

# Sushi Cards

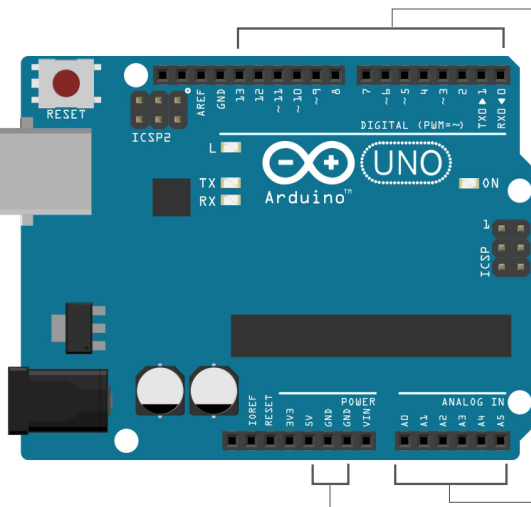
CLUBE DO  
ARDUINO  
OLIDO

# O que é Arduino?

Arduino é uma placa que consegue ler informações de botões e sensores de vários tipos como movimento, som, luminosidade, umidade e, através de um processamento, transforma em saídas para controlar luzes, motores e outros atuadores. Esse processamento é feito através de um código que você pode criar para controlar seu projeto!

Lembra do processamento que o Arduino faz para transformar as informações recebidas em comandos para as luzes, motores e etc? Então, esse processamento é feito através de um código desenvolvido pelo usuário. Esse código é carregado (gravado) no Arduino. Essa gravação acontece via **porta USB**! Você vai precisar conectar o cabo aqui nessa entrada e na USB do computador

Essa aqui é a **entrada de energia** do Arduino. Ele pode ser ligado por aqui ou pela entrada USB ali em cima. Você tem que ligar uma fonte entre 7V a 20V no máximo. Mas tá ótimo se ligar com 9V (uma bateria grande, por exemplo).



O Arduino Uno tem 14 **pinos digitais** que vão de 0 à 13. Esses pinos são usados tanto para receber como para enviar informações. Mas eles só funcionam com os valores: 0 e 1 (ligado e desligado, aceso e apagado, aberto e fechado).

Outros pinos importantes são os **pinos analógicos** que vão de A0 à A5. Esses pinos só recebem informações. Só que eles podem receber valores mais variados como a intensidade luminosa de um ambiente, a distância captado por um sensor, a umidade do ar.

Esses são os pinos **5V** e **GND**. Eles são o positivo e negativo, respectivamente. Todo circuito eletrônico precisa de um positivo e um negativo para funcionar.

# Arduino IDE

IDE Arduino é um programa onde são colocados os códigos para programar o Arduino. Cada código é chamado de **Sketch**. IDE significa Integrated Development Environment ou Ambiente de Desenvolvimento Integrado. O programa pode ser baixado em [arduino.cc](http://arduino.cc)

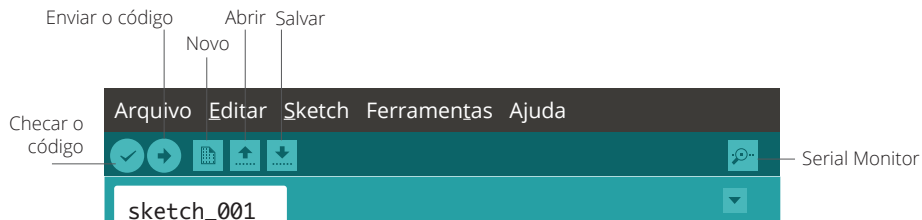
## Vamos carregar um sketch no Arduino?

Vamos usar o exemplo Blink. Vá em **Arquivo >> Exemplos >> Basics** e escolha o Blink. Aparecerá uma nova janela com um sketch pronto.

Para carregar qualquer sketch no Arduino é preciso configurar Placa e Porta dentro da IDE. Isso se faz necessário porque existem diversas placas de Arduino diferentes e a IDE precisa saber qual estamos usando.

Para configurar a **Placa** vá em **Ferramentas >> Placa** e escolha Genuino UNO (o modelo mais comum e popular entre os Arduinos).

Para configurar a **Porta** vá novamente em **Ferramentas >> Porta** e escolha a porta onde o Arduino foi conectado.



```
void setup() {  
  // coloque seu código aqui para rodar uma vez:  
  
}  
  
void loop() {  
  // coloque seu código principal para rodar repetidamente:  
  
}
```



# Conceitos básicos

## O que são Volts, Ampère e Ohms?

Pegue uma pilha comum e veja que está escrito 1,5V. E o que significa isso? Que entre o pólo negativo (-) e positivo (+) há 1,5 de energia elétrica em potencial ou **tensão elétrica (V)** e sua unidade de medida é o **Volts (V)**.

