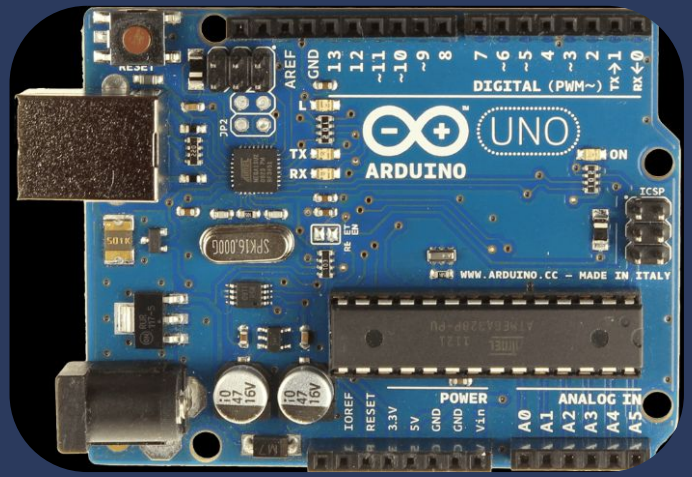
Oficina de introdução à robótica

Segunda Aula

Instrutores: Pedro Rocha e João Pedro Arruda
Organização: Profa. Regiane Kawasaki e Prof. Victor Santiago

> Revisão Aula Passada

>>>>>>>>>>>>>>>>>>>>>>>>>>>> Aula Passada



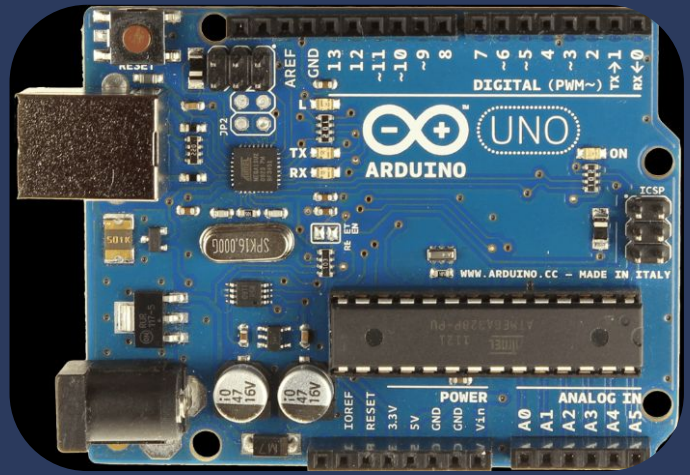
Entradas e Saídas Digitais:

- O Arduino Uno possui 14 pinos digitais que podem ser usados como entradas ou saídas digitais. Esses pinos podem ser configurados para ler sinais digitais (0 ou 1) ou enviar sinais digitais (ligado ou desligado).

Entradas Analógicas:

