

**INSTITUTO
FEDERAL**
Santa Catarina

Câmpus
Chapecó



Arduino/Projeto AHA(2023)

O que é Arduino ?

- O Arduino foi criado em 2005 por um grupo de 5 pesquisadores : Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. O objetivo era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores.
- <https://www.filipeflop.com/blog/o-que-e-arduino/>

Arduino é uma ferramenta de código aberto usado na construção de projetos eletrônicos(*protótipos*).

Arduino é composto por uma placa física programável, um circuito e um ambiente de desenvolvimento, ou IDE, que é executado em seu computador, é utilizado para escrever(línguagem c) e fazer upload de código do computador para a placa.

Depois de programado, o microcontrolador pode ser usado de forma independente, ou seja, você pode colocá-lo para controlar um robô, uma lixeira, um ventilador, as luzes da sua casa, a temperatura do ar condicionado, pode utilizá-lo como um aparelho de medição ou qualquer outro projeto que vier à cabeça.



Como programar

- Para programar o Arduino é necessário realizar o Download do aplicativo no link abaixo:

<https://downloads.arduino.cc/arduino-1.8.19-windows.exe>



Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

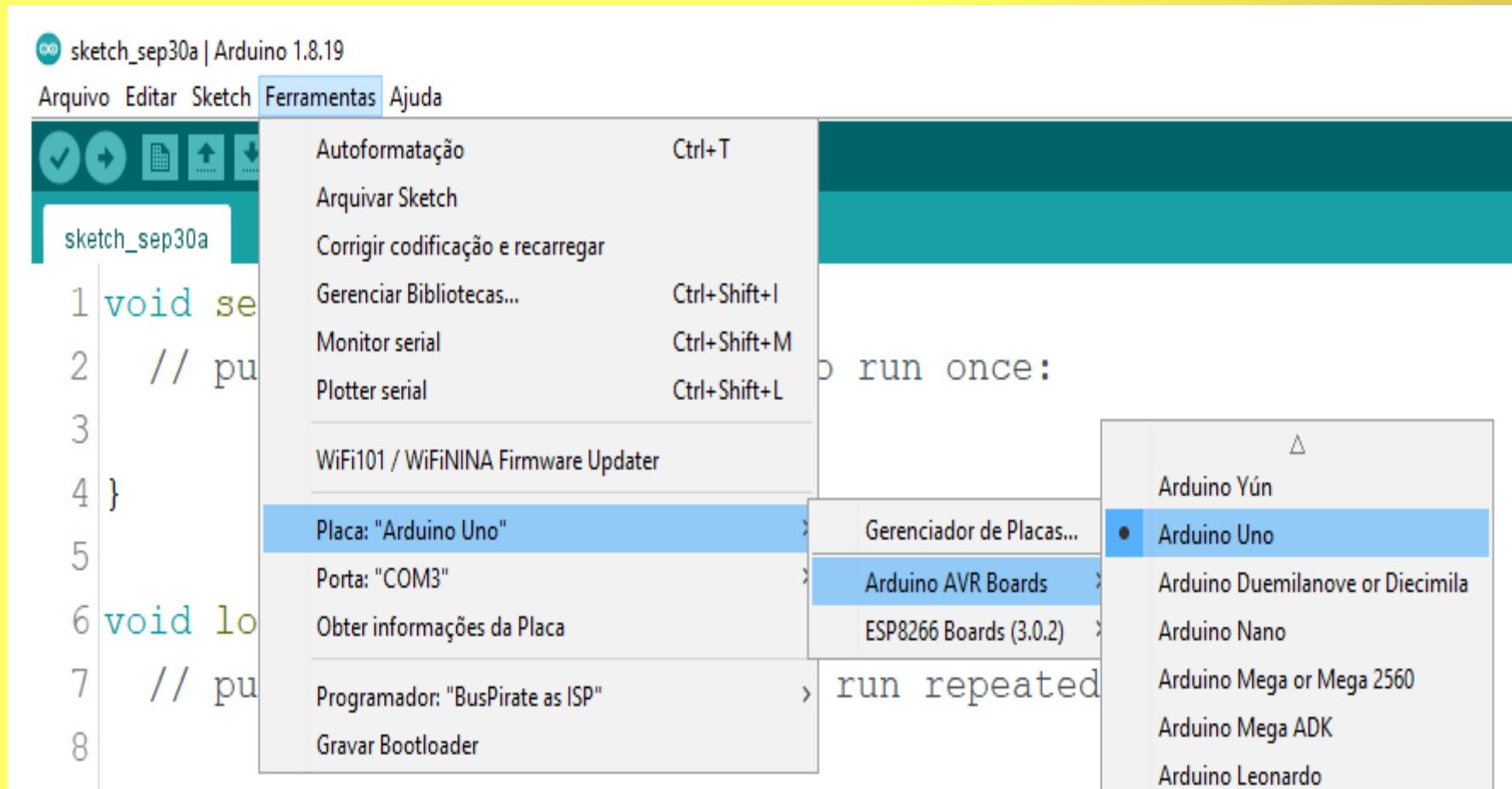
Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 [Get](#)

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Mac OS X 10.10 or newer

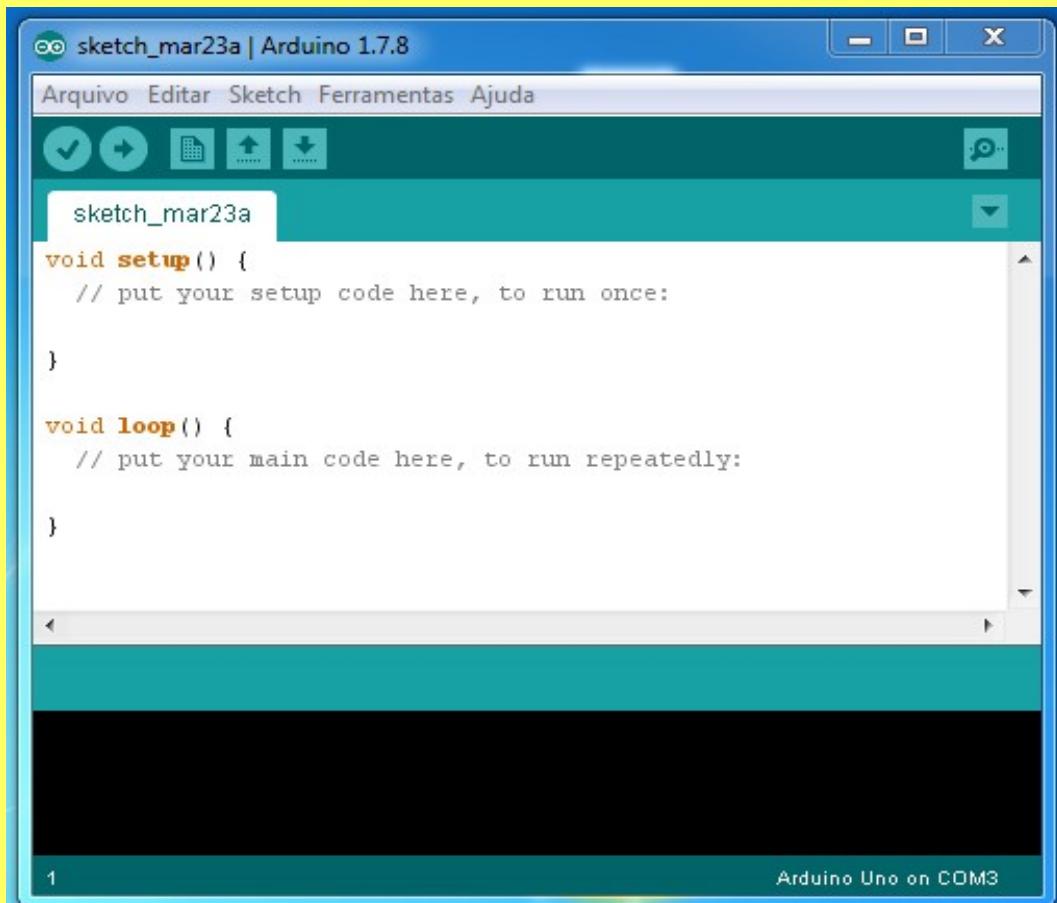
[Release Notes](#)
[Checksums \(sha512\)](#)