Se conectarmos esses dois pólos com um fio, nos primeiros minutos sentiremos o fio esquentar e depois a pilha também esquentar.

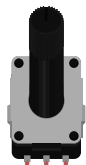
O que aconteceu? Houve um fluxo de energia elétrica do pólo positivo para o negativo durante o período que o fio estava conectado, o que chamamos de **corrente elétrica (I)** e sua unidade de medida é o **Ampère (A)**.

O fio foi o condutor por onde a corrente elétrica passou. Esse meio é o que chamamos de **resistência elétrica (R)** e sua unidade de medida é o **Ohm ( $\Omega$ )**. Como o fio ofereceu pouca resistência, o fluxo da corrente elétrica foi mais intenso, o que gerou mais calor.



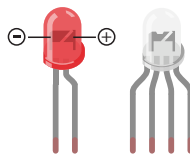
## Resistor

É um componente que, como o nome diz, oferece uma resistência à passagem da corrente elétrica. É utilizado para consumir um pouco da energia do circuito e dessa forma equilibrar e proteger outros componentes. Cada resistor tem um valor de resistência diferente. Por exemplo, um resistor de valor 220  $\Omega$  (se lê 220 Ohms) é identificado através do código de cores impresso no próprio componente. Para isso usamos a **tabela de cores** de resistores.



## Potenciômetro

É um resistor com **resistência variável**. Seu pino móvel percorre uma resistência (material feito geralmente de grafite). Esse componente tem 3 pinos, o pino central - o que vai variar a resistência interna e os pinos laterais que devem ser conectados ao positivo e negativo do circuito (sem polaridade).



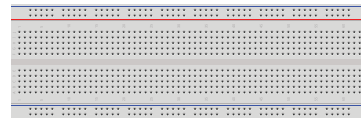
## Led

É um componente que emite luz quando lhe é aplicada uma corrente elétrica. É um semicondutor (diodo) emissor de luz (**Light Emitter Diode**). Suas cores mais frequentes são branca, vermelha, amarela, verde e azul. O led tem polaridade, portanto, há um jeito correto para conectá-lo ao circuito. Há 3 formas de identificar a polaridade de um led:

**1 - Tamanho das pernas:** perna maior é o pólo positivo, perna menor é o negativo;

**2 - Se olharmos o led contra luz** veremos que há um pedaço de metal dividido em dois em seu interior: a parte menor é o pólo positivo, a maior é o negativo;

**3 - No invólucro plástico do led** há um chanfro na borda que indica o lado do pólo negativo.

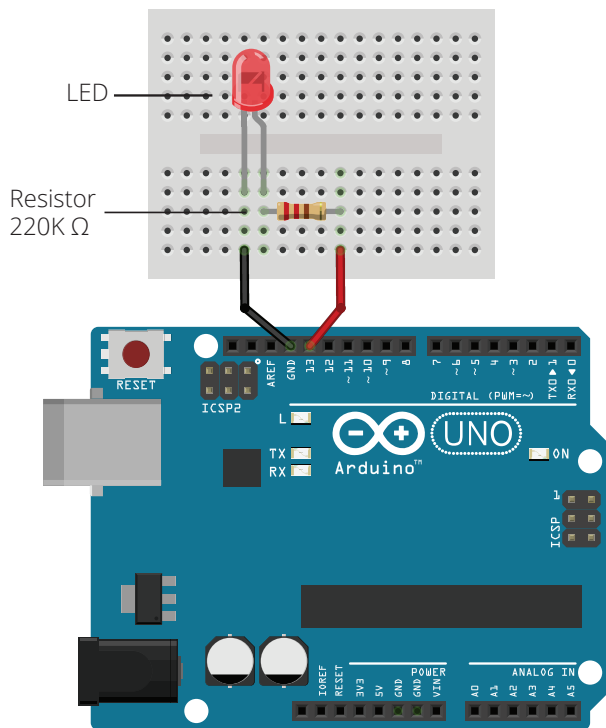


**Protoboard** é uma **placa para prototipação** (construção) rápida e fácil de circuitos eletrônicos. Ela possui trilhas internas que conduzem eletricidade. Há 4 linhas laterais, 2 de cada lado, que possuem marcações de positivo e negativo. Cada linha tem trilhas internas interconectando cada buracinho, ou seja, se conectar um pino no primeiro buraco, ele estará conectado a tudo que estiver encaixado nos outros buracos da linha. Dessa forma, ao conectar o pólo negativo do circuito na linha, haverá uma multiplicação de acessos a esse pólo do circuito. A parte interna é interconectada também, porém a ordem é de que cada 5 buracos forma uma coluna independente da coluna do outro lado da marca central.