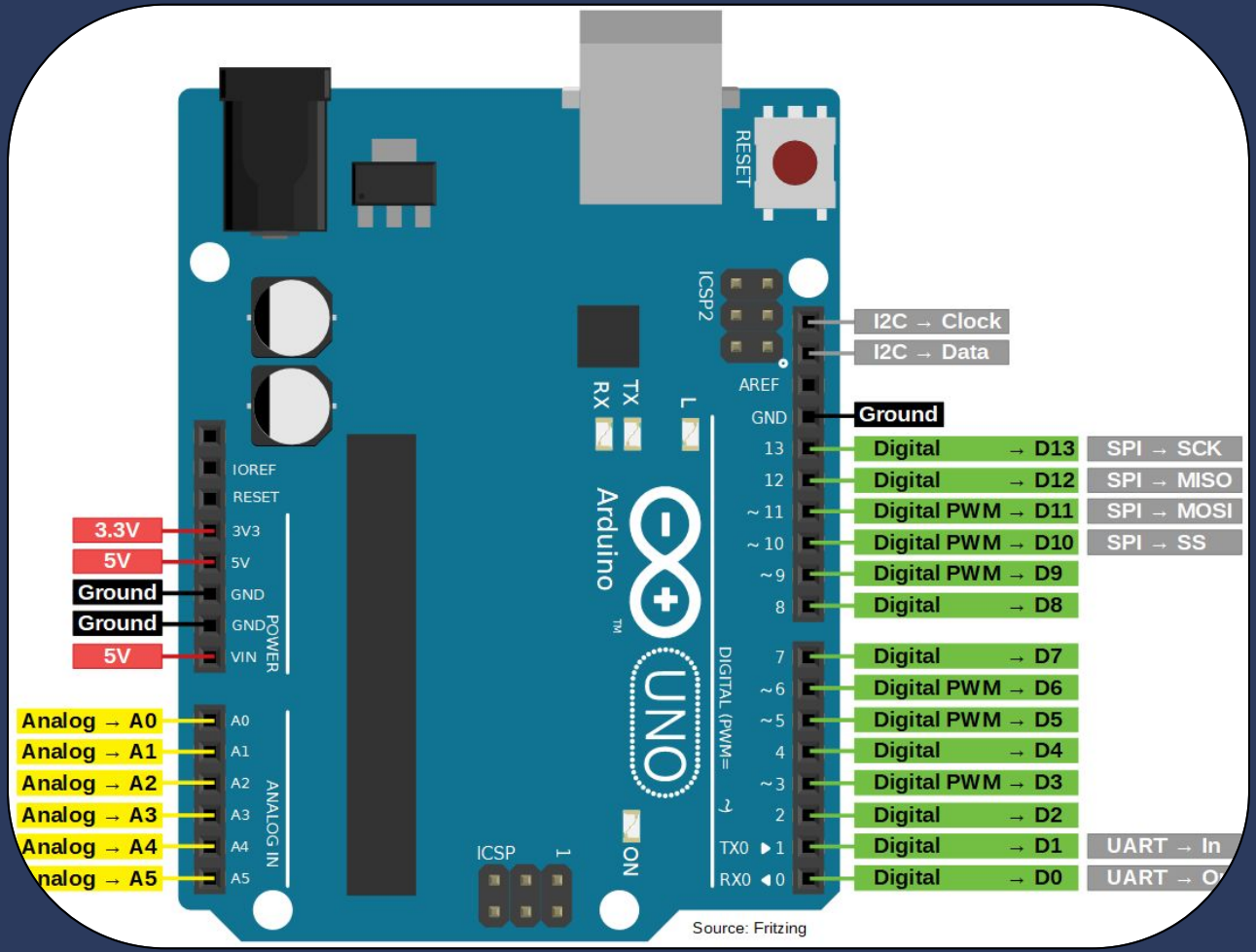
- Além das entradas digitais, o Arduino Uno também possui 6 pinos de entrada analógica, que podem ser usados para ler valores analógicos, como sensores de luz, temperatura e potenciômetros.

>>>>>>>>>>>>>>>>>>>>>>>>>>>> Aula Passada



Programação:

- O Arduino Uno é programado usando a linguagem de programação Arduino, que é baseada em C/C++. Os desenvolvedores podem escrever seu código no ambiente de desenvolvimento Arduino, que inclui uma IDE (Integrated Development Environment) fácil de usar.


RESET
ICSP2
I2C → Clock
I2C → Data
AREF
GND
Ground
Digital → D13
SPI → SCK
Digital → D12
SPI → MISO
Digital PWM → D11
SPI → MOSI
Digital PWM → D10
SPI → SS
Digital PWM → D9
Digital → D8
Digital → D7
Digital PWM → D6
Digital PWM → D5
Digital → D4
Digital PWM → D3
Digital → D2
Digital → D1
UART → In
Digital → D0
UART → O
DIGITAL (PWM=~)
TX
RX
L
ON
Arduino
UNO
IOREF
RESET
3.3V
5V
Ground
Ground
5V
POWER
Analog → A0
Analog → A1
Analog → A2
Analog → A3
Analog → A4
Analog → A5
ANALOG IN
ICSP
Source: Fritzing

> Programação no Arduino

Linguagem Arduino >>>>>>>>>>>>>>>>>>>>>>>

Pinagem:
Para definir as portas, basta declarar o tipo de variável em seguida o nome e em seguida atribuir a variável ao pino que será usado

Linguagem Arduino >>>>>>>>>>>>>>>>>>>>>>>>

Configuração dos pinos (pin mode):
Para começar a usar um pino específico, você precisa configurá-lo como entrada (input) ou saída (output) usando a função `pinMode()`

```
1  pinMode(pare, OUTPUT);
2  pinMode(atencao, OUTPUT);
3  pinMode(siga, OUTPUT);
```

Linguagem Arduino >>>>>>>>>>>>>>>>>>>>>>>>

Estrutura Básica:
Os programas Arduino têm uma estrutura básica composta por duas funções principais: `setup()` e `loop()`.