Configuração do modelo e da porta de comunicação



Primeiro de tudo é importante explicar que a linguagem de programação (c) Arduino é sensível a maiúsculas: uma letra maiúscula não é o mesmo que uma letra minúscula.
O código a seguir representa o mínimo para que um programa possa ser compilado:

O "void setup ()" é normalmente usado para inicializar variáveis , modos de pinos , definição a velocidade de transmissão serial, etc. O software só vai executar uma vez.



```
sketch_mar23a | Arduino 1.7.8
Arquivo Editar Sketch Ferramentas Ajuda
sketch_mar23a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

1 Arduino Uno on COM3

O "void loop ()" é a parte do código em que se faz um loop, ou seja, cria-se um ciclo de repetição para que as instruções dentro do mesmo sejam repetidas.

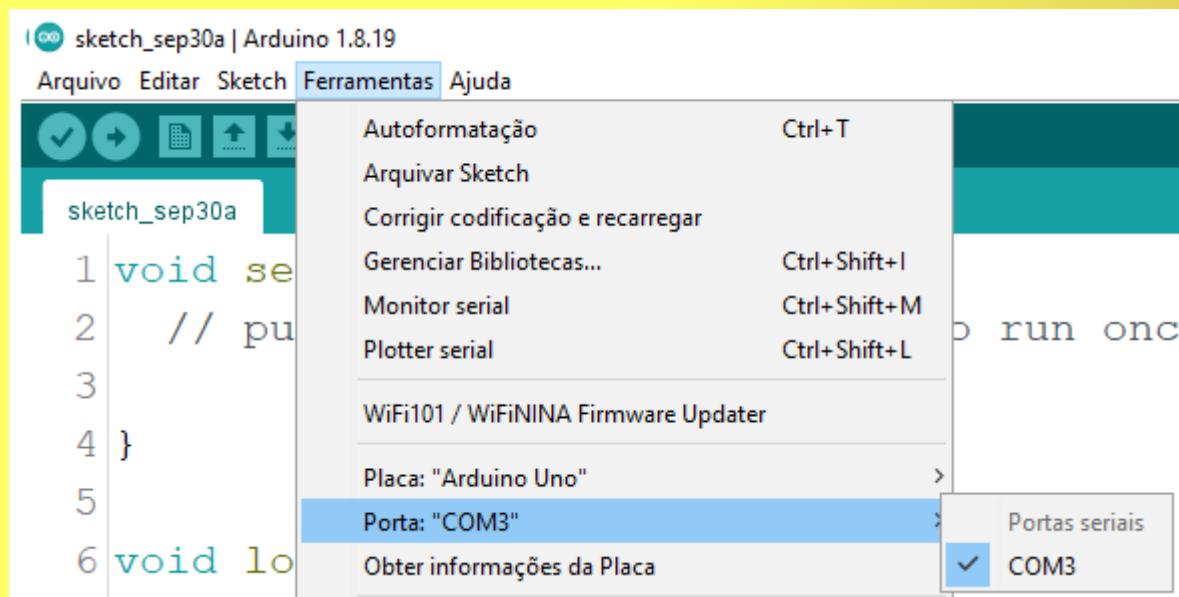
O que você pode fazer com o Arduino ?

A lista de possibilidades é praticamente infinita. Você pode automatizar sua casa, seu carro, seu escritório, criar um novo brinquedo, um novo equipamento ou melhorar um já existente. Tudo vai depender da sua criatividade.

Para isso, o Arduino possui uma quantidade enorme de *sensores* e *componentes* que você pode utilizar nos seus projetos. Grande parte do material utilizado está disponível em módulos, que são pequenas placas que contém os sensores e outros componentes auxiliares como resistores, capacitores e leds.

A porta de comunicação geralmente é maior que COM1 e COM2. Caso não apareça alguma porta maior é necessário executar um aplicativo de comunicação(driver chip) através do seguinte link:

<https://arduino.migueldebarba.com.br/CH341SER/>

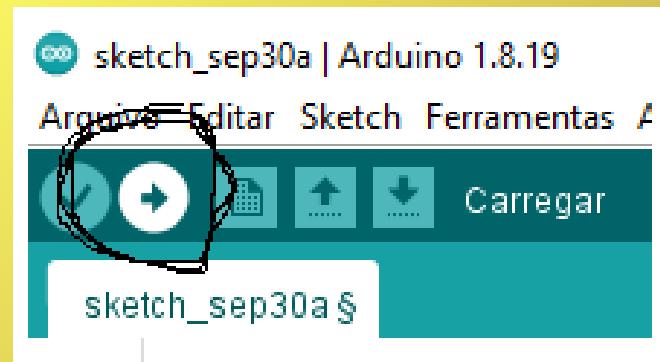


Primeiro projeto

Vamos ligar o Led existente na placa do Arduino:

```
1 void setup() {  
2     pinMode(LED_BUILTIN, OUTPUT); // Configura o Led para saída  
3 }  
4  
5 void loop() {  
6     digitalWrite(LED_BUILTIN, HIGH); // Liga o Led  
7 }
```

Em seguida carregue o código para a placa realizar a atividade programada:



Primeiro projeto

Vamos desligar o Led existente na placa do Arduino:

```
1 void setup() {  
2     pinMode(LED_BUILTIN, OUTPUT); // Configura o Led para saída  
3 }  
4  
5 void loop() {  
6     digitalWrite(LED_BUILTIN, LOW); // Desliga o Led  
7     delay(1000);  
8 }
```

E para deixar o Led piscando?

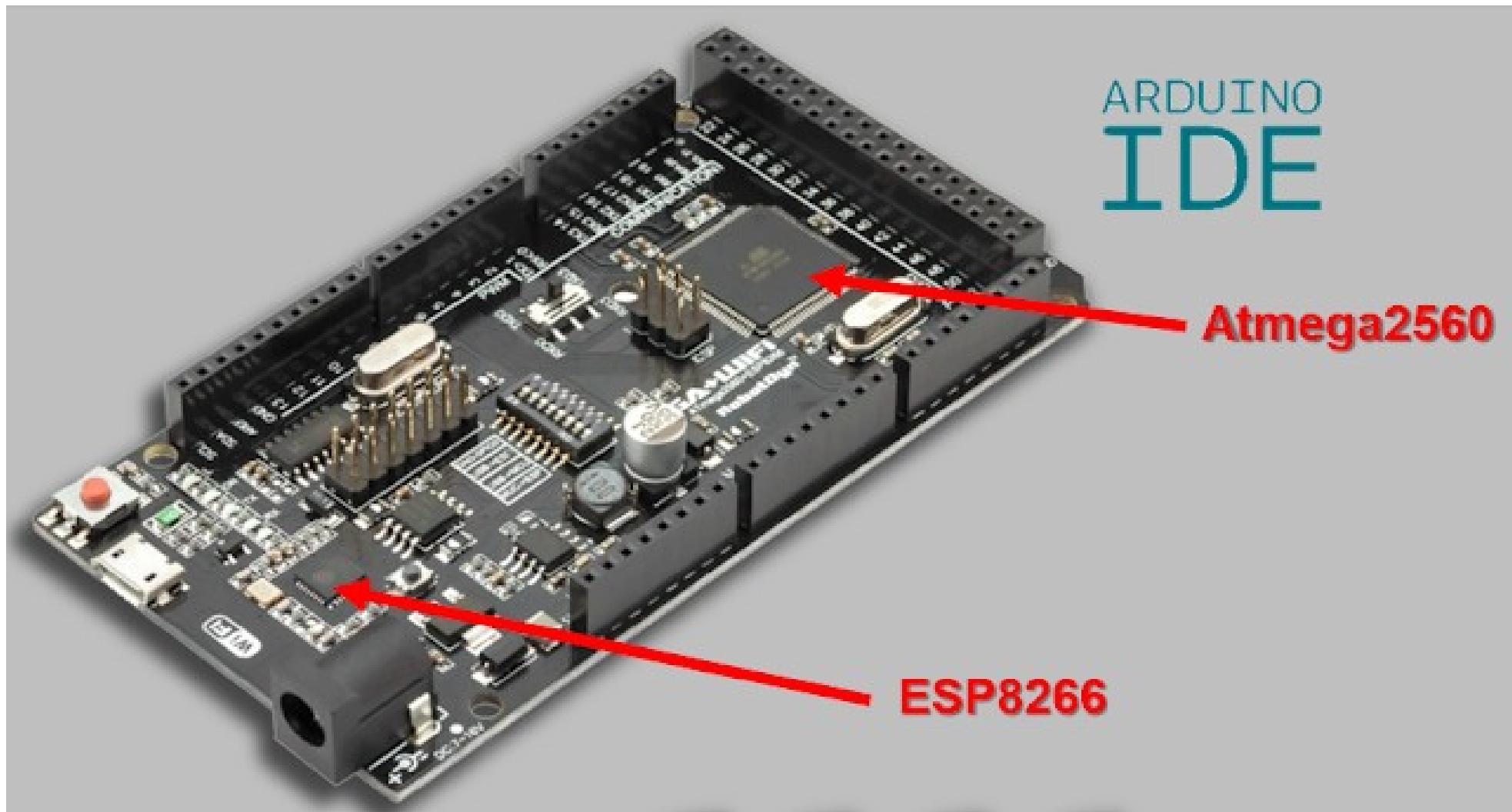
```
1 void setup() {  
2     pinMode(LED_BUILTIN, OUTPUT); // Configura o Led para saída  
3 }  
4  
5 void loop() {  
6     digitalWrite(LED_BUILTIN, HIGH); // Liga o Led  
7     delay(1000);  
8     digitalWrite(LED_BUILTIN, LOW); // Desliga o Led  
9     delay(1000);  
10 }
```

Placas Arduino

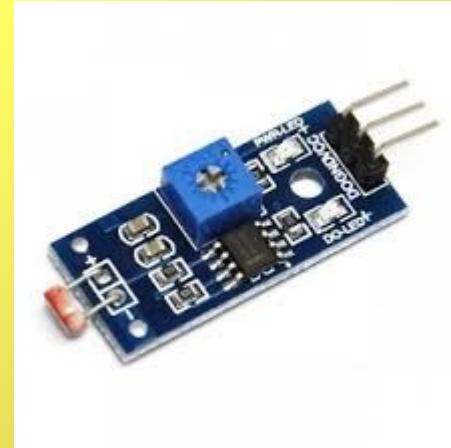
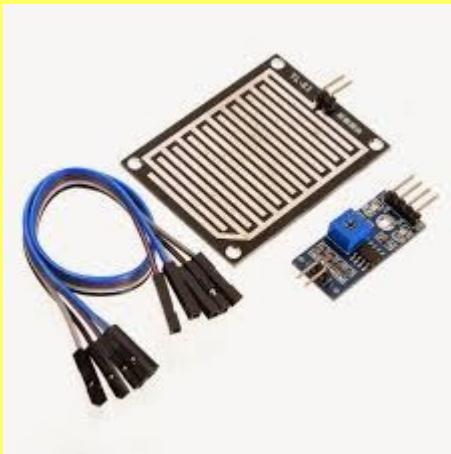
Estima-se que existam mais de quinze modelos/configurações de placas Arduino. Estas variações possuem diferenças de tamanho, memória, velocidade, conectividade e principalmente consumo de corrente elétrica, o que faz toda diferença para projetos complexos.



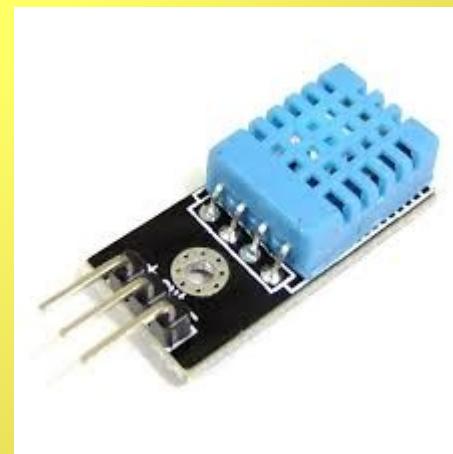
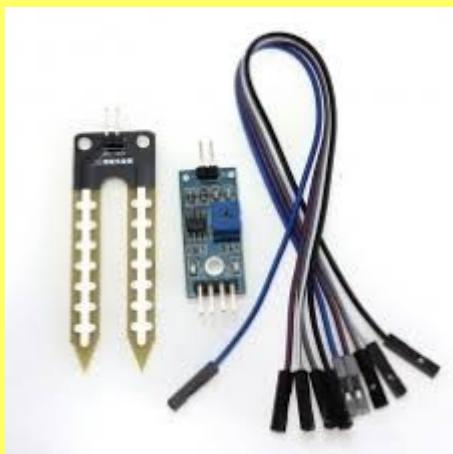
Arduino Mega com ESP8266



Sensores



Sensores



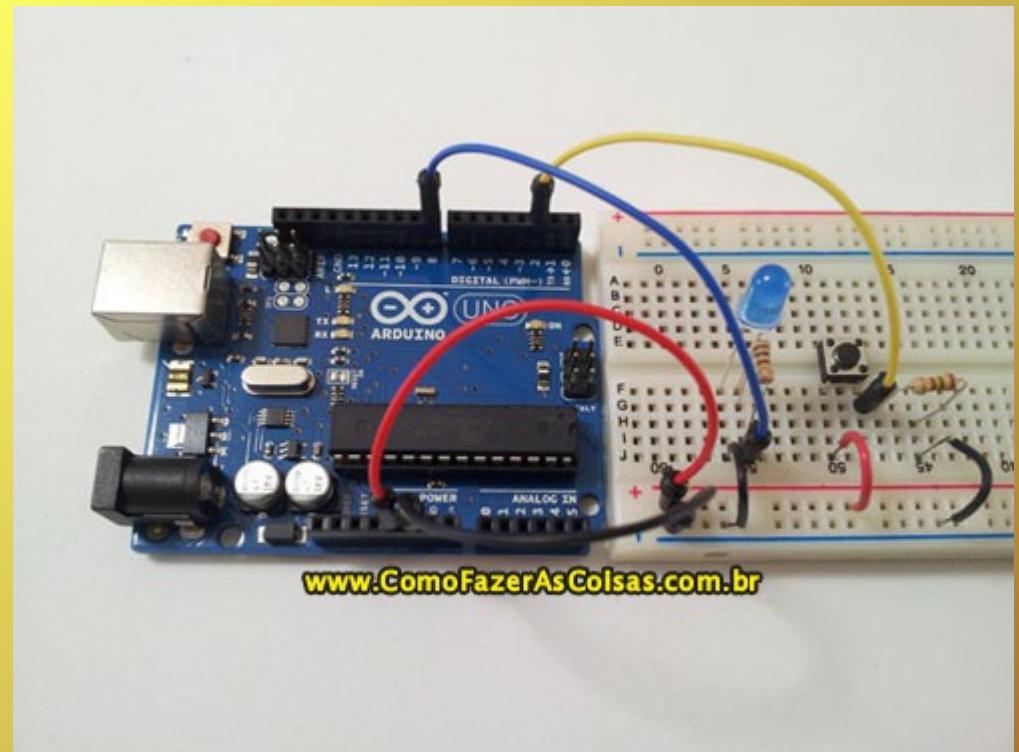
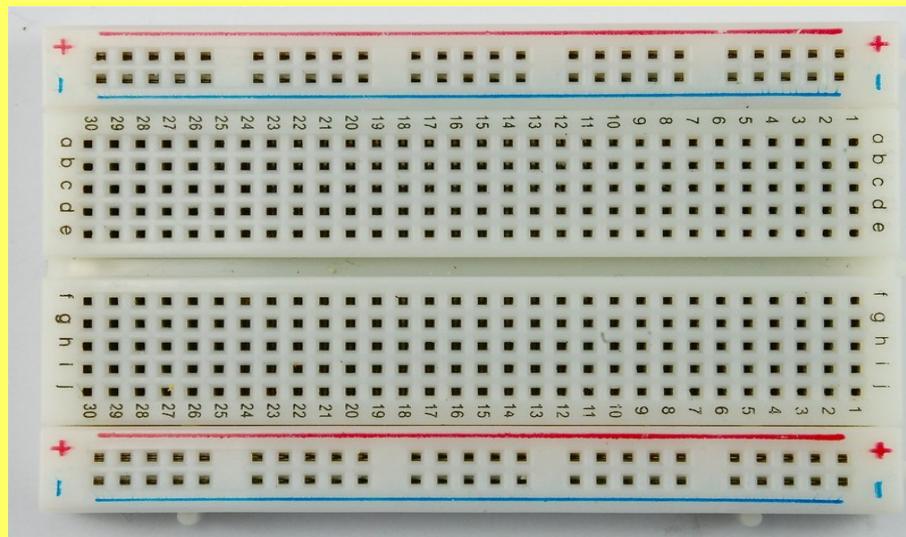
Arduino Shield

- Existem também os chamados Shields, que são placas que você encaixa no Arduino para expandir suas funcionalidades. A imagem a seguir mostra um *Arduino Ethernet Shield* encaixado no Arduino Mega 2560. Ao mesmo tempo que permite o acesso à uma rede ou até mesmo à internet, mantém os demais pinos disponíveis para utilização, assim você consegue, por exemplo, utilizar os pinos para receber dados de temperatura e umidade de um ambiente, e consultar esses dados de qualquer lugar do planeta:

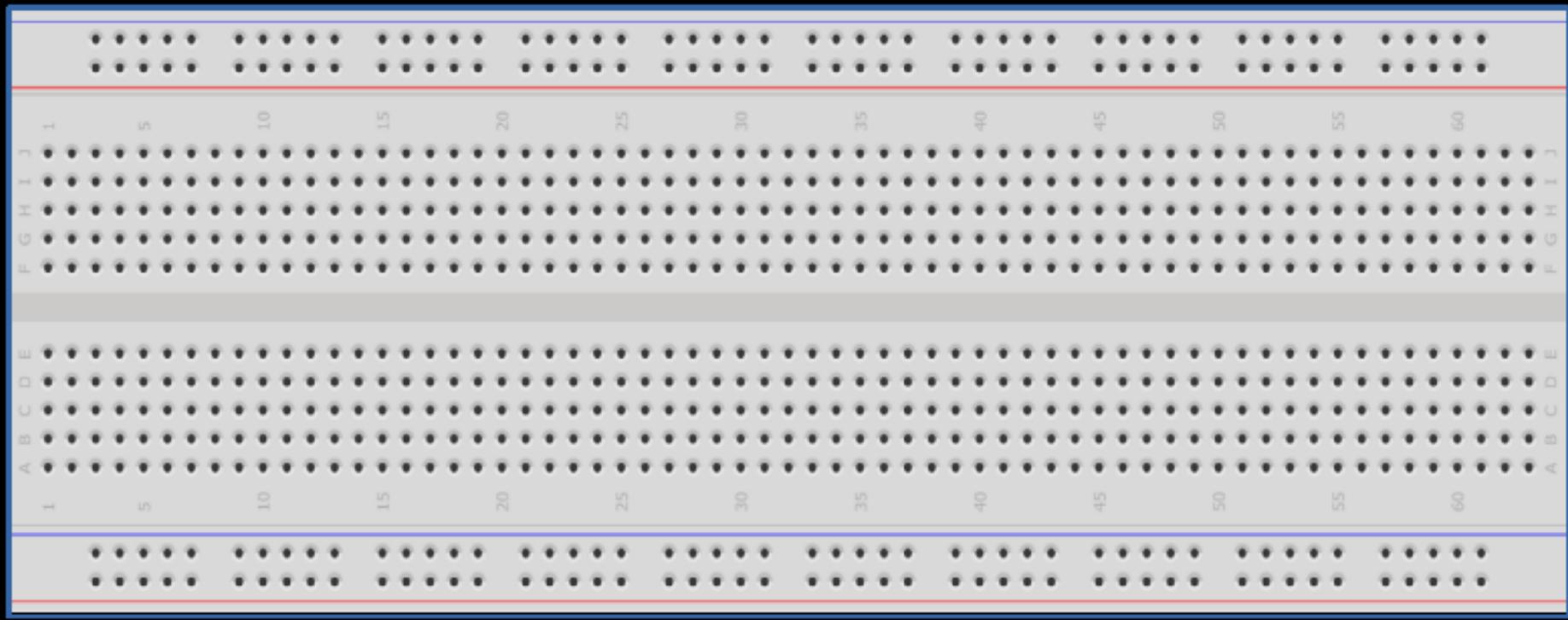
Ethernet Shield