## Jumper

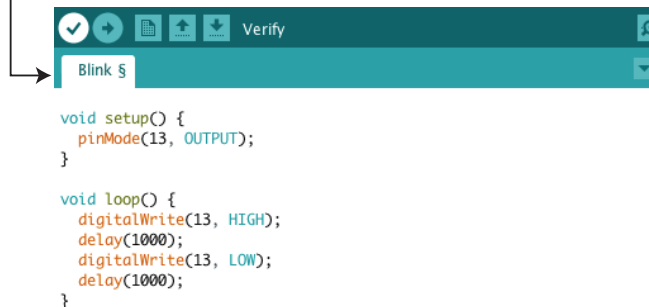
Jumpers são os **fios** utilizados para fazer as conexões elétricas e montar o circuito no Protoboard.



# Blink




// Liga o LED da placa por 1 segundo, depois desliga por um segundo, repetidamente

- 1- Abra o **Arduino IDE**
- 2- Conecte o cabo **USB** do Arduino no computador
- 3- Digite o código abaixo:



```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

- 4- Clique em  para checar se o código foi escrito corretamente
- 5- Selecione a sua versão da **placa** Arduino em Ferramentas
- 6- Selecione a **porta** em que o Arduino está conectado
- 7- Carregue o código no Arduino 
- 8- Abrir o Monitor Serial  para ver o resultado

## Comentários

// Descubra o que significa cada linha que você digitou

```
//A função setup roda uma vez depois que se reinicia ou  
liga a placa
```

```
void setup() {  
  pinMode(13, OUTPUT); //Define o pino 13(LED) como saída  
}
```

```
//A função loop roda continuamente sem parar
```

```
void loop() {  
  digitalWrite(13, HIGH); //Liga o LED  
  delay(1000); //Aguarda 1 segundo  
  digitalWrite(13, LOW); //Apaga o LED  
  delay(1000); //Aguarda 1 segundo  
}
```

Nesse exercício você aprendeu as seguintes funções:

**pinMode**  
**digitalWrite**  
**delay**

E as seguintes variáveis:

**OUTPUT**  
**HIGH**  
**LOW**



## Desafio


Se você conseguiu fazer o exercício, parabéns! Mas a brincadeira não acabou, veja se consegue resolver os 3 desafios abaixo:

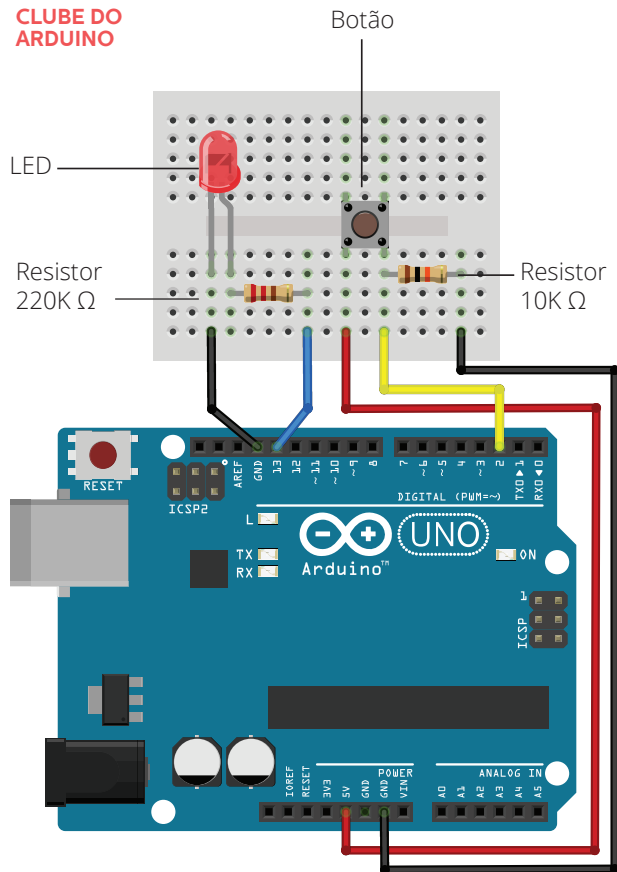
**1-** O tempo que o led fica aceso e apagado é contado em milissegundos. **1000 milissegundos = 1 segundo**. E se você quisesse o led piscando mais rápido, digamos a cada meio segundo, qual o valor que se deve colocar? Mude o código, reenvie para o arduino e veja se funciona.

**2-** O led está conectado no pino digital 13, você pode usar outros **pinos digitais** (ver sushicard nº1). Para isso, terá que mudar essa informação no código e reenviá-la para a placa. Consegue descobrir como fazer isso?

**3-** Supondo que você quisesse colocar mais um led fazendo a mesma coisa, como faria? Lembre-se que ao usar um novo pino digital você terá que **declará-lo no início do código**, só depois disso que poderá programar o que fazer com ele.

**Dica1:** Todo led tem polaridade, ou seja, uma perninha é positiva (mais longa) e outra negativa (mais curta). A positiva é ligada no pino digital e a negativa no **GND** (existem 3 pinos GND em pontos diferentes na placa, pode usar qualquer um).

**Dica 2:** Pode copiar e colar parte do código pra não ter que digitar de novo, só preste atenção onde ele deve ficar e se não falta nenhum caractere (**colchetes** e **ponto e vírgula** são muito importantes!) use o checkbox  pra verificar e boa sorte!



# Led com botão

// O botão comanda o Led. Botão pressionado, led aceso.

- 1- Abra o **Arduino IDE**
- 2- Conecte o cabo **USB** do Arduino no computador
- 3- Digite o código abaixo:

Arquivo Editar Sketch Ferramentas Ajuda

sketch\_botao\_led