Setup (Configurar):
Esta função é executada uma única vez, assim que o Arduino é ligado ou reiniciado. É onde você configura inicializações e configurações iniciais para o seu projeto.

Loop (Laço):
A função `loop()` é o coração do programa Arduino. Ela é executada continuamente após a execução da função `setup()`. Qualquer código dentro da função `loop()` será repetido infinitamente, a menos que o Arduino seja reiniciado ou desligado.

```
void setup() {
    pinMode(pare, OUTPUT);
    pinMode(atencao, OUTPUT);
    pinMode(siga, OUTPUT);
    pinMode(pedSiga, OUTPUT);
    pinMode(pedPare, OUTPUT);
}
```

```
void loop() {
    sinalAmarelo();
    delay(5000);
    sinalVermelho();
    delay(6000);
    sinalVerde();
    delay(6000);
}
```

Funções Void >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Uma função void é uma função que não retorna nenhum valor. Isso significa que você pode usá-la para realizar ações ou tarefas sem a necessidade de retornar um resultado específico

Definição da função:
Para definir a função, é necessário declarar void, escrever o nome da função, abrir um campo vazio de parâmetros e abrir o escopo com chaves. O escopo da função contém o código que será executado quando a função for chamada.

Chamada de Função:
Para chamar uma função basta declará-la dentro do loop


1 void sinalVermelho() {
2 digitalWrite(pare, HIGH);
3 digitalWrite(atencao, LOW);
4 digitalWrite(siga, LOW);
5 }

If, else if e else >>>>>>>>>>>>>>>>>>>>>>>>>>>>

If (se):
A estrutura if permite que você execute um bloco de código somente se uma determinada condição for verdadeira (ou seja, avaliada como verdadeira). Para realizar essa condição, é necessário chamar a condição if, abrir parênteses, colocar as condições e abrir o escopo da função, suas variações são: `else if` e `else`

```
if (ledOrder == 1) {
  sinalVerde();
  pedestrePare();
  delay(5000);
}
else if (ledOrder == 2) {
  sinalAmarelo();
  delay(2500);
}
else {
  sinalVermelho();
  pedestreSegue();;
  delay(5000);

  ledOrder = 0;
}
```