Para ligação dos componentes eletrônicos como led, resistor, sensor de umidade, sensor de temperatura entre vários outros, usa-se a *protoboard*.



A “**protoboard**” ou “**Matriz de contatos**” é utilizada para fazer montagens provisórias e/ou teste de projetos. É constituída por uma base plástica, contendo inúmeros orifícios destinados à inserção de terminais de componentes eletrônicos. Internamente existem ligações determinadas que interconectam os orifícios, permitindo a montagem de circuitos eletrônicos sem a utilização de solda.



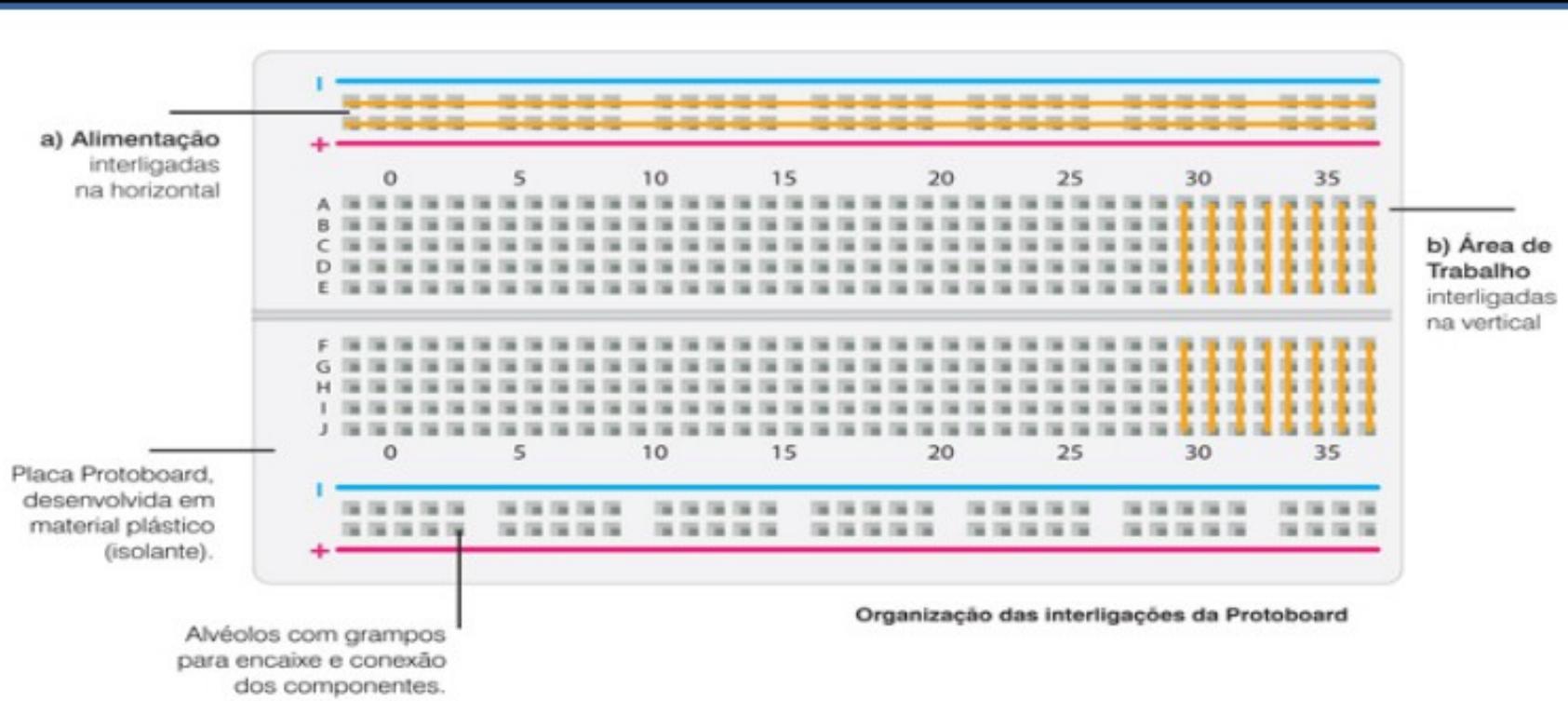
A protoboard possui orifícios dispostos em colunas e linhas. As linhas encontram-se nas extremidades da protoboard e as colunas ao centro.

As colunas são formadas exatamente por cinco furos cada uma.

Todos os cinco orifícios de uma mesma coluna estão internamente conectados. Os orifícios de uma coluna não possuem conexões internas com os de outras colunas.

Os orifícios das linhas estão conectados entre si (em uma mesma linha).

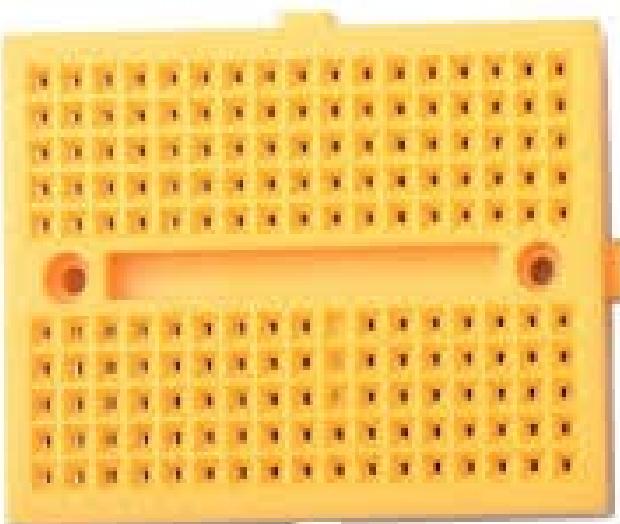
As linhas são eletricamente independentes, isto é, não há conexão elétrica entre os furos de uma linha e de outra.



Proto Shield

★★★★★ (3 Avaliações)

Este shield é uma excelente ferramenta para montagem e testes de seus protótipos com placas Arduino, com ele você pode desenvolver seu próprio shield.



Portas Analógicas X Portas Digitais

O que são Portas Analógicas (entrada)?

Portas analógicas são aquelas que podem assumir valores com intervalo indefinido; Ex: 1 – 1,2 – 1,3 – 2 – 2,99... Como exemplo a temperatura, pressão e umidade são grandezas que variam dessa forma.

No nosso projeto a umidadde do solo será uma grandeza de valor variável e não múltiplo.

O valor de retorno do sensor será interpretado pela porta analógica **A0**.

