras e usados para deixar observações no seu código.

Software de programação  
*Arduino IDE*.



Fonte: Canvas, 2024

Programa online gratuito que  
permite testar e criar circuitos.



Fonte: Canvas, 2024

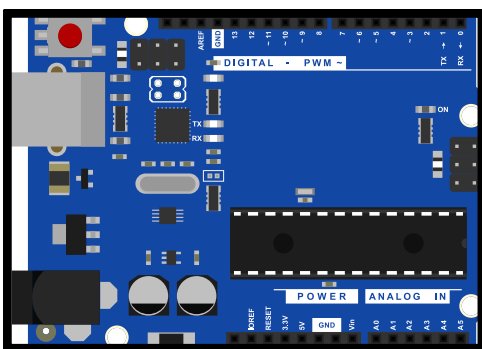
# Introdução

ROBÓTICA

## 1.4 CONHECENDO OS COMPONENTES

- **Arduino UNO:** É uma pequena placa que programa para realizar diversas tarefas. Ele possui entradas e saídas que permitem conectar sensores.
- **LEDs Vermelhos:** São as luzes que acendem para indicar que há algo no ponto cego. Eles são como os indicadores visuais do sistema, alertando sobre a presença de objetos na área monitorada.
- **Sensor Ultrassônico:** Ele mede quanto tempo leva para as ondas sonoras retornarem após atingirem um objeto. Isso ajuda a calcular a distância do objeto em relação ao sensor.

Arduino UNO



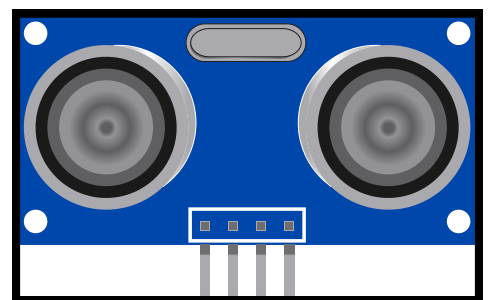
Fonte: Canvas, 2024

LEDs Vermelhos



Fonte: Canvas, 2024

Sensor Ultrassônico



Fonte: Canvas, 2024

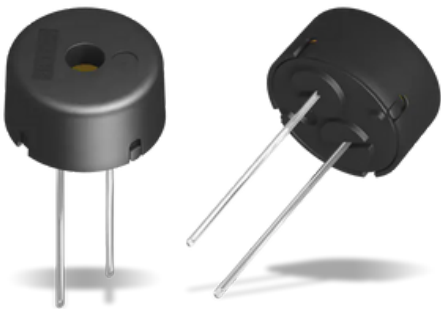
# Introdução

ROBÓTICA

## 1.5 CONHECENDO OS COMPONENTES

- **Buzzers:** São dispositivos que produzem som quando energizados. Eles podem ser úteis para fornecer alertas sonoros quando um objeto é detectado no ponto cego.
- **Protoboard:** É uma placa utilizada para montar circuitos eletrônicos sem a necessidade de solda. Que possui pequenos furos onde é possível encaixar e conectar os componentes eletrônicos de forma temporária.
- **Jumpers:** São pequenos cabos utilizados para fazer conexões elétricas entre os componentes na *protoboard*. Que facilitam a montagem e desmontagem do circuito, permitindo uma configuração flexível.

*Buzzers*



Fonte: PS Piezoelectronic Buzzers - TDK | Mouser, [br.mouser.com](https://br.mouser.com), 2015.

*Protoboard*



Fonte: [smartprojectsbrasil.com.br](https://smartprojectsbrasil.com.br), 2024.

*Jumpers*



Fonte: [eletrogate.com](https://eletrogate.com), 2024.



# Introdução

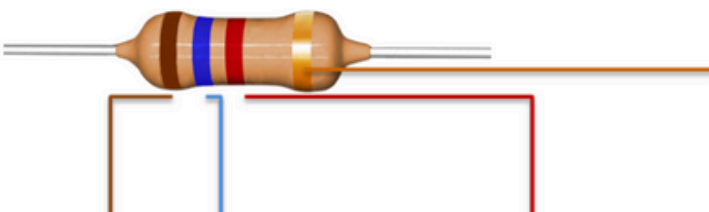
ROBÓTICA

## 1.6 FUNCIONAMENTOS DE RESISTORES

**Resistores de 330 Ohms:** São componentes eletrônicos que limitam a quantidade de corrente que flui por um circuito. Os resistores de 330 Ohms são usados para proteger outros componentes, como LEDs, garantindo que recebam a quantidade correta de energia.