```
int led = 13;
int botao = 3;
int estado = 0;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(botao, INPUT);
}

void loop() {
  estado = digitalRead(botao);

  if (estado == HIGH) {
    digitalWrite(led, HIGH);
  }

  else {
    digitalWrite(led, LOW);
  }
}
```

- 4- Clique em  para checar se o código foi escrito corretamente
- 5- Selecione a sua versão da **placa** Arduino em Ferramentas
- 6- Selecione a **porta** em que o Arduino está conectado
- 7- Carregue o código no Arduino 
- 8- Abrir o Monitor Serial  para ver o resultado

## Comentários

// Descubra o que significa cada linha que você digitou

```
/* Cria 3 variáveis para armazenar 3 valores diferentes, isso serve para organizar melhor o código. Ex: quando for usar o número 13 (pino 13), basta escrever "led" ao invés do número */
```

```
int led = 13;
int botao = 2;
int estado = 0; // Essa variável servirá para armazenar o estado do botão. Pressionado tem o valor 1 (HIGH) e não pressionado tem o valor 0 (LOW).
```

```
void setup() {
  pinMode(led, OUTPUT); // "led" representa o pino 13
  pinMode(botao, INPUT); // "botao" representa o pino 2, que // é definido como entrada
}
```

```
void loop() {
  /* Primeiro é feita a leitura do pino onde o botão está conectado: pressionado (HIGH), não pressionado (LOW) */
```

```
    estado = digitalRead(botao);
```

```
/*Depois disso, tomamos a decisão: Para o estado HIGH acendemos o LED, para estado LOW, apagamos. A estrutura de decisão IF/ELSE (SE/SENÃO) indica uma condição para que a ação ocorra. O sinal "==" indica que quando a condição entre a variável estado e o valor for de igualdade ocorre uma ação */
```

```
    if (estado == HIGH) { // Compara a variável estado com o valor HIGH.
      digitalWrite(led, HIGH); // Acende o LED se o botão estiver pressionado
    }
```