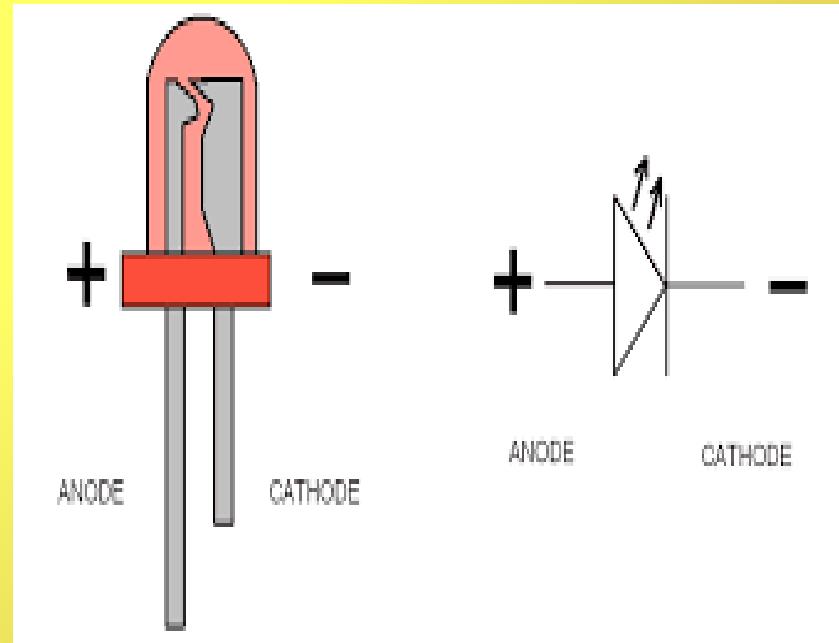


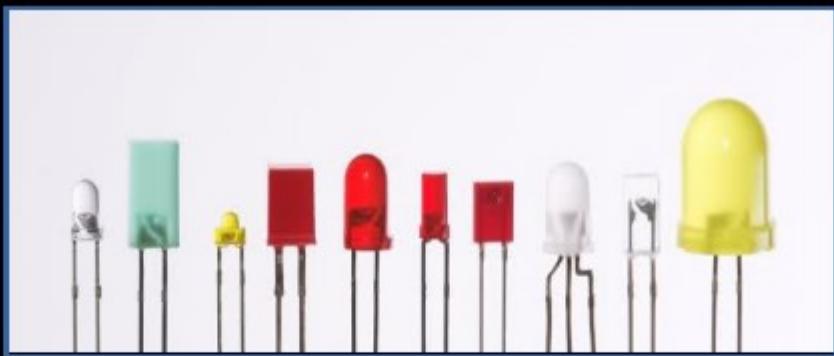
O que são Portas Digitais?

Portas digitais são aquelas que podem assumir apenas dois níveis lógicos bem definidos, nível alto(HIGH) e baixo(LOW). Normalmente, o nível lógico alto é a tensão de alimentação do Arduino (5V ou 3.3V, dependendo do modelo) e nível lógico baixo é 0V (pino conectado ao GND).

As portas digitais são comumente chamadas de I/O ports, que em inglês significa “portas de entrada e saída”. Esse nome vem do fato que uma porta digital poder assumir dois possíveis modos de operação, o modo de **entrada** e o modo de **saída**.

Lâmpadas X LED





O “**diodo emissor de luz**” também é conhecido pela sigla em inglês **LED** (Light Emitting Diode). Sua funcionalidade básica é a emissão de luz em locais e instrumentos onde se torna mais conveniente a sua utilização no lugar de uma lâmpada.



Tensão dos Leds

Tensão: 3.1
Amp: 0.02

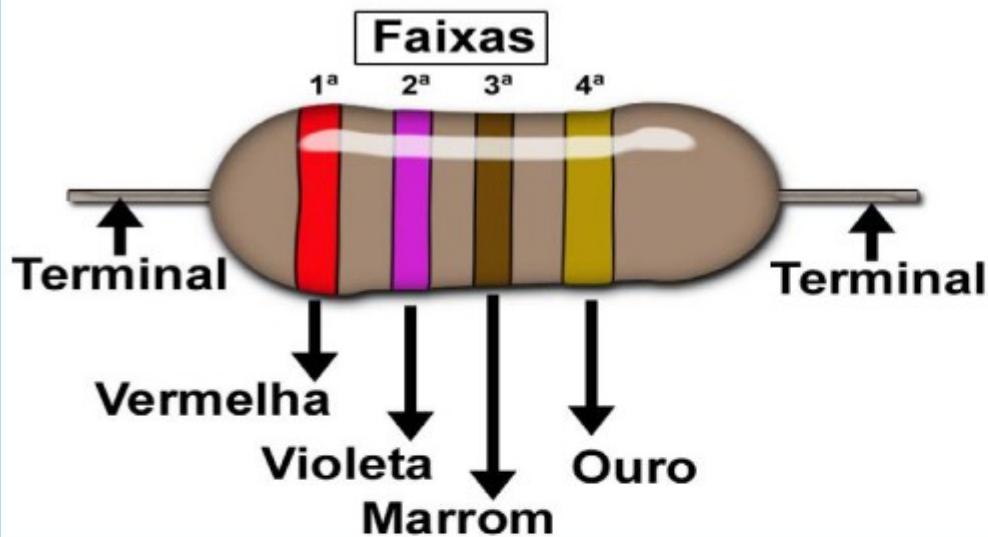
1.8
0.02

3.1
0.02

2.0
0.015

2.1
0.02





O valor da resistência é medido em OHM e seu símbolo e o Ômega Grego Ω

**1 ohm ou 1Ω
1000 ohms = $1K\Omega$
 $1000 K\Omega = 1 M\Omega$**

Um **Resistor** (frequentemente chamado de resistência, que é na verdade a sua medida) é um dispositivo elétrico muito utilizado em eletrônica, ora com a finalidade de transformar energia elétrica em energia térmica por meio do efeito joule, ora com a finalidade de limitar a corrente elétrica em um circuito.

Cálculo de Resistores para Leds

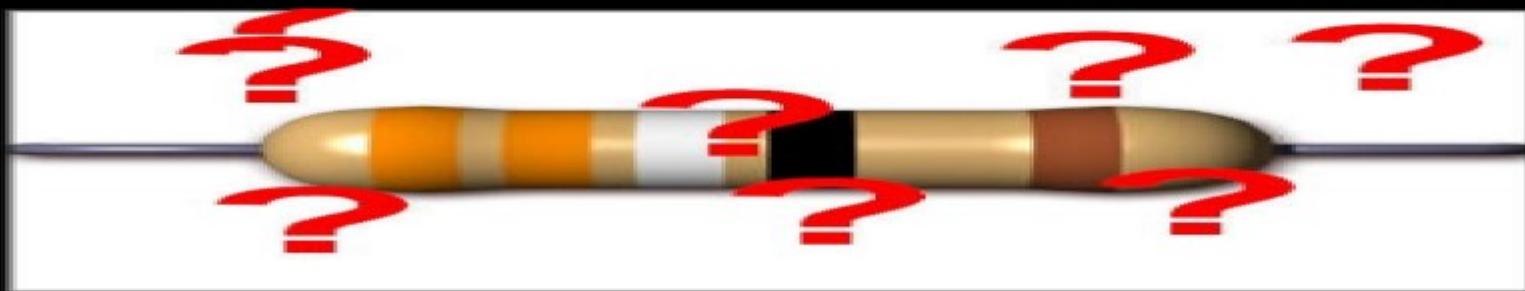
Fórmula: $R = (VF - VL) / CML$

R = Resistor necessário

VF = Voltagem fornecida pela fonte de energia

VL = Voltagem do Led

CML = Corrente máxima do Led



Exemplo: Um led que precisa de 2.1v e sua corrente máxima é de 20 milliampéres, qual resistor deveremos usar?

