

FIAP

1. Arduino

Introdução: O ARDUINO

Basicamente o Arduino é uma plataforma de prototipagem “**Open Source**” de eletrônica que foi desenvolvida para fins educacionais, para projetistas amadores (Makers) e facilitar o desenvolvimento de provas de conceitos (POCs).

Pequeno computador com hardware limitado, livre e de placa única



Introdução: Projeto ARDUINO – arquitetura e história



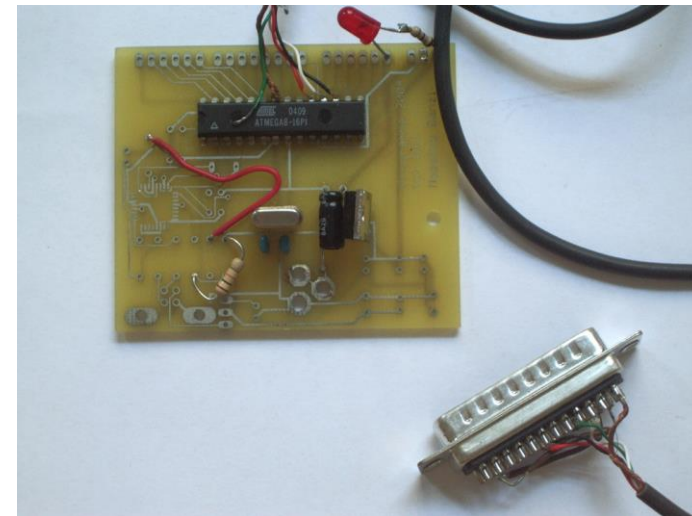
- 1 - Conector USB para o cabo tipo AB
- 2 - Botão de reset
- 3 - Pinos de entrada e saída digital e PWM
- 4 - LED verde de placa ligada
- 5 - LED laranja conectado ao pin13
- 6 - ATmega encarregado da comunicação com o computador
- 7 - LED TX (transmissor) e RX (receptor) da comunicação serial
- 8 - Porta ICSP para programação serial
- 9 - Microcontrolador ATmega 328, cérebro do Arduino
- 10 - Cristal de quartzo 16Mhz
- 11 - Regulador de tensão
- 12 - Conector Jack fêmea 2,1mm com centro positivo
- 13 - Pinos de tensão e terra
- 14 - Pinos de entrada analógica

Introdução: Projeto ARDUINO – arquitetura e história

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.



David Cuartielles, Gianluca Martino, Tom Igoe, David Mellis, and Massimo Banzi



Primeiro protótipo 2005

Introdução: Modelos de placas



Microcontrolador	ATmega328P	ATmega32u4	Intel Curie	ATmega32u4
Tensão de operação	5V	5V	3.3V (5V tolerant I/O)	5V
Tensão de alimentação	7-12V	7-12V	7-12V	
Pinos I/O digital	14 (of which 6 provide PWM output)	20	14 (of which 4 provide PWM output)	
Pinos I/O PWM digital	6	7	4	
Pinos analógicos	6	12	6	
Corrente DC por pino I/O	20mA	40mA	20mA	
Corrente DC por pino I/O de 3,3V	50mA	50mA		

Introdução: Modelos de placas



Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader	32 KB (ATmega32u4) of which 4 KB used by bootloader	196 kB	32 KB of which 4 KB used by bootloader
SRAM	2 KB (ATmega328P)	2.5 KB (ATmega32u4)	24KB	2.5 KB
EEPROM	1 KB (ATmega328P)	1 KB (ATmega32u4)		1 KB
Clock Speed	16 MHz	16 MHz	32Mhz	16 MHz
Peso	25g	20g	34g	53g
Features			Bluetooth LE, 6-axis accelerometer/gyro	Analog joystick; Microphone; Light sensor; Temperature sensor ; three-axis accelerometer; Buzzer

Introdução: Modelos de placas



Microcontrolador	ATmega32U4	ATmega328
Tensão de operação	5V	5V
Tensão de alimentação	7-12V	
Pinos I/O digital	20	22
Pinos I/O PWM digital	7	6
Pinos analógicos	12	8
Corrente DC por pino I/O	20mA	40mA
Corrente DC por pino I/O de 3,3V	50mA	

Introdução: Modelos de placas



Flash Memory	32 KB (ATmega32U4) of which 4 KB used by bootloader	32 KB of which 2 KB used by bootloader
SRAM	2.5 KB (ATmega32U4)	2 KB
EEPROM	1 KB (ATmega32U4)	1 KB
Clock Speed	16 MHz	16 MHz
Peso	13g	7g
Comprimento	48 mm	45 mm
Largura	18 mm	18 mm

Introdução: Modelos de placas



Microcontrolador	ATmega2560	ATSAMD21G18, 32-Bit ARM Cortex M0+	AT91SAM3X8E
Tensão de operação	5V	3,3V	3,3V
Tensão de alimentação	7-12V		7-12V
Pinos I/O digital	54	20	54
Pinos I/O PWM digital	15	7	12
Pinos analógicos	16	6, 12-bit ADC channels	
Corrente DC por pino I/O	20mA	7mA	130 mA (juntos)
Corrente DC por pino I/O de 3,3V	50mA		800 mA

Introdução: Modelos de placas



Flash Memory	256 KB of which 8 KB used by bootloader	256 KB	512 KB
SRAM	8 KB	32 KB	96 KB
EEPROM	4 KB		
Clock Speed	16 MHz	48 MHz	84 MHz
Peso	37 g	12g	36g

Ambiente de programação

- Ambiente integrado de Desenvolvimento (IDE)

Pode ser gratuitamente baixado do site www.arduino.cc



Download the Arduino IDE

ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
 Get 

Mac OS X 10.8 Mountain Lion or newer

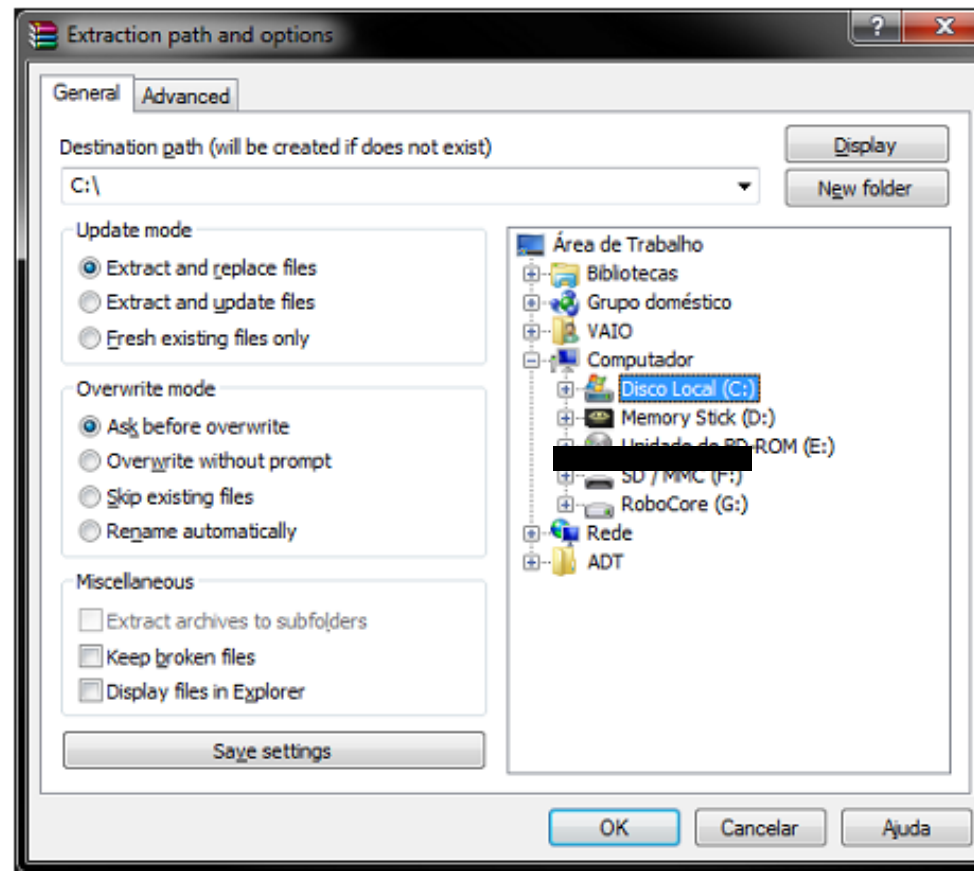
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

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

Ambiente de programação

- Ambiente integrado de Desenvolvimento (IDE)

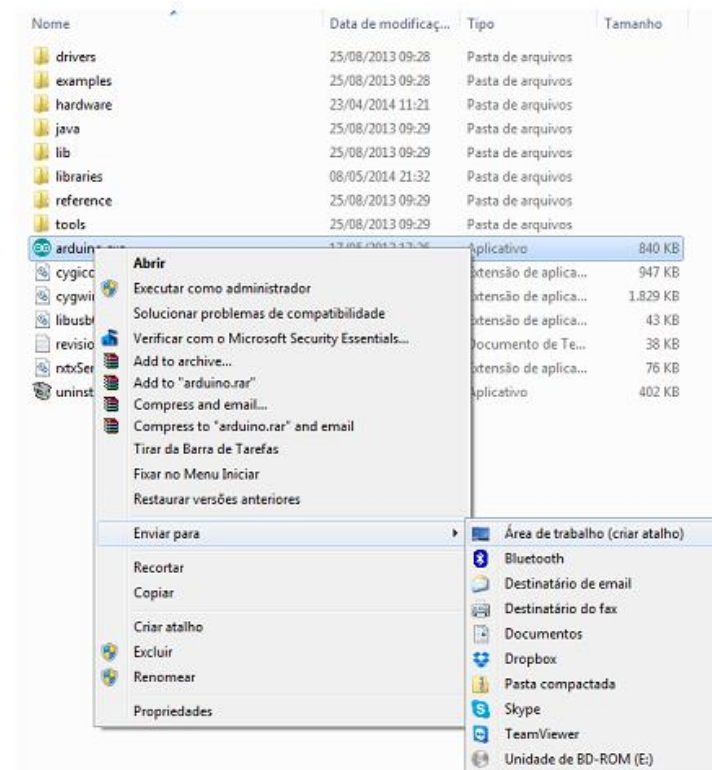
Quando finalizar o download, descompacte a pasta no diretório: **C:** conforme apresentado na figura abaixo.