Estes são apenas alguns exemplos, e a seleção do tipo de resistor dependerá das necessidades específicas do seu projeto no Arduino Uno. Certifique-se de verificar as especificações detalhadas de cada componente antes de utilizá-los.

Tabela de Resistores e suas voltagens.



Cor	1ª Faixa	2ª Faixa	3ª Faixa (se houver)	Multiplicador	Tolerância
Preta	0	0	0	x 1Ω	
Marrom	1	1	1	x 10Ω	± 1%
Vermelha	2	2	2	x 100Ω	± 2%
Laranja	3	3	3	x 1KΩ	
Amarela	4	4	4	x 10KΩ	
Verde	5	5	5	x 100K	
Azul	6	6	6	x 1MΩ	
Violeta	7	7	7	x 10MΩ	
Cinza	8	8	8		
Branca	9	9	9		
Dourada	-	-	-	x 0.1Ω	± 5%
Prateada	-	-	-	x 0.01Ω	± 10%
Sem cor	-	-	-	-	± 20%

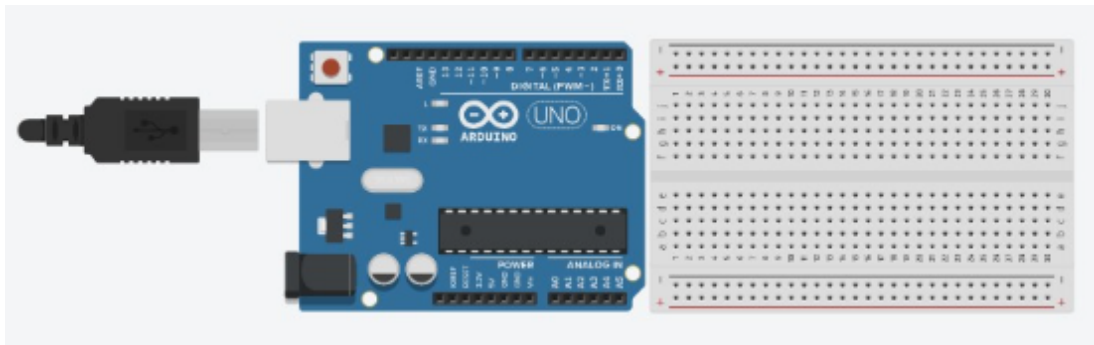
Fonte: <https://curtocircuito.com.br/blog/eletronica-basica/o-que-e-resistor>, 2024

# Passo a passo

ROBÓTICA

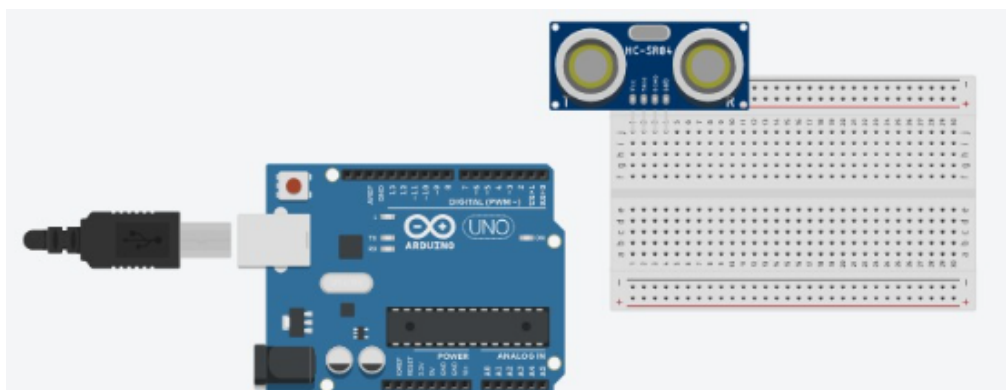
## 2.1 MONTAGEM DO PROJETO ; SELEÇÃO DE EQUIPAMENTOS

**Passo 1:** No simulador “Tinkercad”, selecionamos criar um novo circuito e colocamos um “Arduino Uno” e uma “Protoboard”.



Fonte: Tinkercad, 2024.