```
    else { // SENÃO estiver pressionado, FAÇA a ação abaixo
      digitalWrite(led, LOW); // Apaga o LED
    }
```

## Desafio

**1-** Faça com que quando o botão estiver livre, o LED acenda. E ao pressionar o botão, o led se apague. Você terá que usar a mesma estrutura de **decisão IF/ELSE**, mas agora suas ações serão **opostas**.

**2-** Conecte um outro LED em outro pino e **declare sua variável**, não esqueça da parte de configuração desse pino, e faça com que o botão, ao ser pressionado acenda um e apague o outro e inverta isso quando não estiver pressionado.

**3-** Agora use a função **delay** para que ao pressionar o botão o LED fique aceso por 800 milissegundos.

**Dica:** Quando o botão está pressionado, o circuito fecha e a informação HIGH chega ao pino digital. Mas e quando ele não está pressionado? Se não houvesse nada ligado o circuito ficaria aberto, o pino digital receberia **ruídos elétricos, estática** e o programa não funcionaria corretamente. É por isso que ligamos um **resistor** do GND ao botão e pino digital, para garantir que quando o botão não estiver pressionado, o pino esteja recebendo o valor LOW.

Nesse exercício você aprendeu a seguinte função:

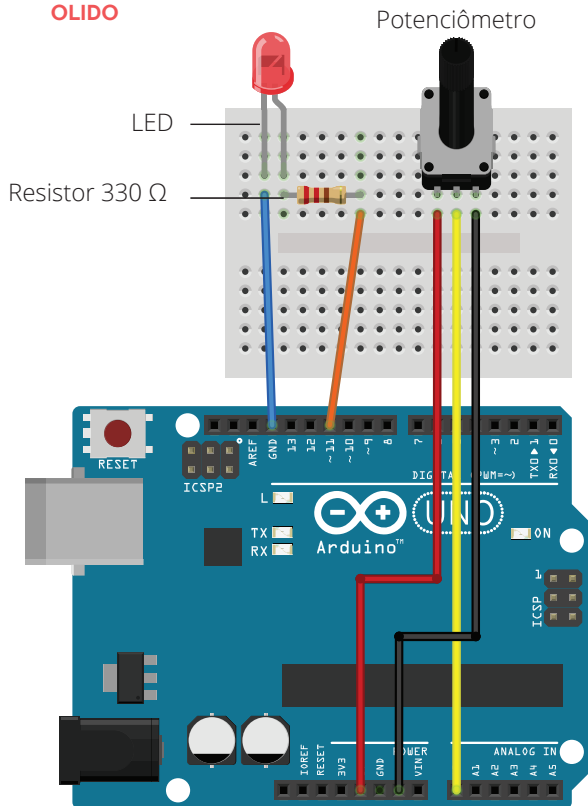
**digitalRead**

Estrutura de decisão **if/else**

E a seguinte variável: **int**







# Potenciômetro

// O potenciômetro controla a luminosidade do Led.

- 1- Abra o **Arduino IDE**
- 2- Conecte o cabo **USB** do Arduino no computador
- 3- Digite o código abaixo:

```
Arquivo  Editar  Sketch  Ferramentas  Ajuda
[ícones]
sketch_pot

int led = 11;
int pot = A0;
int valor = 0;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  valor = analogRead(pot);
  valor = map(valor, 0, 1023, 0, 255);
  analogWrite(led, valor);
}
```

- 4- Clique em para checar se o código foi escrito corretamente
- 5- Selecione a sua versão da **placa** Arduino em Ferramentas
- 6- Selecione a **porta** em que o Arduino está conectado
- 7- Carregue o código no Arduino
- 8- Abra o Monitor Serial para ver o resultado

## Comentários

// Descubra o que significa cada linha que você digitou

```
int led = 11;
int pot = A0; // A0 é um dos pinos analógicos do Arduino
int valor = 0;

void setup() {
  /* Como os pinos analógicos são SÓ ENTRADAS diferentemente dos
  pinos digitais que podem ser entradas ou saídas, não precisamos
  declarar que são entradas usando pinMode como fazemos com os
  digitais.*/

  pinMode(led, OUTPUT);
}

void loop() {

  valor = analogRead(pot); // Lê a entrada analógica (pino A0)
                          // e armazena o valor em "valor"

  /* Os pinos analógicos funcionam em uma escala diferente dos
  digitais. Quando recebemos um sinal de entrada pelo pino
  analógico e queremos dar saída desse sinal num pino digital
  precisamos fazer uma conversão entre as duas escalas. Para isso
  usamos a função map abaixo. Aqui ela pega a variável "valor" e
  transforma uma escala de 0-1023 (entrada) em uma escala 0-255
  (saída). */

  valor = map(valor, 0, 1023, 0, 255);

  /* Diferentemente da função digitalWrite, que envia os
  valores binários (1 ou 0) para os pinos digitais, a
  função analogWrite envia qualquer valor entre 0 e 255. Dessa
  forma conseguimos variar a intensidade luminosa do LED. */