Ambiente de programação

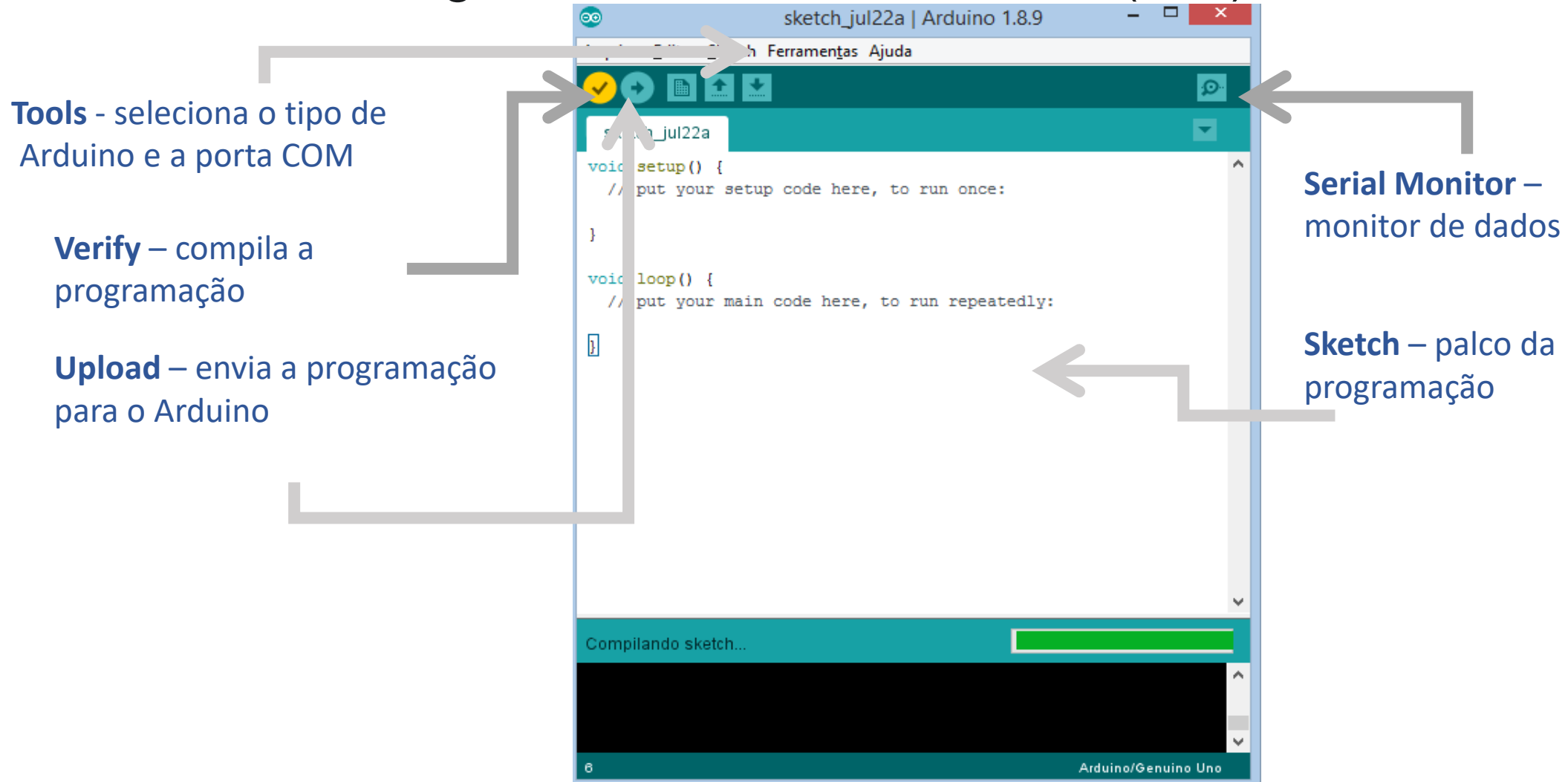
- Ambiente integrado de Desenvolvimento (IDE)

Agora basta criar um atalho da IDE na área de trabalho e você já poderá programar sua placa!



Ambiente de programação

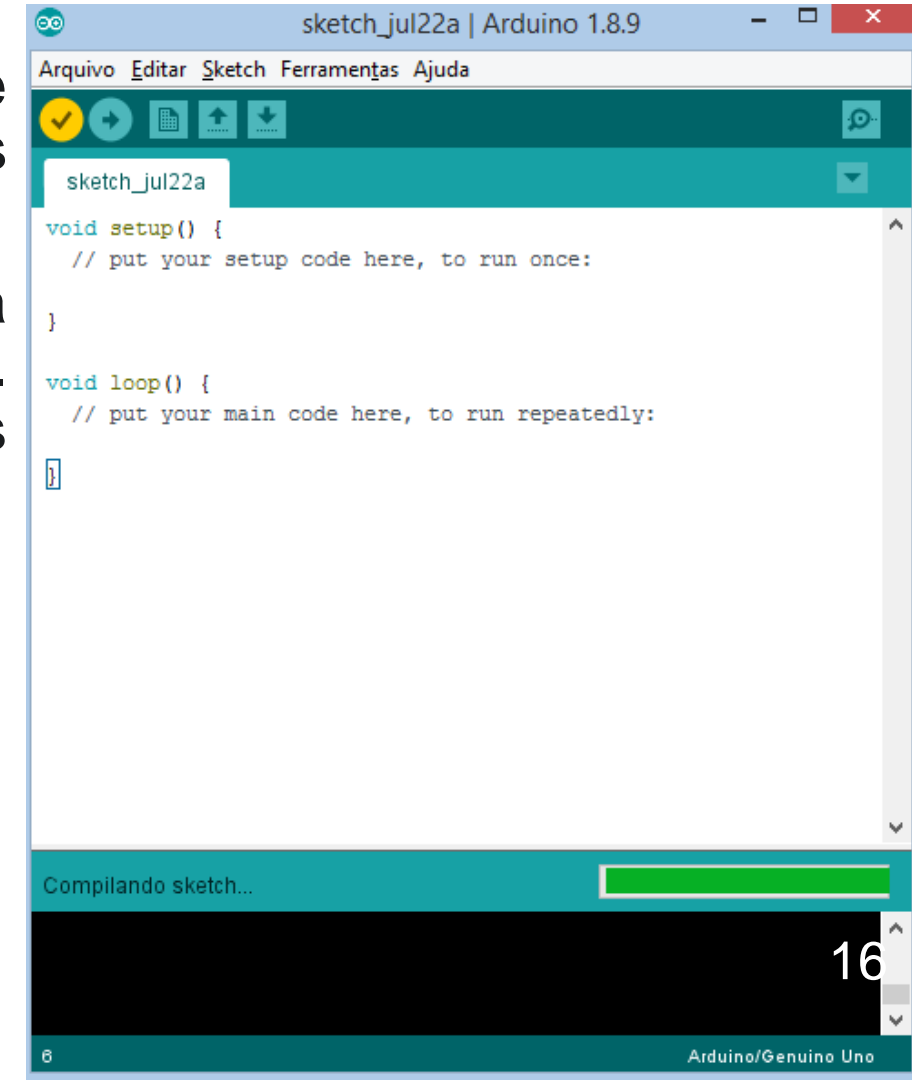
- Ambiente integrado de Desenvolvimento (IDE)



Ambiente de programação

- O IDE é muito simples e intuitivo. Um programa, que no Arduino é chamado de sketch, apresenta duas funções básicas: `setup()` e `loop()`.
- A função **`setup()`** deverá conter o código que irá executar apenas uma vez, quando o sketch iniciar. Normalmente colocamos nesta função as definições iniciais do programa.