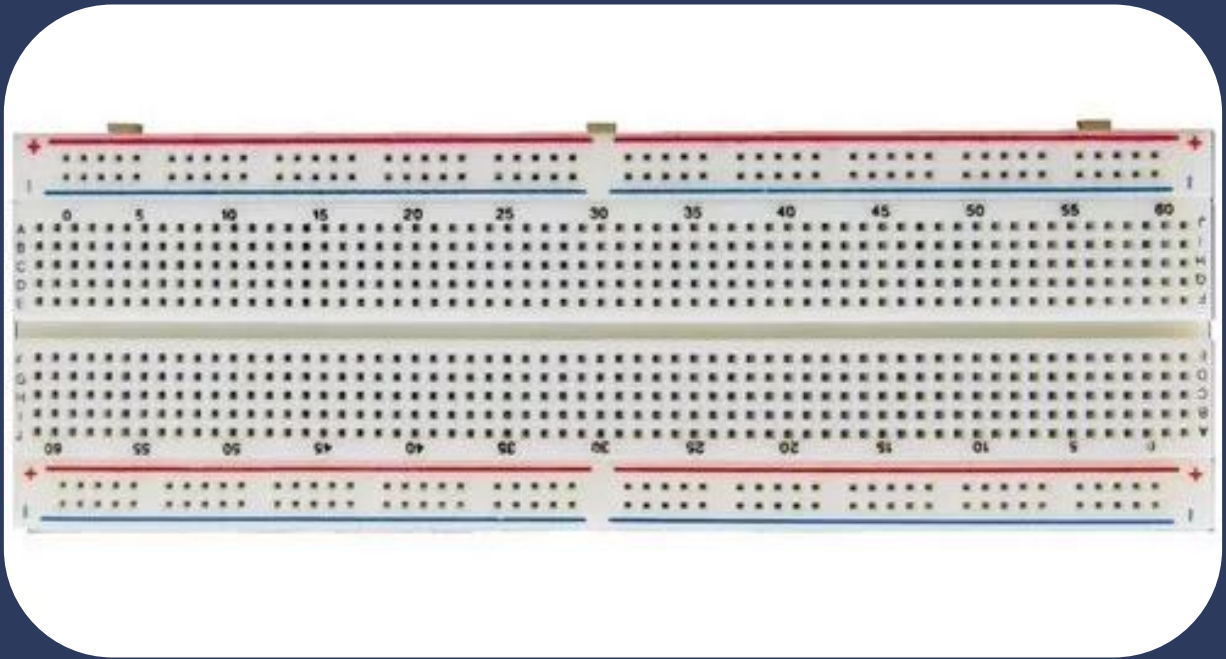
> Componentes

Componentes >>>>>>>>>>>>>>>>>>>>>>>>>>>>

Protoboard:
Ela possui furos interconectados, permitindo que componentes eletrônicos, como resistores e LEDs, sejam inseridos e conectados sem a necessidade de solda.

LED:
É um componente eletrônico semicondutor que emite luz quando uma corrente elétrica passa por ele. Ele é usado para indicar estados, fornecer iluminação em displays e dispositivos eletrônicos.

Jumpers:
São fios que permitem a passagem de corrente e a propagação do sinal entre a placa e a protoboard


0 5 10 15 20 25 30 35 40 45 50 55 60
A B C D E F G H I J

> Desafio Passado

Desafio Passado >>>>>>>>>>>>>>>>>>>>>>>>>

1. Acenda um LED
2. Semáforo
3. Semáforo com semáforo de pedestres
4. Pisque o LED a cada 3 segundos

/Segunda Aula

/Segunda Aula

Tópicos Principais:

01. Resistores
02. Entradas PWM
03. Buzzer
04. Botões (Push Button)
05. Sensores de Linha

/Resistores

/Resistores

- Componentes elétricos utilizados para limitar ou controlar a corrente elétrica em um circuito, convertendo energia elétrica em energia térmica, dissipando parte da energia em forma de calor.

- Os resistores são identificados por seu valor de resistência, medido em ohms (Ω).

- São usados em uma variedade de aplicações eletrônicas, como divisores de tensão, limitadores de corrente, filtros de sinal e para ajustar o brilho em LEDs.

Cor	1ª Faixa	2ª Faixa	Faixa multiplicadora	Tolerância
Preto	0	0	×1Ω	-
Marrom	1	1	×10Ω	±1%
Vermelho	2	2	×100Ω	±2%
Laranja	3	3	×1kΩ	-
Amarelo	4	4	×10kΩ	-
Verde	5	5	×100kΩ	±0,5%
Azul	6	6	×1MΩ	±0,25%
Violeta	7	7	×10MΩ	±0,1%
Cinza	8	8	-	±0,05%
Branco	9	9	-	-
Dourado	-	-	×0,10Ω	±5%
Prateado	-	-	×0,01Ω	±10%
Sem cor	-	-	-	±20%