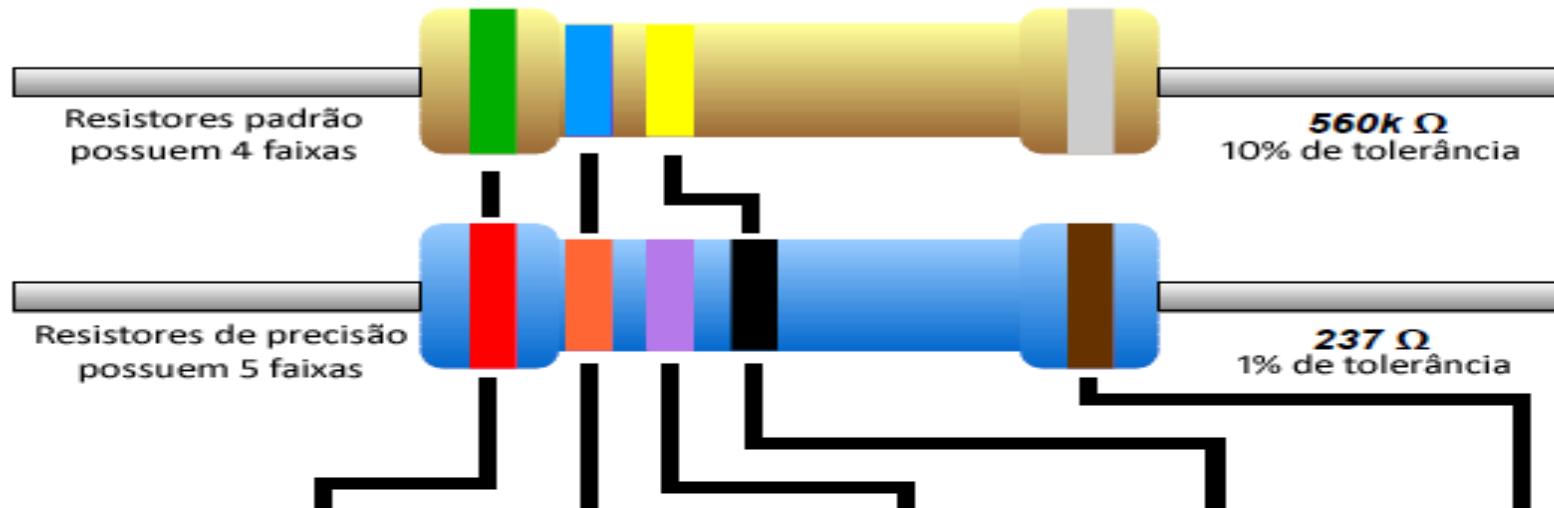
		Led				
		Vermelho	Amarelo	Verde	Branco	Azul
Arduino(v)	5					
Led(v)		1,8	2	2,1	3,1	3,1
Sobra(v)		3,2	3	2,9	1,9	1,9
Sobra/Consumo(a)		0,02	0,02	0,02	0,02	0,02
Resistência(ohms)		160	150	145	95	95

http://www.audioacustica.com.br/exemplos/Valores_Resistores/Calculadora_Grafica_Resistores_4-bandas.html

Tabela de resistores

Código de Cores

A extremidade com mais faixas deve apontar para a esquerda

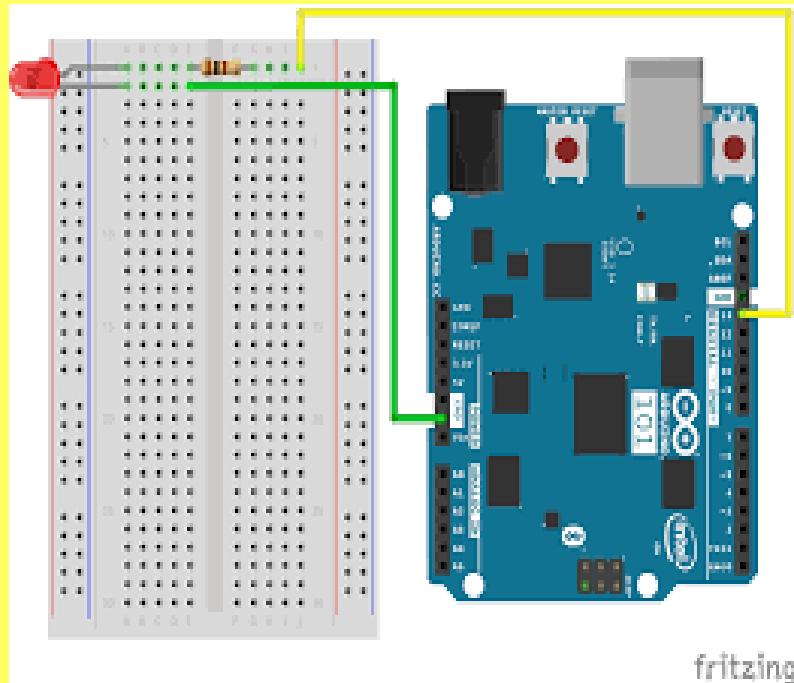


Cor	1 ^a Faixa	2 ^a Faixa	3 ^a Faixa	Multiplicador	Tolerância
Preto	0	0	0	$\times 1 \Omega$	
Marrom	1	1	1	$\times 10 \Omega$	+/- 1%
Vermelho	2	2	2	$\times 100 \Omega$	+/- 2%
Laranja	3	3	3	$\times 1K \Omega$	
Amarelo	4	4	4	$\times 10K \Omega$	
Verde	5	5	5	$\times 100K \Omega$	+/- .5%
Azul	6	6	6	$\times 1M \Omega$	+/- .25%
Violeta	7	7	7	$\times 10M \Omega$	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				$\times .1 \Omega$	+/- 5%
Prateado				$\times .01 \Omega$	+/- 10%

1º desafio: acendendo o LED

Nossa primeira atividade será acender um Led através do Arduino e protoboard. *Mãos à obra...*

Monte o circuito da seguinte forma:



- o pino (-) do Led no resistor e o outro pino do resistor no GND da placa Arduino;
- o pino (+) do Led deverá ser conectado à porta 5v da placa Arduino;
- Conecte o Arduino pela porta USB para alimentação;
O Led deverá acender.

2º desafio...'blink'

- Através do curcuito anterior, vamos montar um 'blink'(pisca).
- Para esta montagem, será necessário configurar uma porta digital para saída da tensão elétrica e juntamente com o código, determinar o tempo em que o led ficará ligado/desligado, em torno de 1s cada;

Arquivo Editar Sketch Ferramentas Ajuda



sketch_apr17a