**Passo 2:** Coloque um “Sensor Ultrassônico” na “Protoboard” com os pinos *VCC*, *TRIG*, *ECHO* e *GND* conectados em 1, 2, 3 e 4, respectivamente. O pino *TRIG* controla o envio de pulsos ultrassônicos, enquanto o pino *ECHO* recebe pulsos refletidos, com informações sobre a distância. O pino *GND* é a referência elétrica, conectada ao potencial zero, e o pino *VCC* fornece a alimentação ao sensor, geralmente conectado a uma fonte de energia (como 5V ou 3.3V).



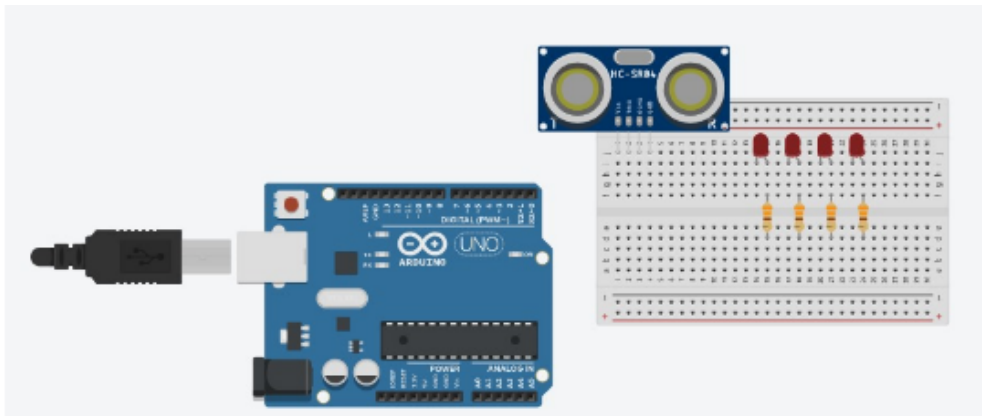
Fonte: Tinkercad, 2024.

# Passo a passo

ROBÓTICA

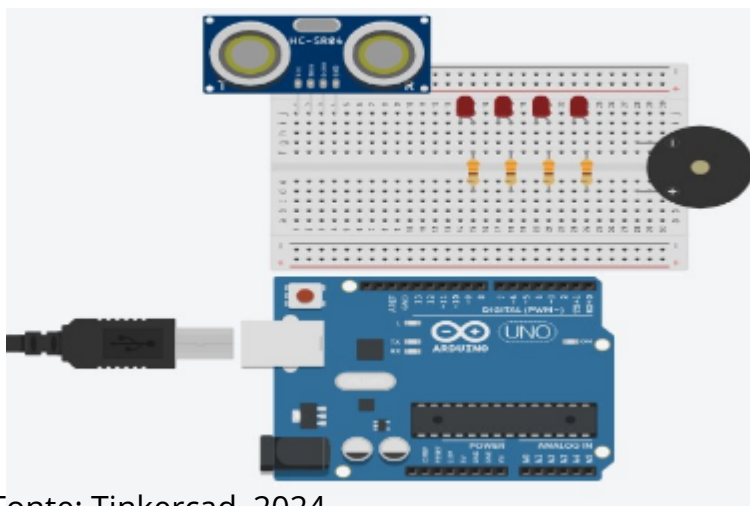
## 2.2 MONTAGEM DO PROJETO; CONEXÕES DE LEDS E BUZZER

**Passo 3:** Colocaremos os *LEDs* de qualquer cor e os resistores de 330 *ohms* conectados na placa. Os *LEDs*, onde a perna que tem a curvatura simboliza a perna positiva do *LED*, são geralmente as maiores na realidade, e as que não têm são as negativas. Depois, serão colocados os resistores de 330 *ohms* que serão conectados à perna negativa do *LED*, realizando a ligação com as duas partes.



Fonte:Tinkercad, 2024.

**Passo 4:** Neste passo, o *buzzer* foi conectado à *protoboard*. O *buzzer* tem a funcionalidade de emitir sons. Seu pino positivo foi conectado a D28 e o negativo a G28.



Fonte: Tinkercad, 2024.