/Sinal PWM

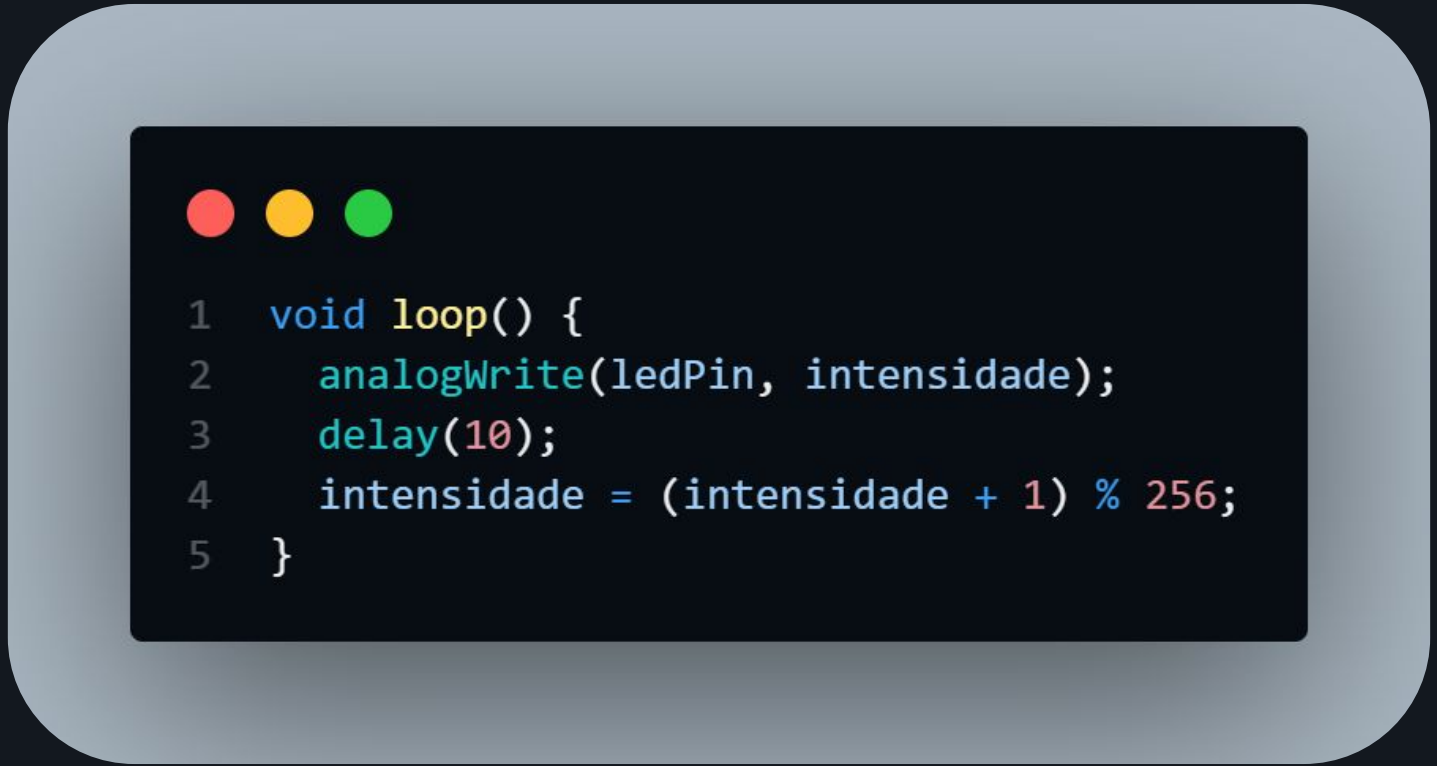
/Sinal PWM

O que é:
A sigla PWM significa **"Pulse Width Modulation"** em inglês, que pode ser traduzida como "Modulação por Largura de Pulso". No contexto da eletrônica em geral, o PWM é uma técnica usada para controlar a intensidade de um sinal elétrico, como uma tensão ou corrente, variando a largura dos pulsos em um sinal digital.

Como identificar:
No Arduino, você pode usar saídas PWM em alguns dos pinos digitais para gerar sinais PWM. O Arduino Uno tem vários pinos PWM, como o 3, 5, 6, 9, 10 e 11. Esses pinos são marcados com o símbolo **"~"** para indicar que eles são capazes de gerar sinais PWM.

/Sinal PWM

Como programar:
Para “ativar” o sinal PWM no arduino é necessário usar a função `analogWrite()` no código, o qual recebe dois argumentos, que são, a porta que será utilizada pelo componente e em seguida o ciclo de trabalho que será utilizado pela porta. As portas PWM aceitam ciclos de trabalho de 0 até 256. 0 sendo o desempenho mínimo do componente de 256 sendo o desempenho máximo.É importante notar que, apesar do nome, a saída de analogWrite() é digital e não produz uma saída analógica real.


void loop() {
analogWrite(ledPin, intensidade);
delay(10);
intensidade = (intensidade + 1) % 256;
}

/Buzzer

/Buzzer

O que é:
Buzzer é um dispositivo utilizado em eletrônica para produzir som ou emitir um sinal sonoro. Existem dois tipos de Buzzers, ativos e passivos. Usaremos os buzzers passivos, os quais não vêm com oscilador interno e necessitam de uma entrada PWM.