```
void setup() {  
  // put your setup code here, to run once:  
  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
}
```

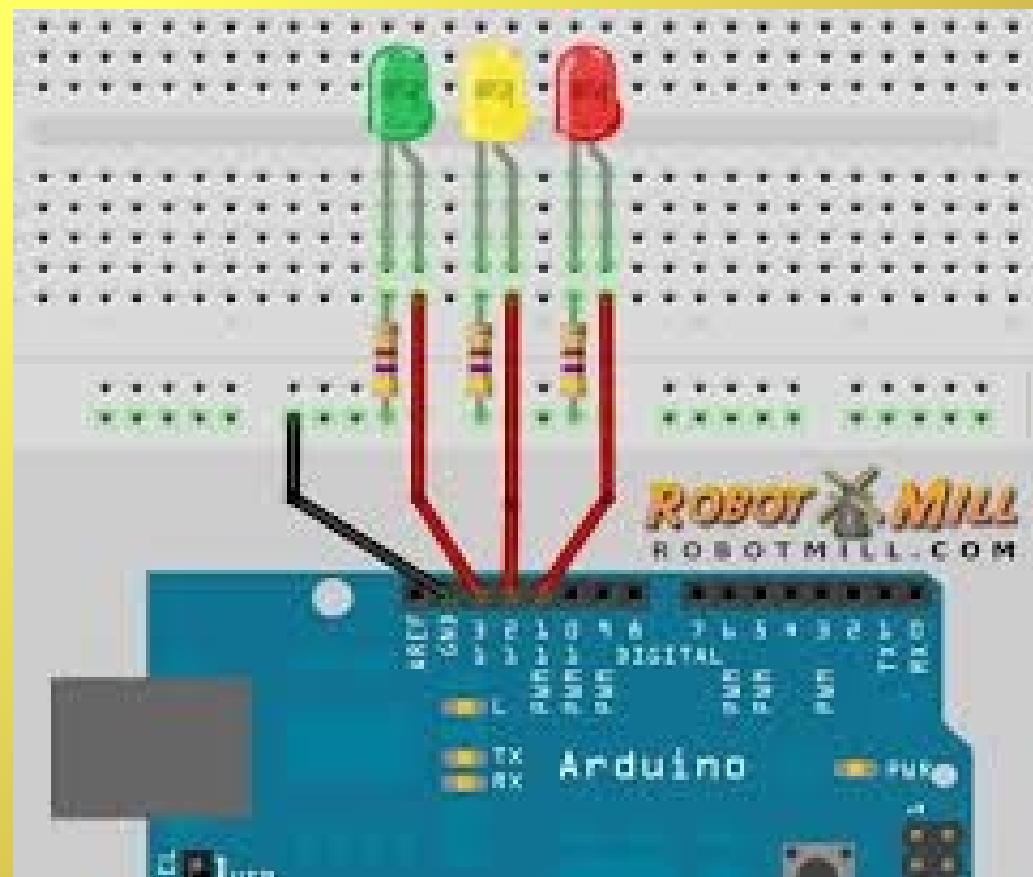
```
const int LED = 2; // inicializa LED na porta digital 2