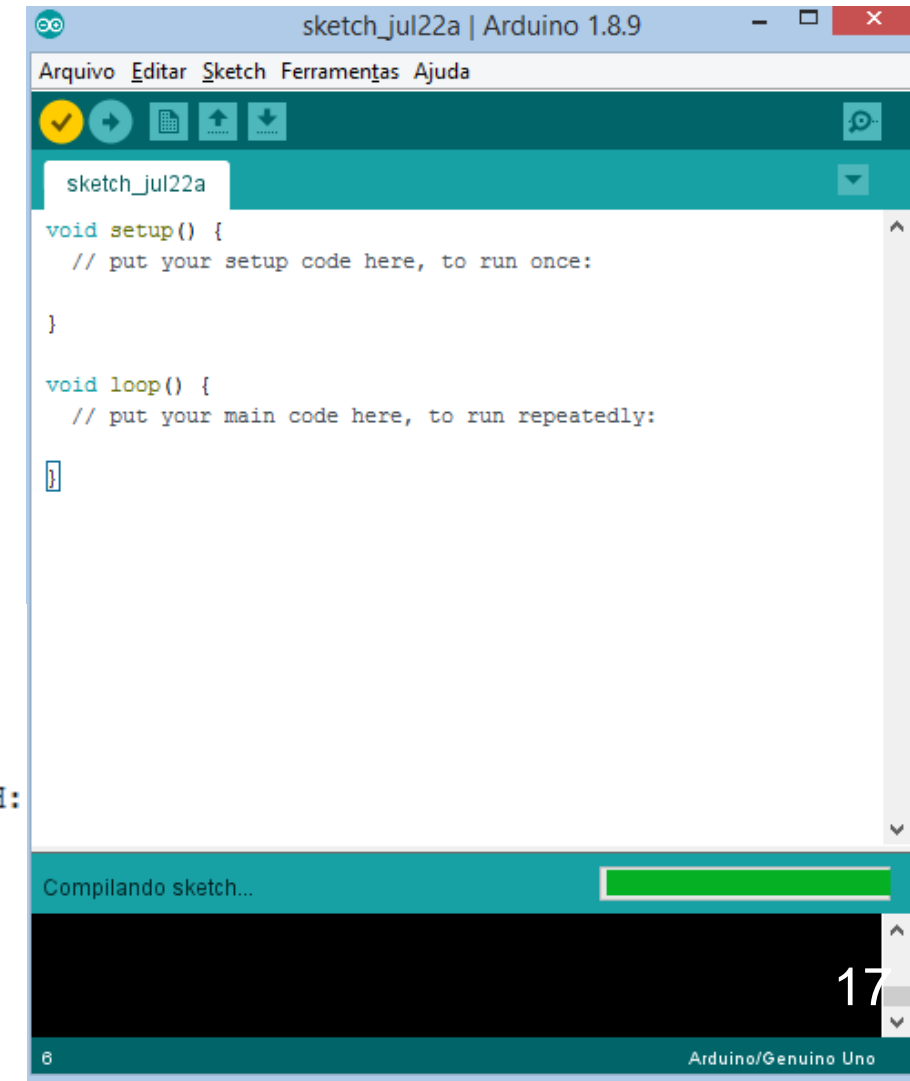
```
void setup() {  
    // initialize the LED pin as an output:  
    pinMode(ledPin, OUTPUT);  
    // initialize the pushbutton pin as an input:  
    pinMode(buttonPin, INPUT);  
}
```



Ambiente de programação

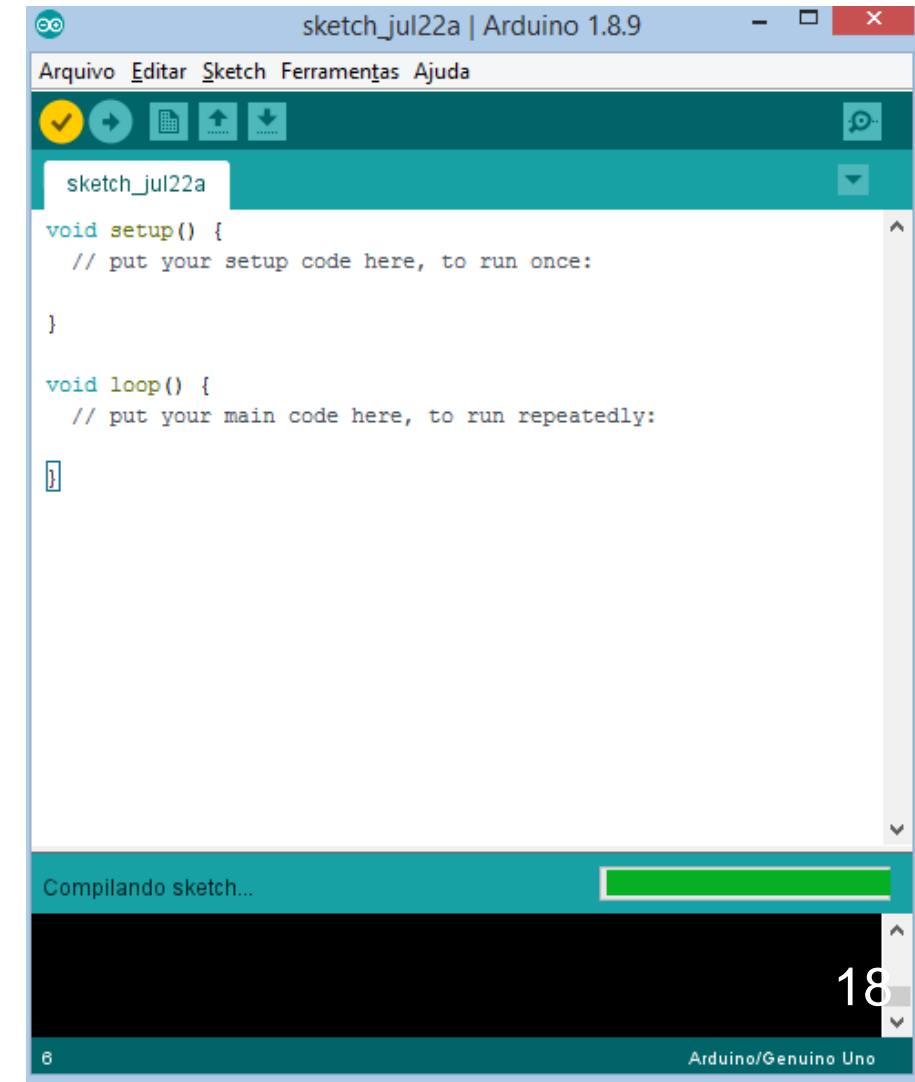
- A função **loop()** irá executar continuamente as instruções que estão lá até que outro sketch seja carregado na memória “flash” do Arduino.
- É importante notar que no Arduino é possível armazenar e executar um sketch por vez, desta forma, sempre quando transferimos um sketch esse irá substituir o programa que estava anteriormente carregado na memória.

```
void loop() {  
  // read the state of the pushbutton value:  
  buttonState = digitalRead(buttonPin);  
  
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:  
  if (buttonState == HIGH) {  
    // turn LED on:  
    digitalWrite(ledPin, HIGH);  
  } else {  
    // turn LED off:  
    digitalWrite(ledPin, LOW);  
  }  
}
```



Ambiente de programação

- Também observe que como o sketch fica armazenado na memória “flash”, que é permanente, mesmo quando desligamos o Arduino, o programa continua armazenado e irá entrar novamente em execução quando o Arduino for ligado novamente.
- Note também que, nestas duas funções, a palavra reservada **void** indica que as funções não apresentam um valor de retorno, sendo usadas exclusivamente para realizar a execução de um conjunto de instruções.



Laboratório – TinkerCAD

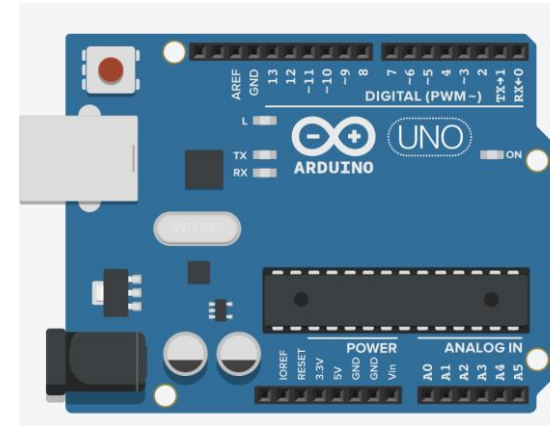
O Objetivo deste laboratório é conhecer o aplicativo

TinkerCad, um simulador de circuitos eletrônicos básicos.

Usaremos esse simulador como ferramenta para testarmos e avaliarmos nossos projetos, antes de partirmos para a montagem prática, pois “Se no simulador funciona, e na montagem não, então tem algum fio solto...”



AUTODESK Tinkercad



Acesse o <https://www.tinkercad.com/>, crie uma conta e vamos montar o nosso primeiro projeto!