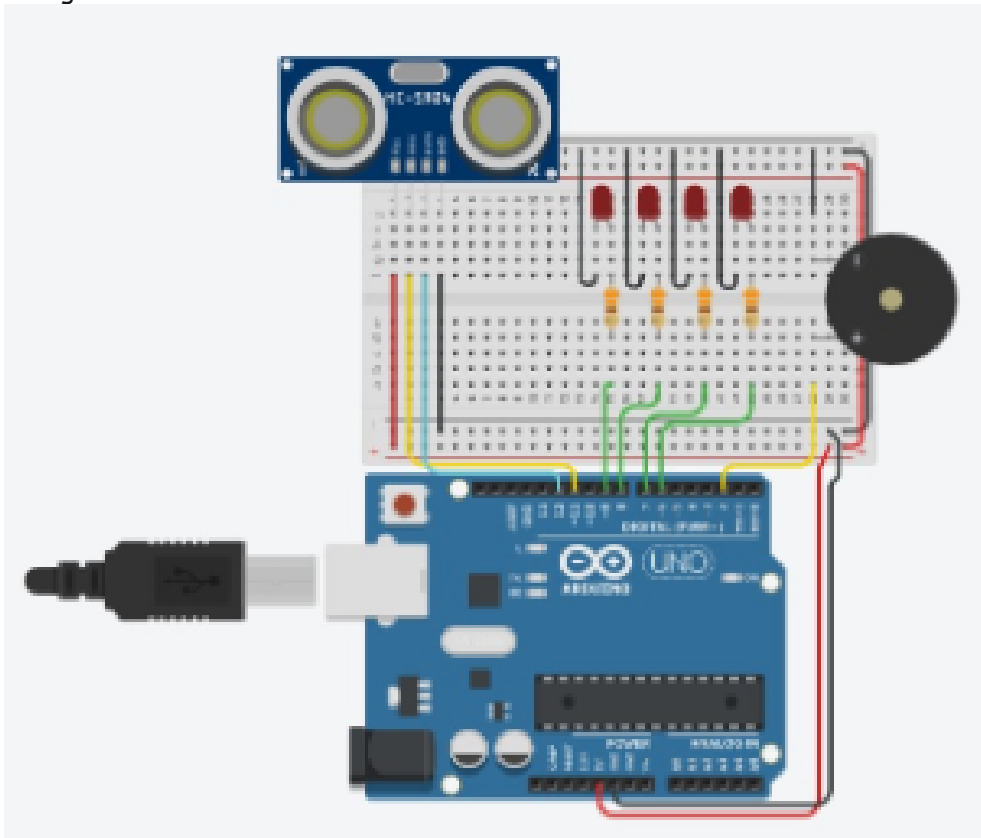


# Passo a passo

ROBÓTICA

## 2.4 MONTAGEM DO PROJETO; CONEXÕES DE FONTE DE ENERGIA

**Passo 6:** Os LEDs vermelhos estão conectados aos seguintes pinos do Arduino: D6, D7, D8 e D9. Cada um deles está ligado ao positivo, sendo D6 à protoboard A24, D7 à protoboard A21, D8 à protoboard A18 e D9 à protoboard A15. Os outros pinos dos LEDs estão conectados ao GND (terra) do Arduino usando jumpers para comunicação.