  analogWrite(led, valor); // Envia o número "valor" para o pino
  "led"
}
```

## Desafio

- 1- Vamos treinar a inclusão de outro **pino digital** com LED? Inclua outro LED no projeto e faça com que o potenciômetro controle a **intensidade luminosa** dos dois.
- 2- Reconstrua o mesmo projeto do Sushi Card Blink. Sim, aquele onde o LED fica piscando num tempo definido. Mas agora faça com que o potenciômetro controle o **tempo** em que o LED fica aceso e apagado.

**Dica 1:** Os **pinos digitais** que conseguem enviar valores diferentes de 1 e 0 são os que estão acompanhados dos sinais "~". Observe no Arduino. Esses são os pinos **PWM (Pulse Width Modulation)**, ou seja, conseguem controlar a largura do pulso e assim enviar sinais com essa variação.

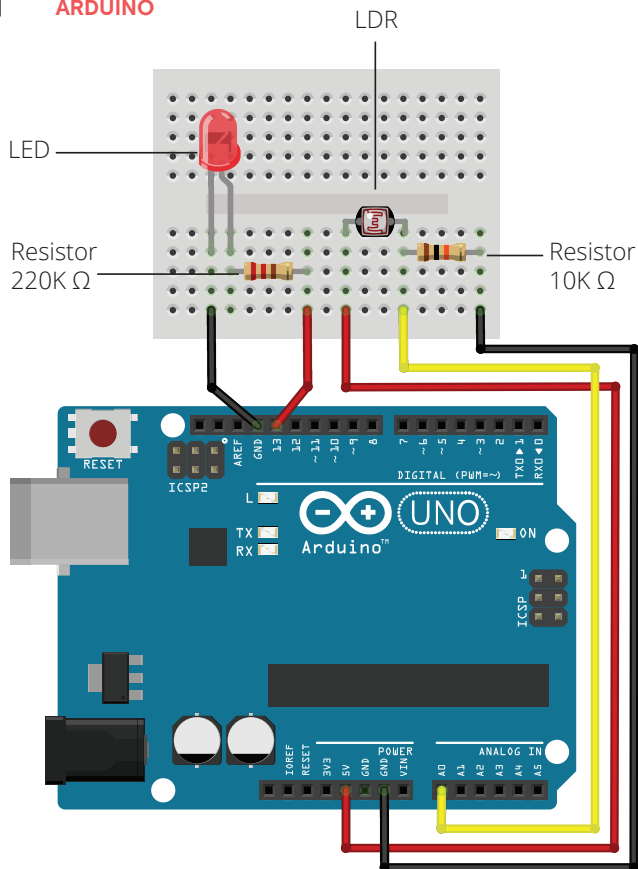
**Dica 2:** A ligação do potenciômetro sempre será da seguinte forma: são 3 pinos, os dois nas extremidades são **GND** e **5V**, que não possuem polaridade, portanto podem ser ligados na fonte de energia de qualquer lado, e o do meio que é ligado no **pino analógico** do Arduino.

Nesse exercício você aprendeu as seguintes funções:

**analogRead**  
**analogWrite**  
**map** - Para fazer conversão entre escalas

Pino analógico não precisa ser configurado como entrada  
Para que servem os pinos **PWN**





# LDR com Led

// O botão comanda o Led. Botão pressionado, led aceso.

- 1- Abra o **Arduino IDE**
- 2- Conecte o cabo **USB** do Arduino no computador
- 3- Digite o código abaixo:

Arquivo Editar Sketch Ferramentas Ajuda

✓ ↻ 📄 ⚙️ ⬇️

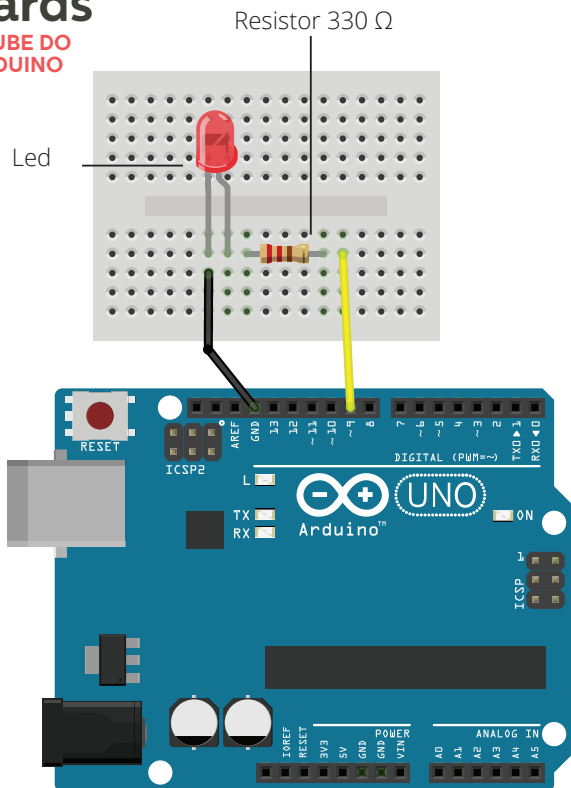
sketch\_ldr\_led