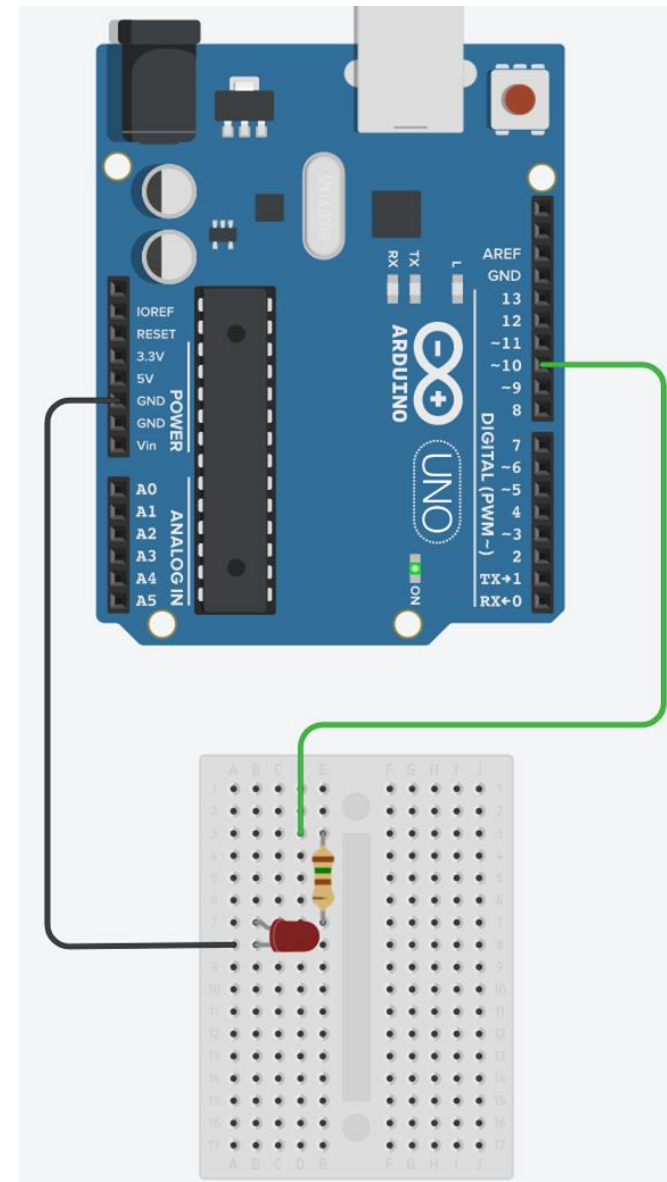
Laboratório – Piscando LED

Agora que temos acesso ao **TinkerCad**, vamos montar o nosso primeiro circuito. **Um Pisca Led Simples.**

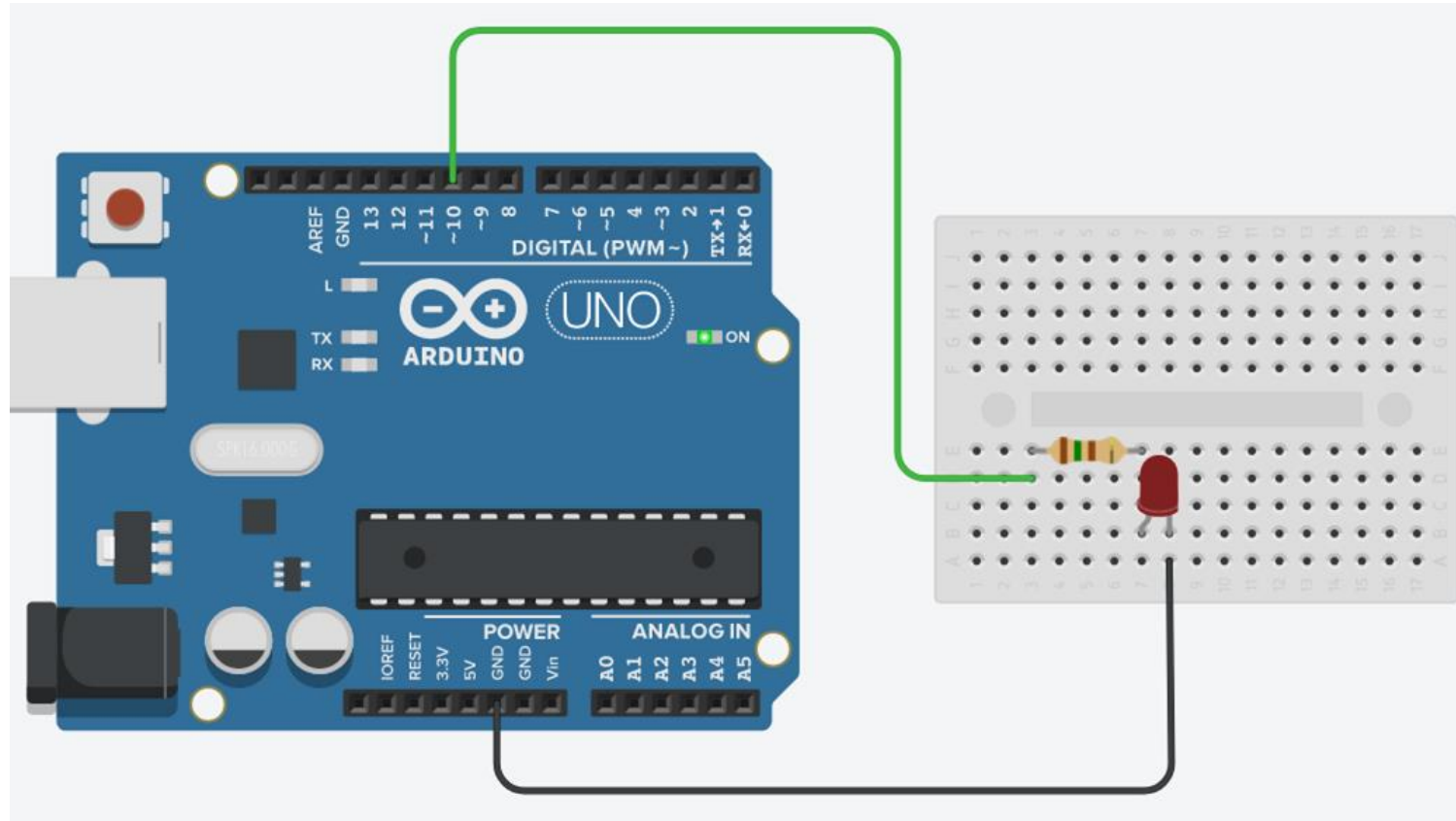
Nesse projeto vamos conhecer a interface de programação do Arduino e entender um poquinho como o hardware de prototipagem funciona.

Material necessário:

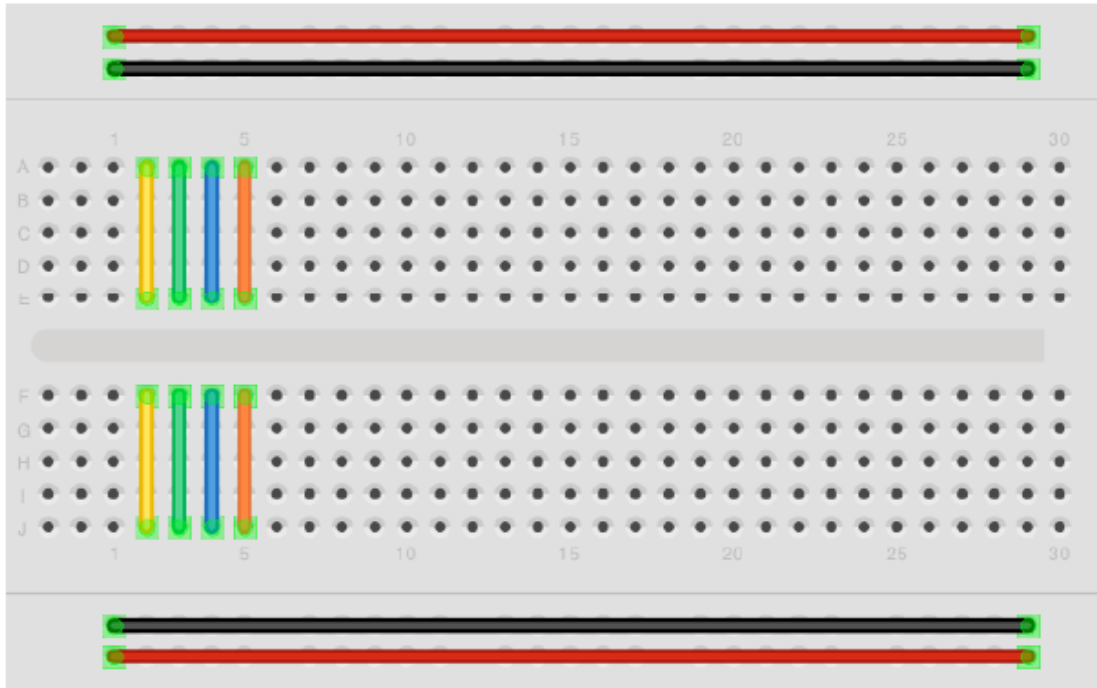
- 1 Arduino;
- 1 Resistor de 150;
- 1 Led (qualquer cor);
- 1 Protoboard;
- Jumpers cables.