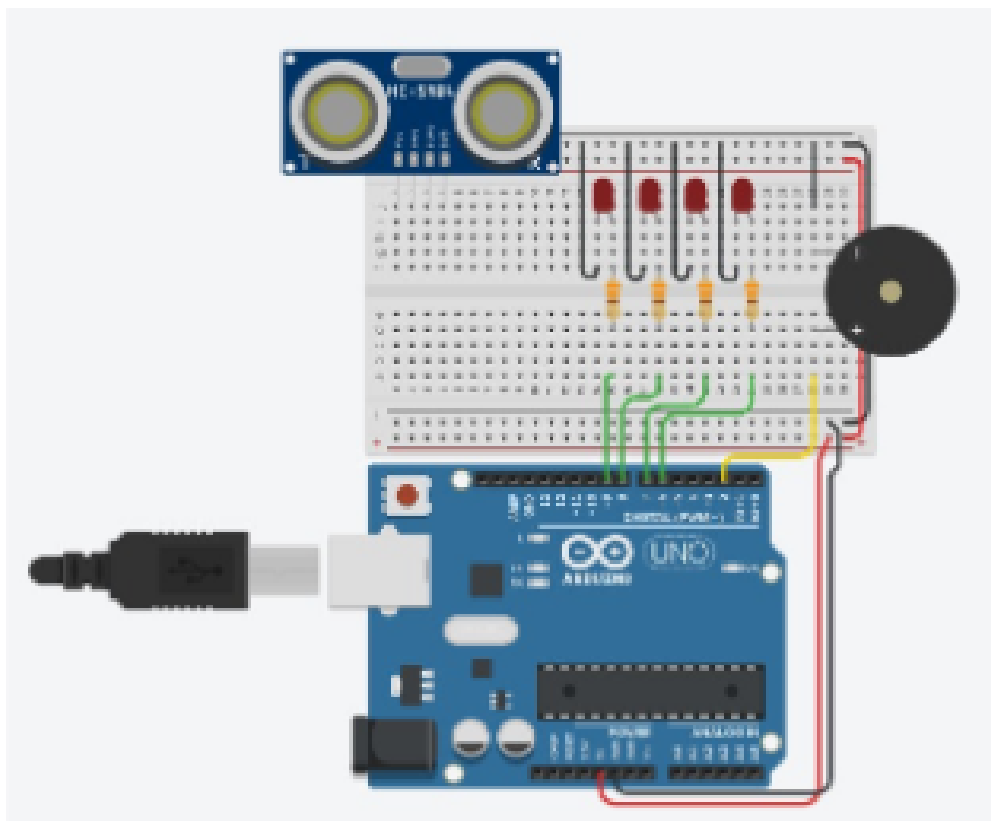
Fonte: Tinkercad, 2024.

# Passo a passo

ROBÓTICA

## 2.5 MONTAGEM DO PROJETO; CONCLUSÃO

**Passo 7:** Conectamos o sensor *ultrassônico* em dois pinos digitais do *Arduino*: o pino de *trigger* (pino 10) e o pino de *echo* (pino 11). Além disso, conectamos um pino do sensor no 5V e outro no *GND* da *protoboard*.



Fonte: Tinkercad, 2024.

# Passo a passo

ROBÓTICA

## 2.6 CÓDIGO; DEFINIÇÕES E CÁLCULO DE SOM PARA CENTIMETROS

### Passo 1- DEFINIÇÕES

É preciso que nas primeiras linhas de código identifiquemos as portas de cada componente eletrônico conectado ao *Arduino*. Para isso, usamos o `#define` e na frente identificamos qual é o componente e qual é a porta que está conectado.

```
#include <Ultrasonic.h>
#define LED1 6
#define LED2 7
#define LED3 8
#define LED4 9
#define TRIGGER 11
#define ECHO 12
#define BUZZER 2
```

### Passo 2- CALCULO DE SOM PARA CENTIMETROS

A constante som definida como 34300,0 representa a velocidade do som no ar, usada para calcular a distância medida por um sensor ultrassônico. A velocidade do som varia com a temperatura e pressão atmosférica, mas em condições normais é aproximadamente 343 m/s ou 34300 cm/s. Esta constante é essencial para o cálculo da distância, ajustado pelo tempo que o pulso ultrassônico leva para retornar ao sensor, dividido pela metade (considerando o caminho de ida e volta) e pela velocidade do som. A função `calcularDistancia()` usa essa fórmula para determinar a distância do sensor até o objeto detectado.

```
// Constantes utilizadas
const float som = 34300.0; // Velocidade do som em cm/s

void setup() {

  // Inicie a comunicação Serial a 9600bps
  Serial.begin(9600);
```

# Passo a passo

ROBÓTICA

## 2.7 CÓDIGO CONFIGURANDO INPUT/OUTPUT E UM LOOP

### Passo 3- CONFIGURANDO *INPUT* (ENTRADA) e *OUTPUT* (SAÍDA)

Utilizando a função *pinMode* configuramos quais pinos são de entrada/saída, as de entradas podem receber sinais digitais e as de saídas podem fornecer sinais digitais, como acender um *LED* e um sensor *ultrassônico*, e um *buzzer* para alerta.

```
// Modo de entrada / saída do pino
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
pinMode(LED4, OUTPUT);
pinMode(ECHO, INPUT);
pinMode(TRIGGER, OUTPUT);
pinMode(BUZZER, OUTPUT);

// Função que desliga os LEDs
desligarLEDs();
```