```
int led = 9;
int ldr = 0;
int ldrVal = A0;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  ldrVal = analogRead(ldr);
  ldrVal = map(ldrVal, 0, 1023, 0, 255);
  analogWrite(led, ldrVal);
  delay(2);
}
```

- 4- Clique em para checar se o código foi escrito corretamente
- 5- Selecione a sua versão da **placa** Arduino em Ferramentas
- 6- Selecione a **porta** em que o Arduino está conectado
- 7- Carregue o código no Arduino
- 8- Abrir o Monitor Serial para ver o resultado



# Fade

// Mostra a alteração progressiva do brilho de um LED




- 1- Abra o **Arduino IDE**
- 2- Conecte o cabo **USB** do Arduino no computador
- 3- Digite o código abaixo:

```
Arquivo  Editar  Sketch  Ferramentas  Ajuda
[Icons]
sketch_fade

int led = 9;
int brilho = 0;
int fade = 5;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  analogWrite(led, brilho);
  brilho = brilho + fade;
  if (brilho == 0 || brilho == 255) {
    fade = -fade;
  }
  delay(30);
}
```

- 4- Clique em  para checar se o código foi escrito corretamente
- 5- Selecione a sua versão da **placa** Arduino em Ferramentas
- 6- Selecione a **porta** em que o Arduino está conectado
- 7- Carregue o código no Arduino 
- 8- Abrir o Monitor Serial  para ver o resultado

# Buzzer

// Detecta a batida no piezo e mostra a palavra no monitor serial

- 1- Abra o **Arduino IDE**
- 2- Conecte o cabo **USB** do Arduino no computador
- 3- Digite o código abaixo:

Arquivo Editar Sketch Ferramentas Ajuda




✓ ↻ 📄 ⬆ ⬇ ⬇  
sketch\_piezo

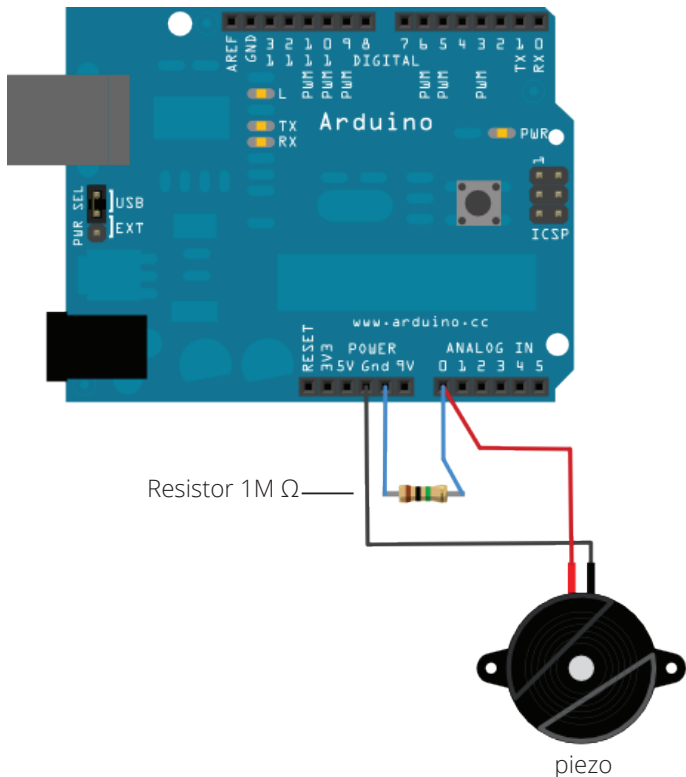
```
int led = 13;
int SensorToc = A0;
int entrada = 100;

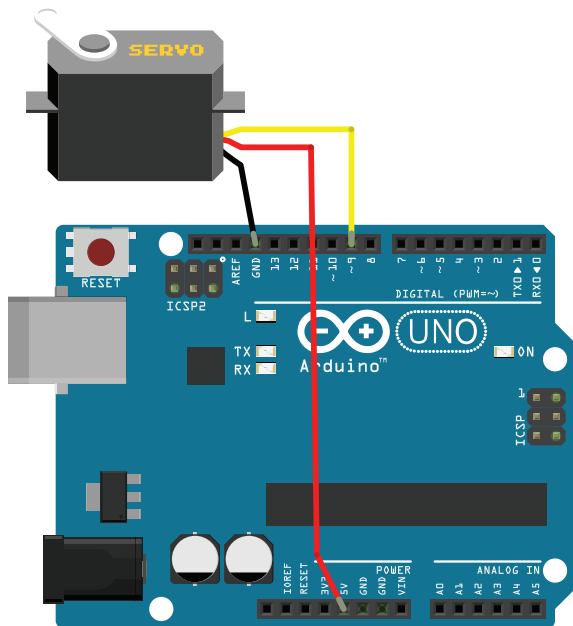
int estadoSensor = 0;
int estadoLed = LOW;