Conhecendo o Hardware



Conhecendo o Hardware – Protoboard – Matriz de Contatos Elétricos



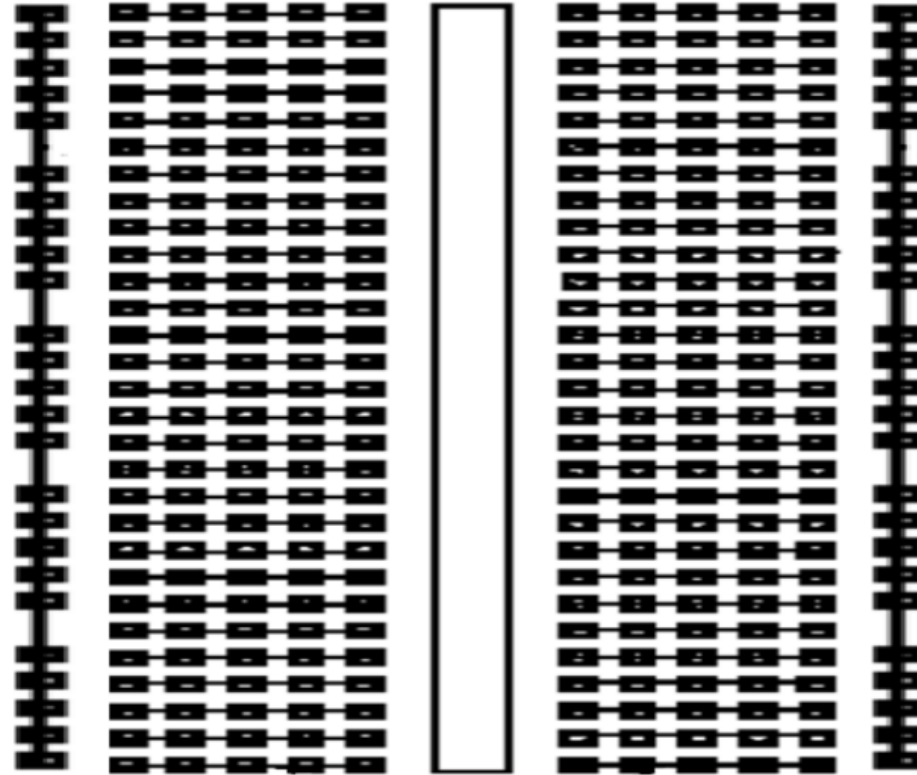
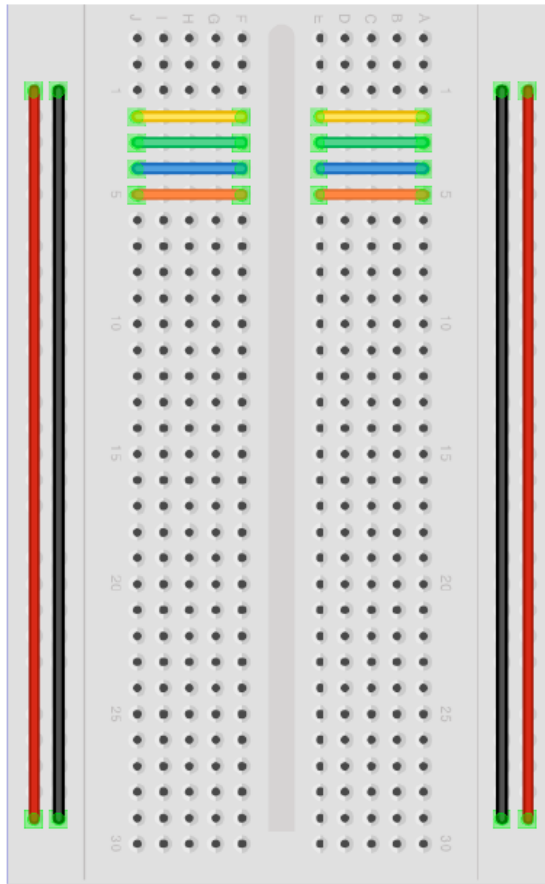
A linha **Vermelha** é toda interligada e serve para ligar o **Positivo** da fonte de alimentação: VCC, VDD, 3.3V, 5V, 12V, +

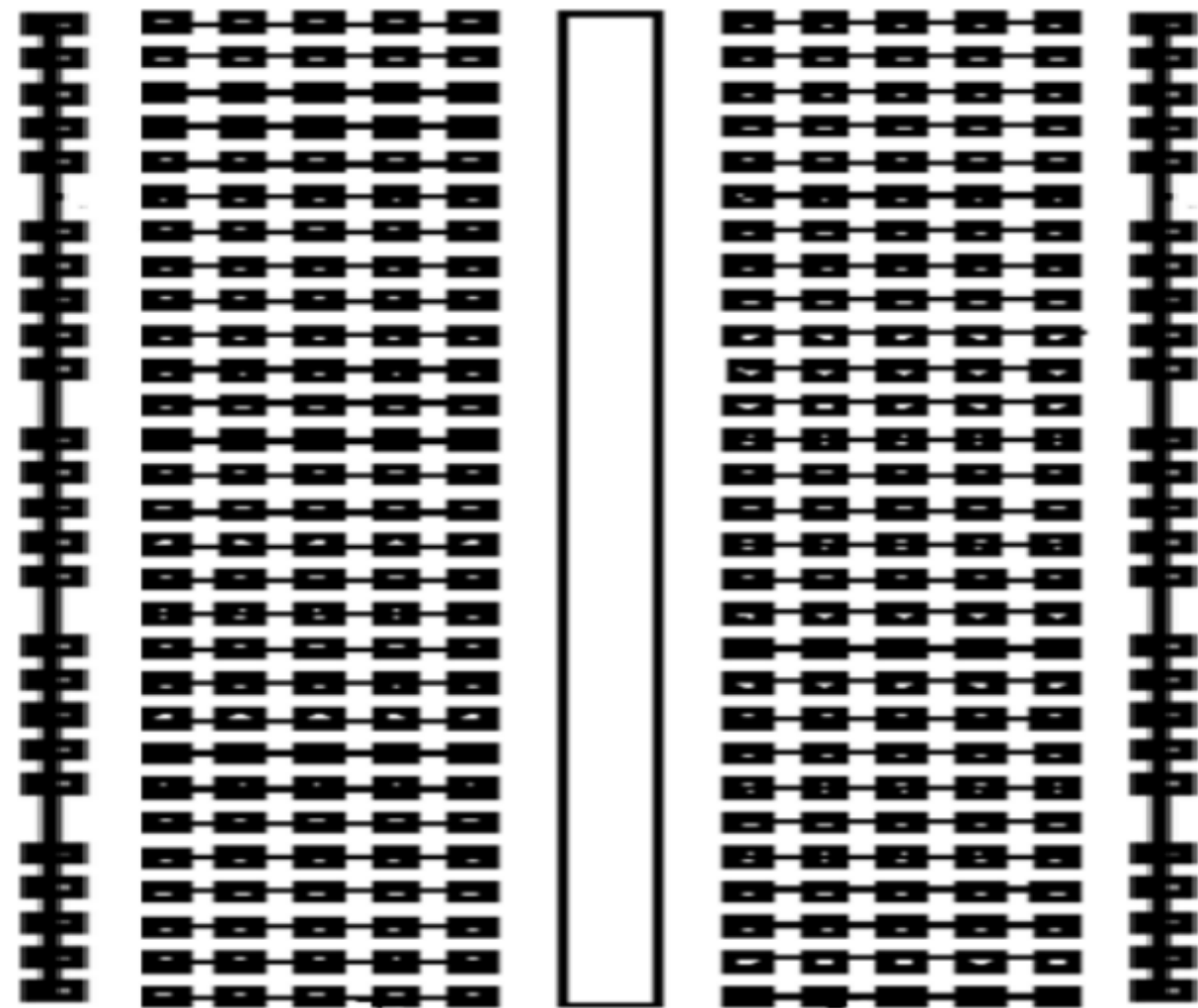
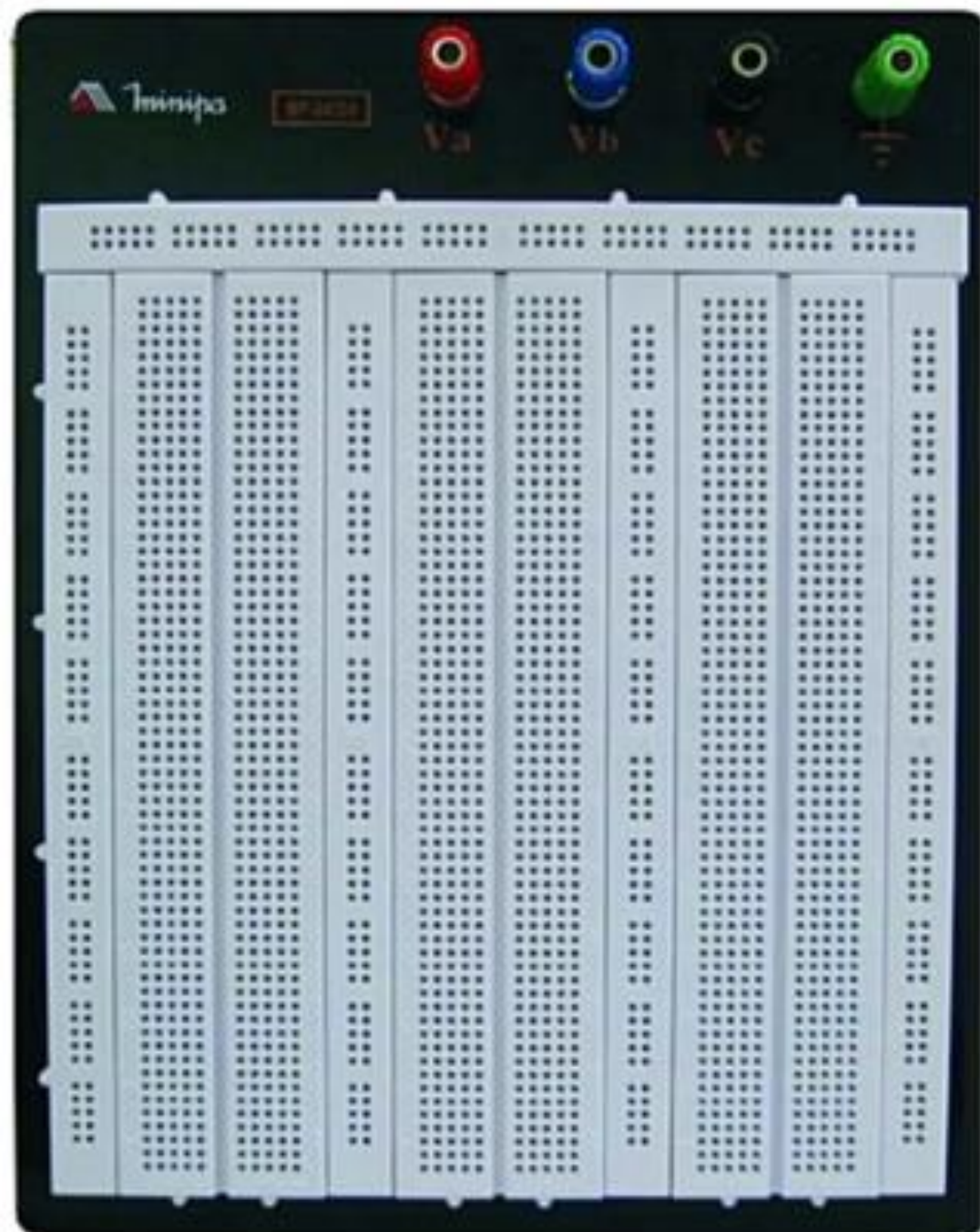
A linha **Preta** é toda interligada e serve para ligar o **Negativo** da fonte de alimentação: GND, VSS, 0V, Terra, -