### Passo 3- INICIALIZAÇÃO DE UM LOOP

```
void loop() {

// Inicializa o sensor ultrassônico
inicializarTrigger();

// Obtenha a distância
float distancia = calcularDistancia();

// Desligar os LEDs
desligarLEDs();

// Alerta se estiver dentro da zona de perigo
if (distancia < 25){
// Dispara os alertas
alertas(distancia);
}
}
```

A função *void loop()* no código *Arduino Uno* realiza as seguintes ações em um ciclo contínuo:

- Inicializa o sensor *ultrassônico* enviando um pulso curto para iniciar a medição da distância.
- Calcula a distância medida pelo sensor *ultrassônico*.
- Desliga os *LEDs* para preparar a próxima medição.



# Passo a passo

ROBÓTICA

## 2.8 CÓDIGO CONFIGURANDO AS DISTÂNCIAS

### Passo 4- CONFIGURANDO AS DISTÂNCIAS

Usando o comando *digitalWrite* que nos permite escrever um valor *HIGH* ou *LOW*, configuramos os *LEDs* e um sensor *ultrassônico* para detectar objetos próximos, acendendo *LEDs* e emitindo sons com um *buzzer*. Ele começa com *LEDs* apagados e, em um *loop*, mede a distância do sensor *ultrassônico*. Se um objeto estiver próximo, os *LEDs* acendem e o *buzzer* faz som conforme a distância que determinamos .

```
// Função de desligar os LEDs
void desligarLEDs()
{
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
}

// Verifique se algum alerta visual ou sonoro é necessário
void alertas (float distancia){

  // Se a distância estiver entre 30 e 20 cm
  if (distancia <= 30 && distancia > 20){
    // Ligue o LED 1
    digitalWrite(LED1, HIGH);
    tone(BUZZER, 1500, 50);
  }
}
```

# Passo a passo

ROBÓTICA

## 2.9 CÓDIGO CONFIGURANDO CALCULO DE MEDIÇÃO

### Passo 5- CONFIGURANDO CALCULO DE MEDIÇÃO

A função `calcularDistancia()` é usada para medir a distância entre o sensor *ultrassônico* e um objeto. Ela mede o tempo que um pulso ultrassônico leva para ir ao objeto e voltar, calcula a distância com base na velocidade do som e exibe a distância medida no monitor serial. Essa medição permite que o sistema responda com *LEDs* e um *buzzer*, indicando a proximidade de um objeto.

```
// Função usada para calcular a distância entre o som e o objeto
float calcularDistancia(){
  unsigned long time = pulseIn(ECHO, HIGH);
  float distancia = time* 0.000001 * som / 2.0;
  Serial.print(distancia);
  Serial.print("cm");
  Serial.println();
  delay(250);
  return distancia;
}
```

# Passo a passo

ROBÓTICA

## 2.10 CÓDIGO CONFIGURANDO SENSOR ULTRASSÔNICO

### Passo 5- CONFIGURANDO SENSOR ULTRASSÔNICO

A função `inicializarTrigger()` prepara o sensor *ultrassônico* para medir a distância, enviando um pulso curto para o pino `TRIGGER` do sensor. Primeiro, o pino é definido como `LOW` por 2 microssegundos, depois como `HIGH` por 10 microssegundos para emitir o pulso *ultrassônico*, e retorna finalmente a `LOW`. Este processo é essencial para calcular a distância com precisão, ao permitir que o pulso *ultrassônico* seja refletido de volta ao sensor para calcular a distância do objeto.

```
// Função usada para inicializar o Trigger
void inicializarTrigger(){
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
}
```

# Referências

ROBÓTICA

## 3.0 REFERÊNCIAS DO PROJETO

Aula 05 - Construção de um sensor de estacionamento com Arduino no TINKERCAD; Disponível em: <<https://www.youtube.com/watch?v=q1NAqFh5nLY>>.

MOTA, A. HC-SR04 - Sensor ultrassônico com Arduino. Disponível em: <<https://portal.vidadesilicio.com.br/hc-sr04-sensor-ultrassonico/>>.

TINKERCAD. Tinkercad | From mind to design in minutes. Disponível em: <<https://www.tinkercad.com/>>.

Sensor Ultrassônico Arduino HCSR04, JSN-SR04T e Outros. Disponível em: <<https://www.usinainfo.com.br/sensor-ultrassonico-502>>.