void setup() {
  // inicializa LED como sendo output.
  pinMode(LED, OUTPUT);
}

// executa este código indefinidamente.....
void loop() {
  digitalWrite(LED, HIGH); // LED (HIGH tensão positiva +5v)
  delay(1000); // aguarda 1000 milisegundos=1segundo
  digitalWrite(LED, LOW); // LED (LOW sem tensão de saída 0v)
  delay(1000); // aguarda 1000 milisegundos=1segundo
}
```

3º desafio

- Montar um 'blink' com três led's:
- Verde
- Amarelo
- Vermelho
- #partiuArduino



Enviando comandos para o Arduino

- O primeiro protótipo terá como desafio ligar/desligar os leds *verde*, *amarelo* e *vermelho* em conjunto e não apenas no formato *blink*(pisca-pisca) que foi o primeiro experimento.
- O circuito elétrico usado anteriormente deverá ser utilizado para ligar os três leds ao mesmo tempo através de comandos, ou seja, será necessário alterar o algoritmo.

Comandos de programação

- Definição dos Leds e portas digitais:

```
const int ledGreen = 2;  
const int ledYellow = 4;  
const int ledRed = 6;
```

```
void setup() {  
    pinMode(ledGreen ,OUTPUT);  
    pinMode(ledYellow ,OUTPUT);  
    pinMode(ledRed ,OUTPUT);  
  
    digitalWrite(ledGreen , LOW);  
    digitalWrite(ledYellow , LOW);  
    digitalWrite(ledRed , LOW);  
}
```

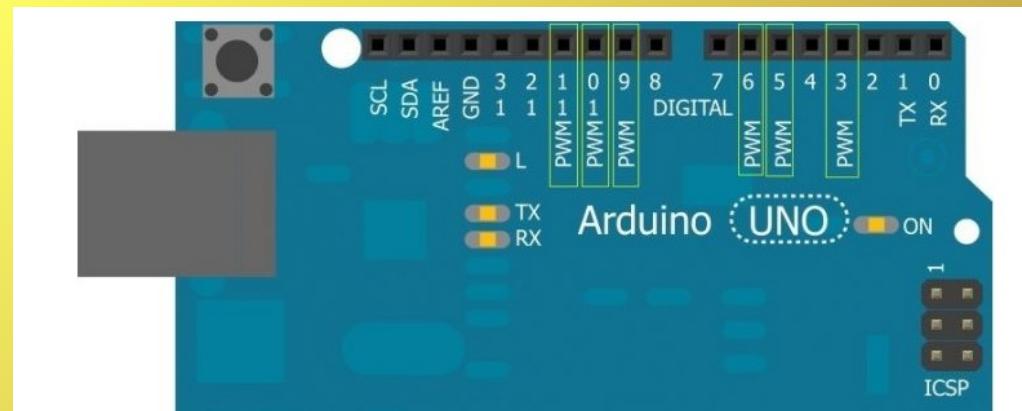
```
void loop() {  
    digitalWrite(ledGreen, HIGH);  
    delay(500);  
    digitalWrite(ledGreen, LOW);  
    delay(500);  
    digitalWrite(ledYellow, HIGH);  
    delay(500);  
    digitalWrite(ledYellow, LOW);  
    delay(500);  
    digitalWrite(ledRed, HIGH);  
    delay(500);  
    digitalWrite(ledRed, LOW);  
    delay(500);  
}
```

Portas PWM

PWM, do inglês Pulse Width Modulation, é uma técnica utilizada por sistemas digitais para variação do valor médio de uma forma de onda periódica. A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto.

PWM do Arduino

A placa Arduino Uno possui pinos específicos para saídas PWM e são indicados pelo caractere ‘~’ na frente de seu número, conforme exibido a seguir:



A função `analogWrite()` escreve um valor de PWM em um pino digital que possui a função PWM. Após a chamada dessa função, o pino passa a operar com uma onda quadrada de frequência fixa e com duty cycle conforme valor passado pela função. A frequência dessa onda, na maioria dos pinos é em torno de 490 Hz, porém, os pinos 5 e 6 da Arduino UNO operam em 980 Hz.

Para utilizar a função `analogWrite()` , deve-se configurar o pino correspondente como saída digital.

A função `analogWrite` deve ser utilizada da seguinte forma:

Sintaxe:

```
analogWrite(pino, valor);
```

O valor deve ser de 0 a 255 onde com 0 a saída permanece sempre em nível baixo e 255 a saída permanece sempre em nível alto.