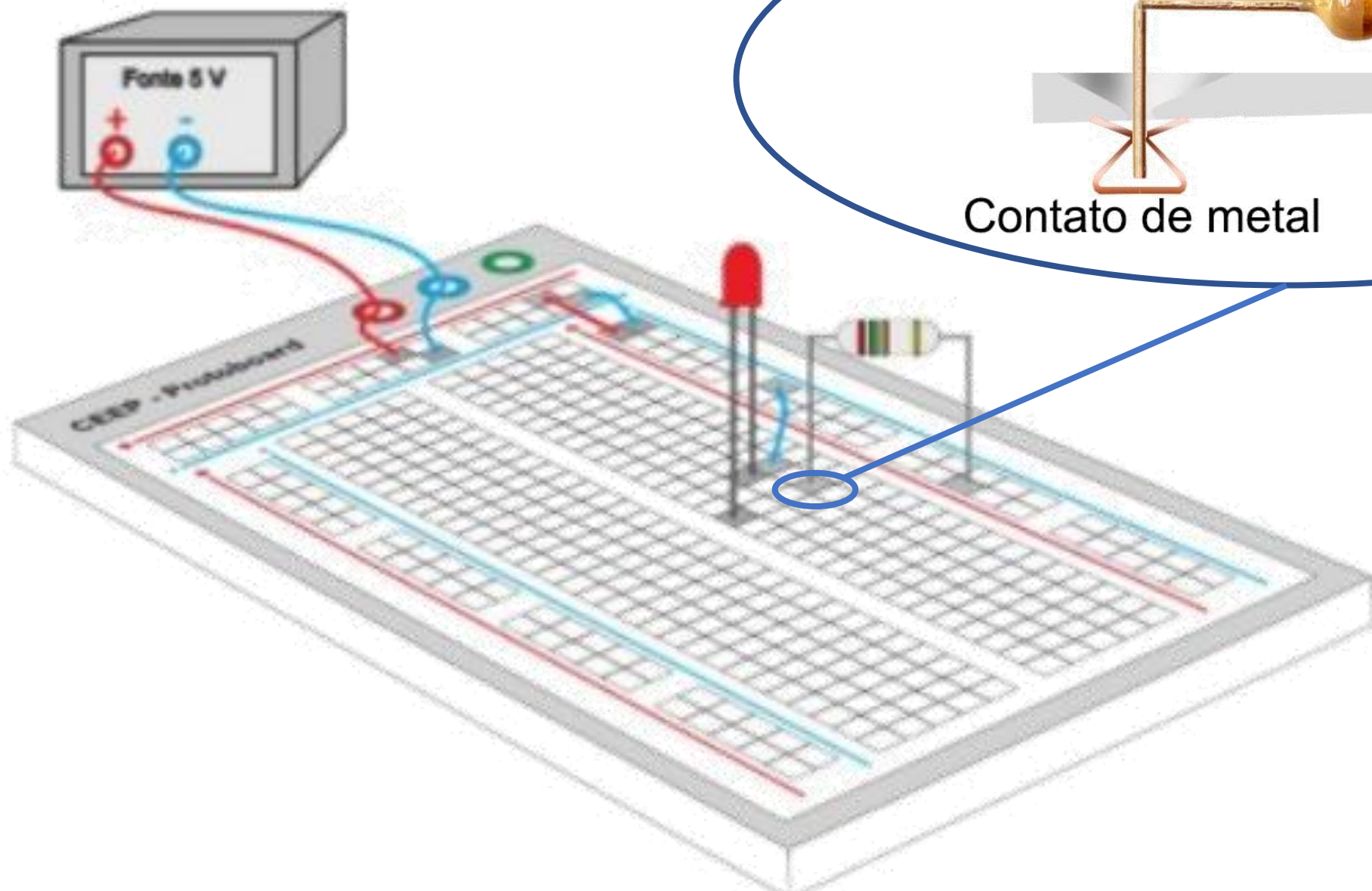
As linhas A, B, C, D e E estão ligadas na **VERTICAL**, em forma de colunas, e uma **coluna não fala com a outra**.

As linhas F, G, H, I e J seguem o mesmo padrão, com a diferença que **não falam com a coluna de cima**.

Conhecendo o Hardware – Protoboard – Matriz de Contatos Elétricos







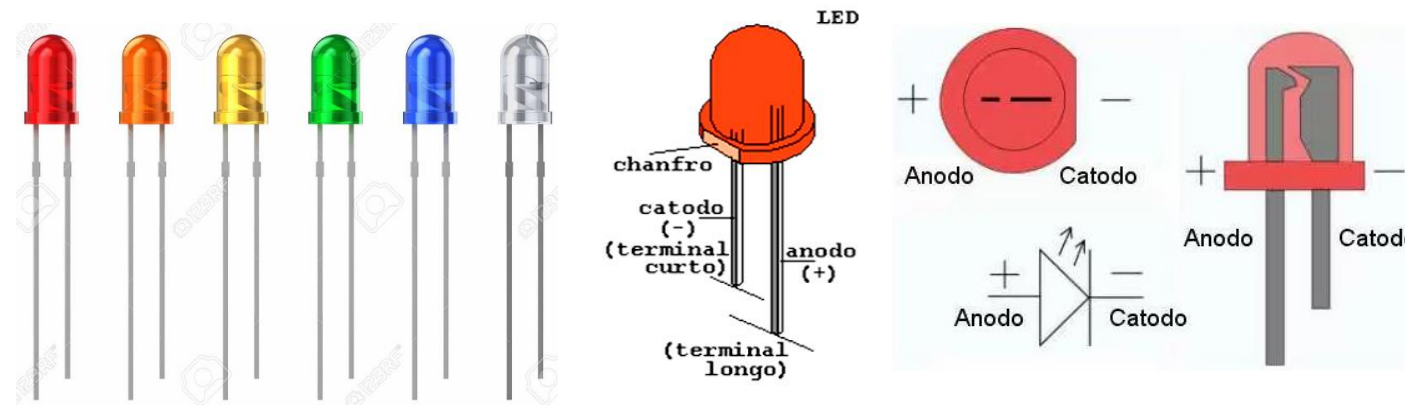
Terminal

Componente

Contato de metal

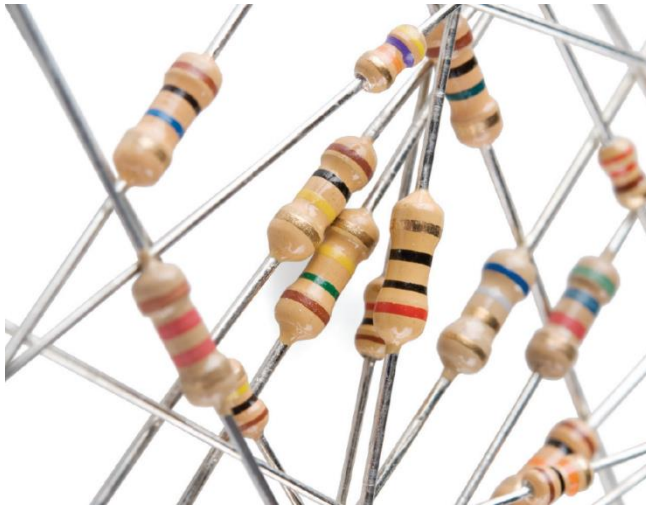
Conhecendo o Hardware – LED

O “**LED**” é um dispositivo emissor de luz



- As informações mais importantes são: **Polaridade**, **Tensão Limite** e a **Corrente Máxima**;
- O Led tem a posição correta de ser ligado, onde tem um chanfro ou terminal menor é o cátodo (**Negativo**) e o terminal maior é o ânodo (**Positivo**)
- Existe em diversos tamanhos e formatos redondo, quadrado, retangular, pequenos, grandes...

Conhecendo o Hardware – Resistor



Componente eletrônico usado para limitar a passagem de corrente elétrica;



Causam uma queda de tensão controlada no circuito eletrônico;



Sua medida é em **Ohms (Ω)** e são regidos pela Lei de Ohm;



Possuem muitos valores e são identificados por um Código de Cores;



Também são usados para esquentar alguma coisa (chuveiro);

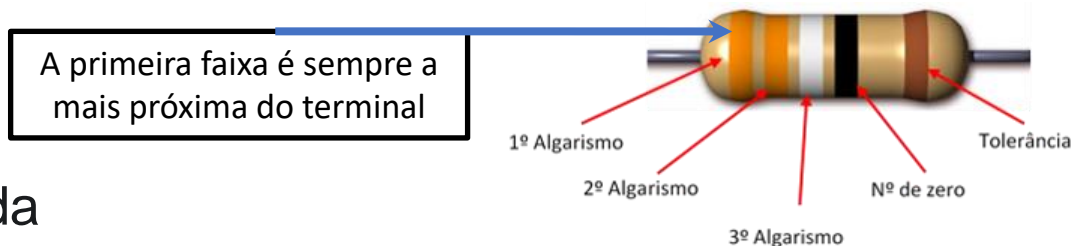
Conhecendo o Hardware – Resistor

Os “**resistores**” são componentes com a finalidade de oferecer resistência à passagem da corrente elétrica.