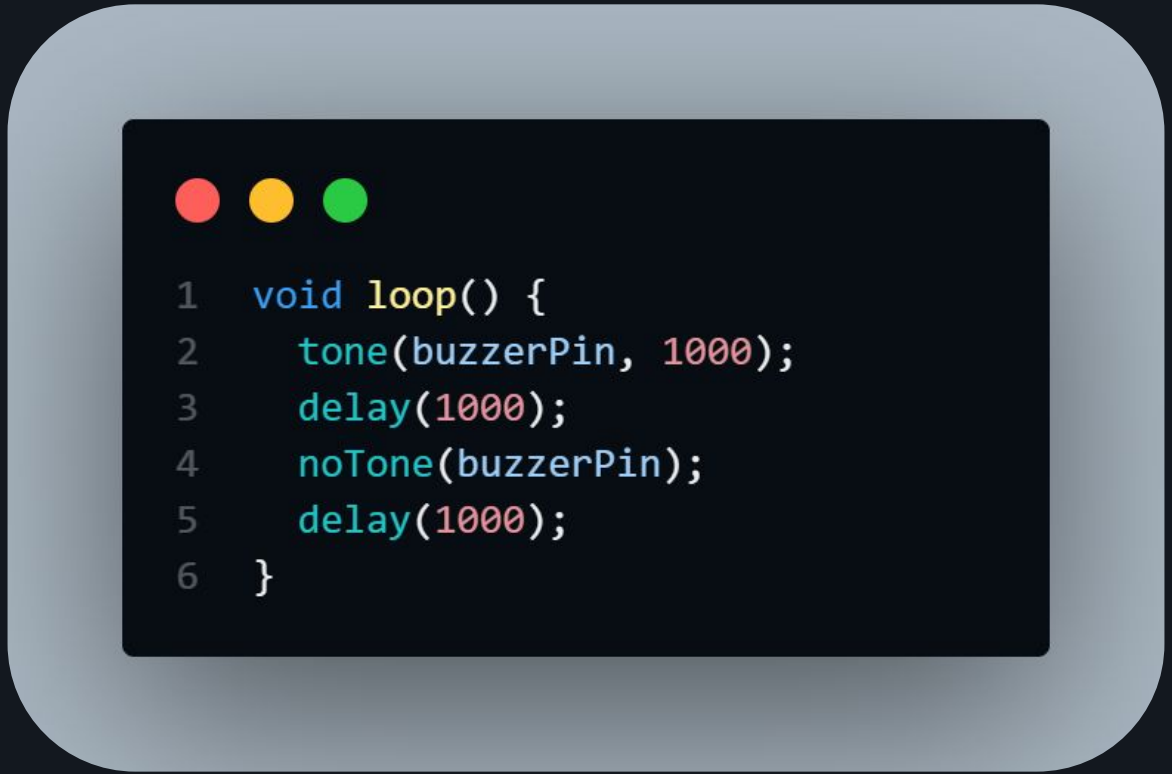
Como usar:
Para usar um buzzer basta atribuí-lo à uma porta PWM (as que possuem sinal “~”), juntamente com um sinal GND para o aterramento .

/Buzzer

Programação:
Diferentemente dos LEDs, que possuem maneiras diferentes de funcionamento diferentes dependendo da porta que é colocado, os buzzers que usaremos não precisam fazer o uso da função `analogWrite()`, pois só funcionam com sinais PWM, e possui duas funções principais.

Tone:
É a função responsável por atribuir o pin que se contra o buzzer e a frequência do som que ele irá emitir.

No Tone:
Essa função recebe somente um argumento que é o pin do buzzer e é responsável por parar o som que ele está emitindo .


void loop() {
tone(buzzerPin, 1000);
delay(1000);
noTone(buzzerPin);
delay(1000);
}

/Hora do Desafio #1

/Hora do Desafio

1. Toquem uma sequência de sons no buzzer
2. Acenda um led e altere sua intensidade
3. Faça uma sirene usando o buzzer

/Botão(Push Button)

/Botão (Push Button)

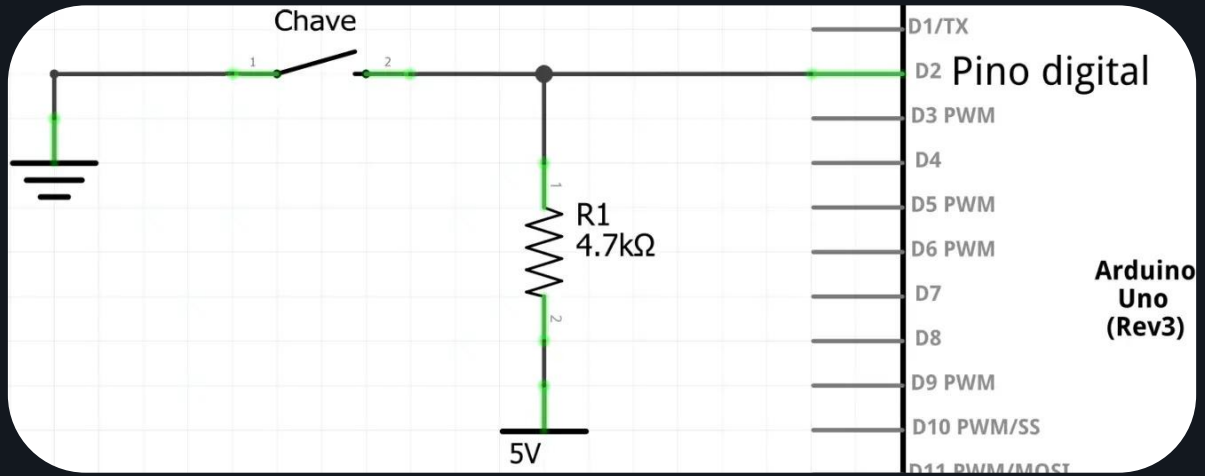
O que é:

- É um dos componentes eletrônicos mais utilizados para prototipagem de projetos.
- Esta chave é um tipo de interruptor pulsador (conduz somente quando está pressionado).

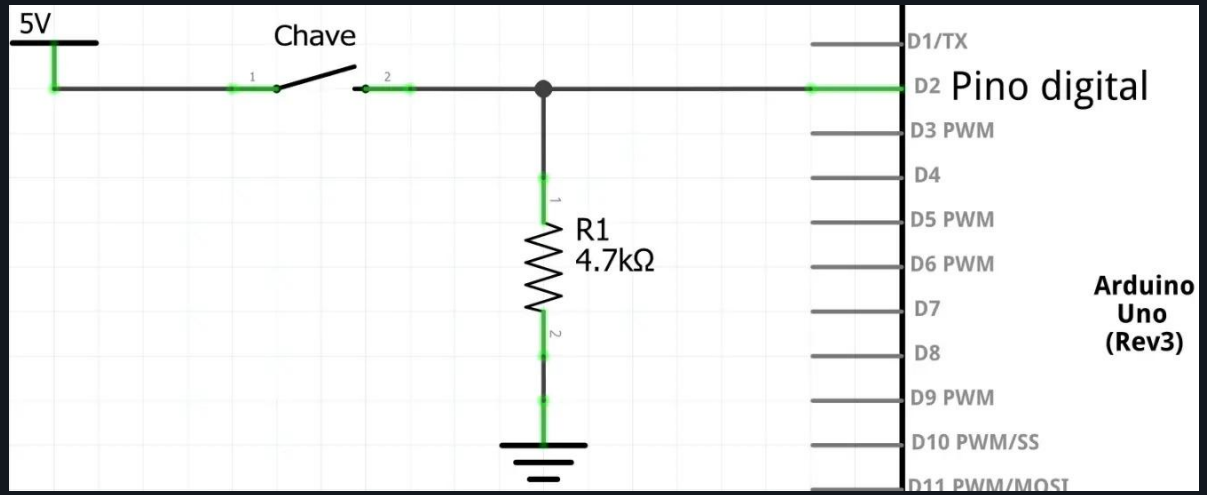
Pull UP e Pull DOWN:

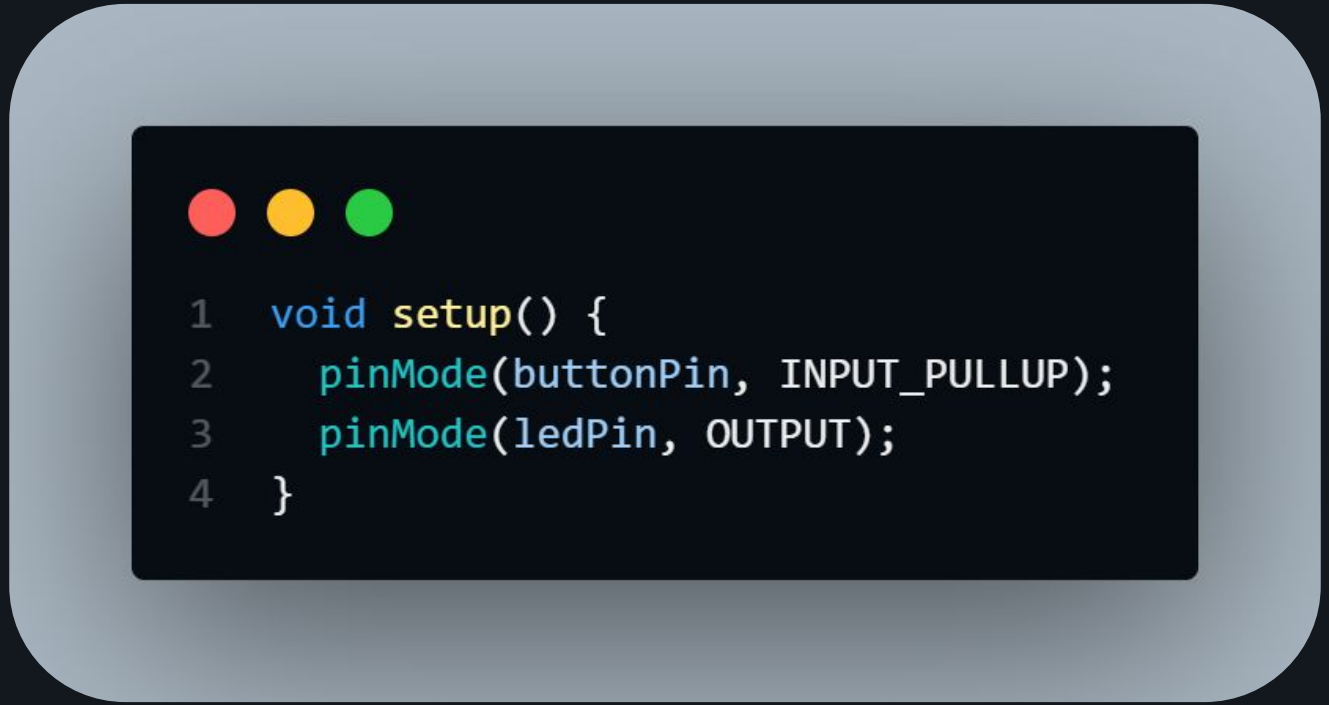
- São usados em eletrônica para garantir que um circuito funcione corretamente e evite problemas indesejados.
- Ajudam a eliminar “ruídos elétricos” que podem causar leituras erradas no sistema.

O resistor Pull DOWN permite que, quando o botão não está pressionado, o pino digital receberá corretamente o valor **HIGH**.



O resistor Pull DOWN permite que, quando o botão não está pressionado, o pino digital receberá corretamente o valor **LOW**.




1 void setup() {
2 pinMode(buttonPin, INPUT_PULLUP);
3 pinMode(ledPin, OUTPUT);
4 }

/Hora do Desafio #2

/Hora do Desafio

1. Façam o led acender com o botão
2. Toque um som quando apertar o botão
3. Use 3 botões e LEDs. Para cada conjunto, toque um som diferente quando pressionar o botão

/Sensores de Linha

/Sensor de Linha

O que é:
sensor de linha é um dispositivo que é usado para detectar e seguir linhas em superfícies, como linhas desenhadas no chão ou faixas em robôs seguidores de linha.

Funcionamento:
Os sensores de linha são baseados na emissão de luz infravermelha e um fototransistor ou fotodiodo que detecta a luz refletida. Quando o sensor está sobre uma superfície, a luz infravermelha é refletida de volta para o fototransistor ou fotodiodo. O valor da reflexão **depende do contraste entre a linha e a superfície** circundante.

/Sensor de Linha

Serial Begin:
Para programar o sensor, é necessário usar a função `Serial.begin()` a qual é uma função em Arduino que é usada para inicializar a comunicação serial entre o Arduino e um dispositivo externo. Essa função aceita um argumento que especifica a taxa de transmissão (baud rate) na qual os dados serão enviados e recebidos pela porta serial. O baud rate determina a velocidade com que os dados são transmitidos e recebidos, medido em bits por segundo (bps).


1 void setup() {
2 pinMode(sensorPin, INPUT);
3 pinMode(ledPin, OUTPUT);
4 Serial.begin(9600);
5 }

/Sensor de Linha

Digital Read:
Outra função que será usada é chamada de `digitalRead()`, que é uma função em Arduino que é usada para ler o estado digital de um pino específico. Ela retorna o valor lido no pino, que pode ser HIGH (1) ou LOW (0), dependendo do nível de tensão presente no pino no momento da leitura.

```
void loop() {
  int sensorValue = digitalRead(sensorPin); // Lê o estado do sensor (HIGH ou LOW)
  if (sensorValue == HIGH) {
    Serial.println("Linha detectada");
  } else {
    Serial.println("Linha não detectada");
  }
  delay(100);
}
```