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  estadoSensor = analogRead(SensorToc);
  if (estadoSensor >= entrada) {
    estadoLed = !estadoLed;
    digitalWrite(led, estadoLed);
    Serial.println("TocToc");
  }
  delay(100);
}
```

- 4- Clique em  para checar se o código foi escrito corretamente
- 5- Selecione a sua versão da **placa** Arduino em Ferramentas
- 6- Selecione a **porta** em que o Arduino está conectado
- 7- Carregue o código no Arduino 
- 8- Abrir o Monitor Serial  para ver o resultado





# Servo Motor

// Cria um objeto chamado servomotor

- 1- Abra o **Arduino IDE**
- 2- Conecte o cabo **USB** do Arduino no computador
- 3- Digite o código abaixo:

Arquivo Editar Sketch Ferramentas Ajuda

✓ ↻ 📄 ⬇️ ⬆️  
sketch\_servo

```
int potpin = 0;  
int int val;
```

```
void setup() {  
  servomotor.attach(9);  
}
```

```
void loop() {  
  val = analogRead(potpin);  
  val = map(val, 0, 1023, 0, 180);  
  servomotor.write(val);  
  delay(15);  
}
```

- 4- Clique em ✓ para checar se o código foi escrito corretamente
- 5- Selecione a sua versão da **placa** Arduino em Ferramentas
- 6- Selecione a **porta** em que o Arduino está conectado
- 7- Carregue o código no Arduino ➡️
- 8- Abrir o Monitor Serial 🖨️ para ver o resultado

# Tabela de resistores

Arduino é uma placa que consegue ler informações de botões e sensores de vários tipos como movimento, som, luminosidade, umidade e, através de um processamento, transforma em saídas para controlar luzes, motores e outros atuadores. Esse processamento é feito através de um código que você pode criar para controlar seu projeto!



## 1º Dígito

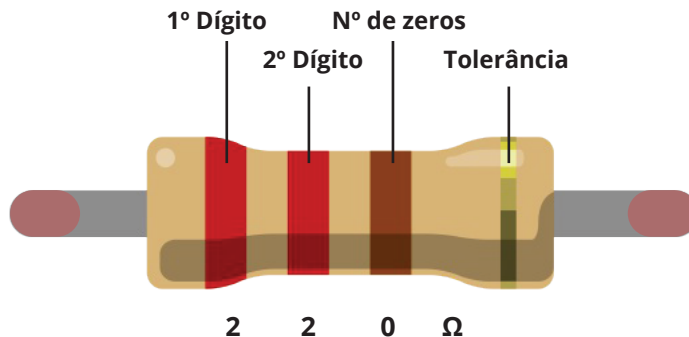
Compare a cor com os números correspondentes

## 2º Dígito

Faça a mesma coisa que com o 1º dígito

## Nº de zeros

Pegue o número correspondente a cor e coloque essa quantidade de zeros



## Da cor para o valor

Para descobrir o valor do **resistor** basta encontrar os números correspondentes a cada cor na ordem indicada.

## Do valor para a cor

Para descobrir a cor do resistor é só encontrar trocar os números pelas cores correspondentes na ordem indicada.



## Tolerância

Quando não tiver **nenhuma** faixa, significa que a tolerância é de **20%**. Quando for **prateada** = **10%** e **dourada** = **5%**

## Fórmula: Lei de Ohm

Com essa fórmula você descobre qual o resistor (**R = Resistência**) precisa para seu experimento. Basta dividir o valor da voltagem (**V = Voltagem**) pelo valor da corrente (**Ω = Amperes**)

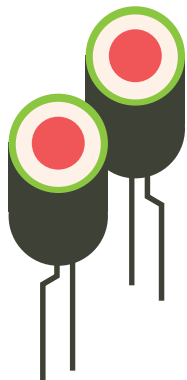
$$R = V / \Omega$$

$$V = \Omega \times R$$

$$\Omega = V / R$$

# CERTIFICADO DE PARTICIPAÇÃO DO

## DESAFIO DOS Sushi Cards 2018



---

Olido Maker Club

---

Participante