Quanto vale esse resistor?

1ª Faixa – Laranja -> 3
2ª Faixa – Laranja -> 3
3ª Faixa – Branco -> 9
4ª Faixa – Preto ->
Mult. 1
5ª Faixa – Marrom -> 1%

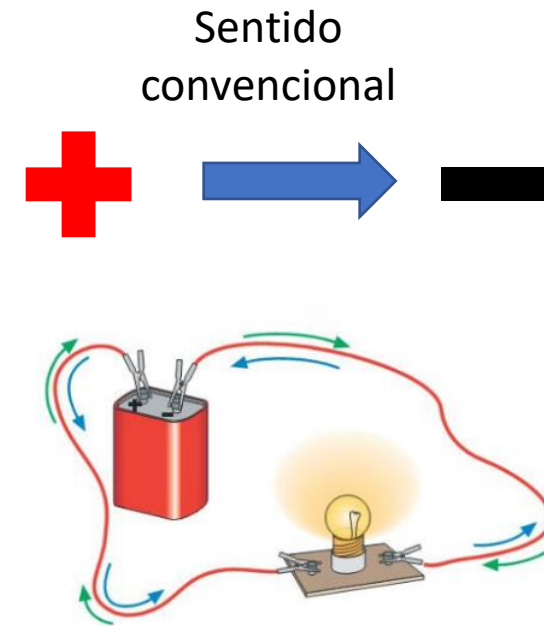
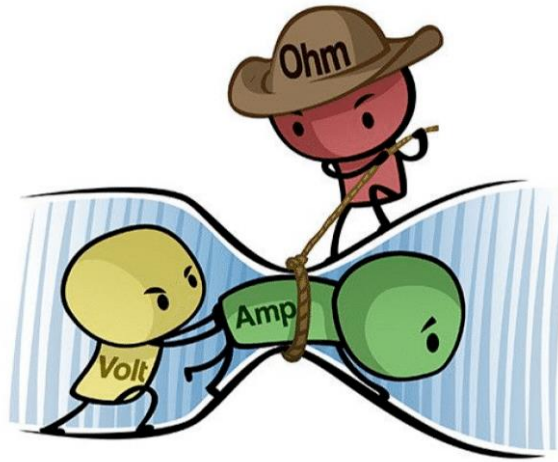
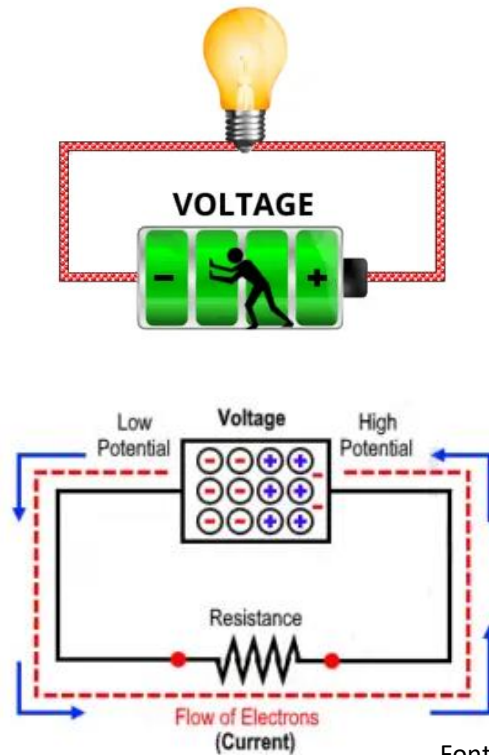
Resistor = 339 x 1, 1%
Resistor = 339 Ohms +/- 1%



Cores	Valores			Multiplicadores X	Tolerância %
	Faixa 1	Faixa 2	Faixa 3		
Prata	-	-	-	0,01	10%
Ouro	-	-	-	0,1	5%
Preto	-	0	0	1	-
Marrom	1	1	1	10	1%
Vermelho	2	2	2	100	2%
Laranja	3	3	3	1000	-
Amarelo	4	4	4	10000	-
Verde	5	5	5	100000	5%
Azul	6	6	6	1000000	0,25%
Violeta	7	7	7	10000000	0,10%
Cinza	8	8	8	-	-
Branco	9	9	9	-	-
Sem cor	-	-	-	-	20%

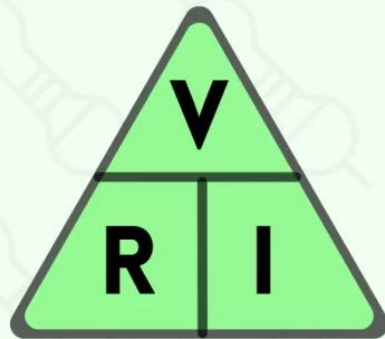
Fonte: <https://aprendendoeletrica.com/codigo-de-cores-para-resistores/>

Conhecendo o Hardware – Resistor



Fonte: <https://portald Engenharia.com/instalacao-eletrica/o-que-e-tensao-eletrica/>
Fonte: <https://embarcados.com.br/lei-de-ohm/>

LEI DE OHM

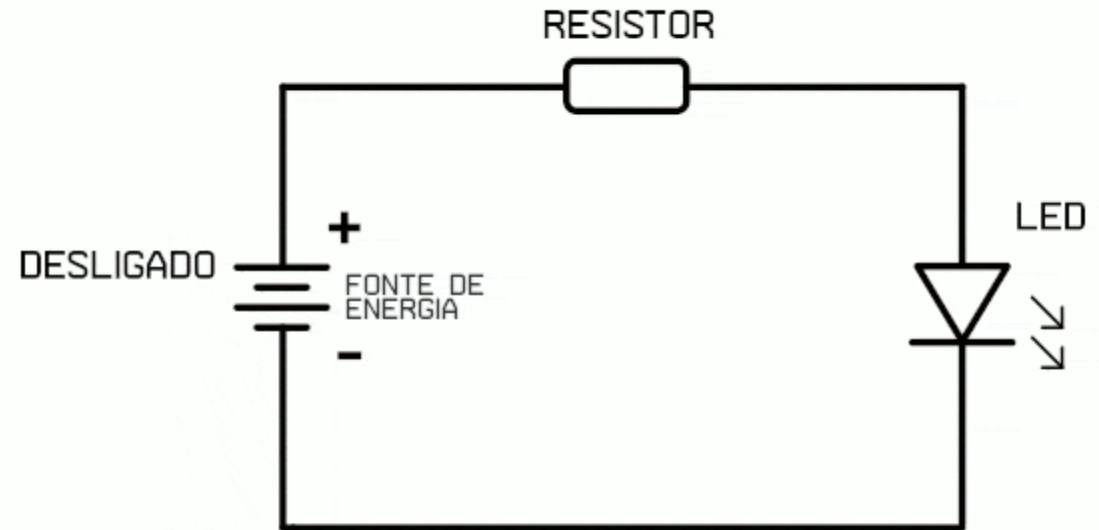


@Trick.Drawing


V = Tensão elétrica [V];


R = Resistência elétrica [Ω];


I = Corrente elétrica [A].



Conhecendo o Software

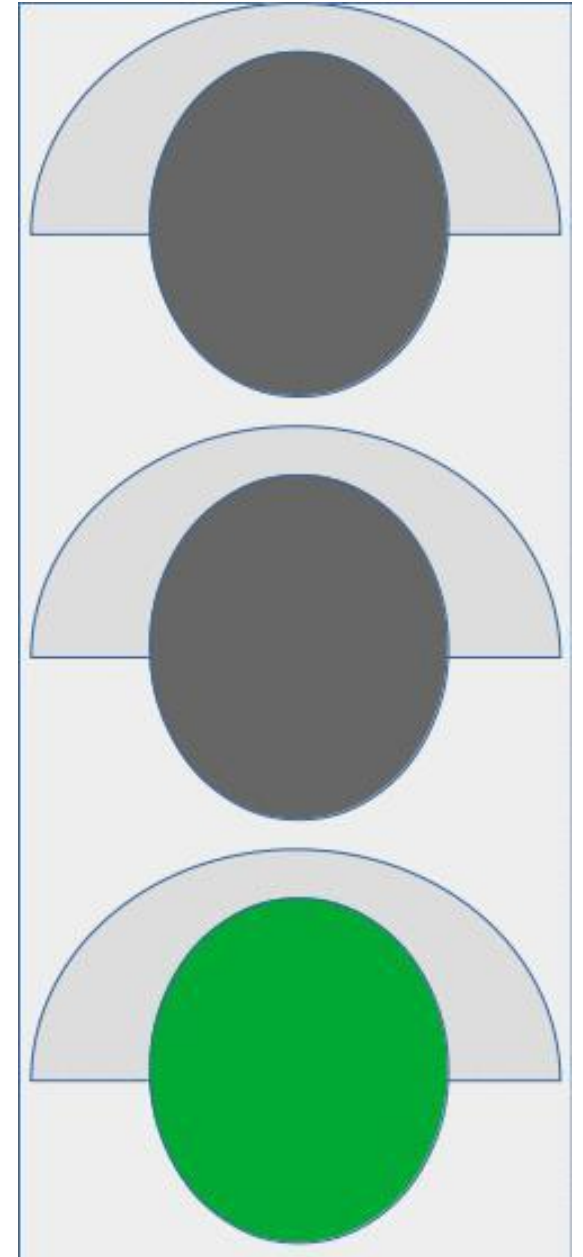
 Primeiro declaramos em qual pino vamos ligar o Led, nesse caso, no pino 10;

 Depois, na função setup, é onde falamos que esse pino será uma saída.

 E por fim, na função loop, é onde roda o nosso programa principal.

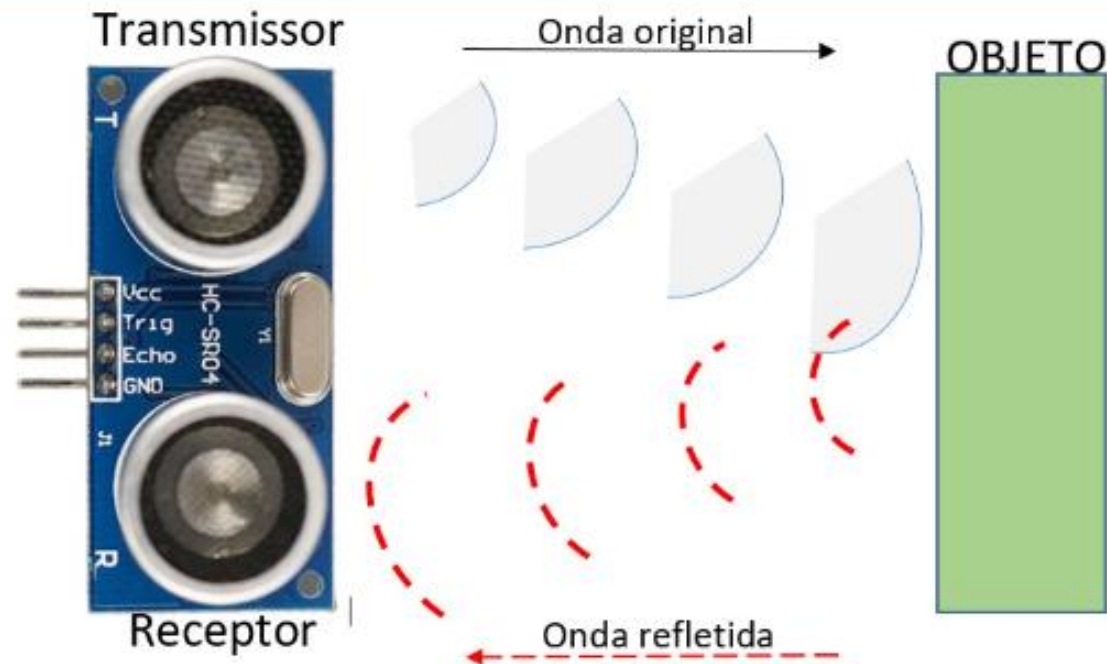
```
1 // Project 1 - LED Flasher
2 int ledPin = 10;
3
4 void setup() {
5     pinMode(ledPin, OUTPUT);
6 }
7
8 void loop() {
9     digitalWrite(ledPin, HIGH);
10    delay(1000);
11    digitalWrite(ledPin, LOW);
12    delay(1000);
13 }
14
```

DESAFIO 1

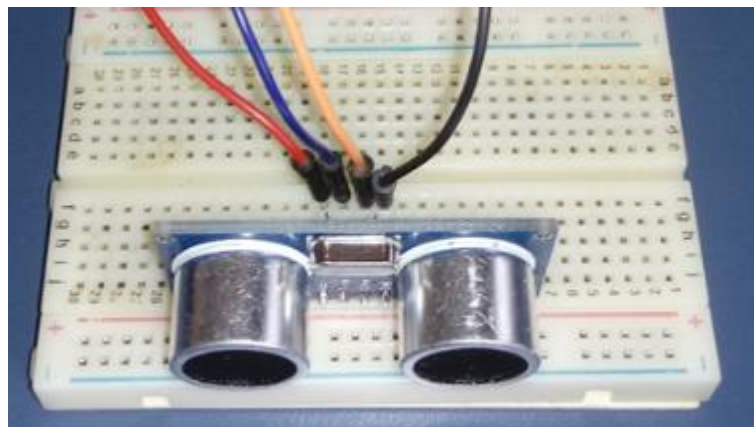
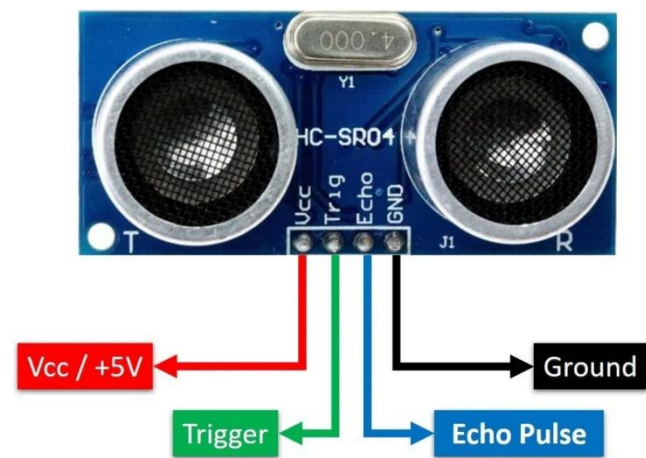
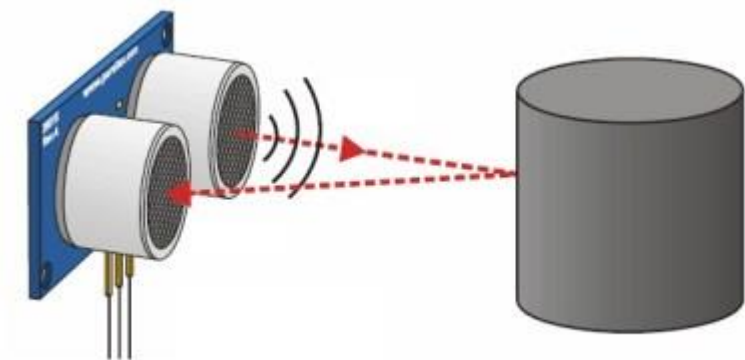


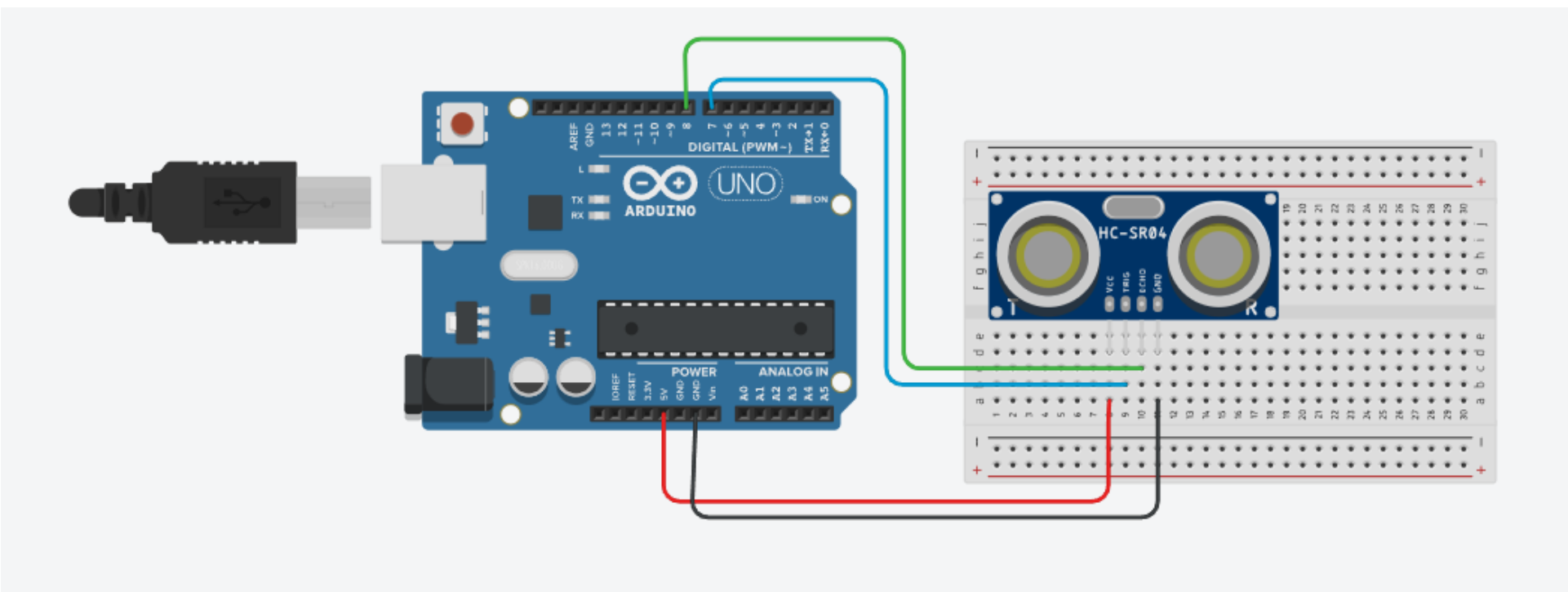
1. Ultrassonico

Sensor de proximidade



Distância de um objeto = ((velocidade do som no ar) x tempo) / 2

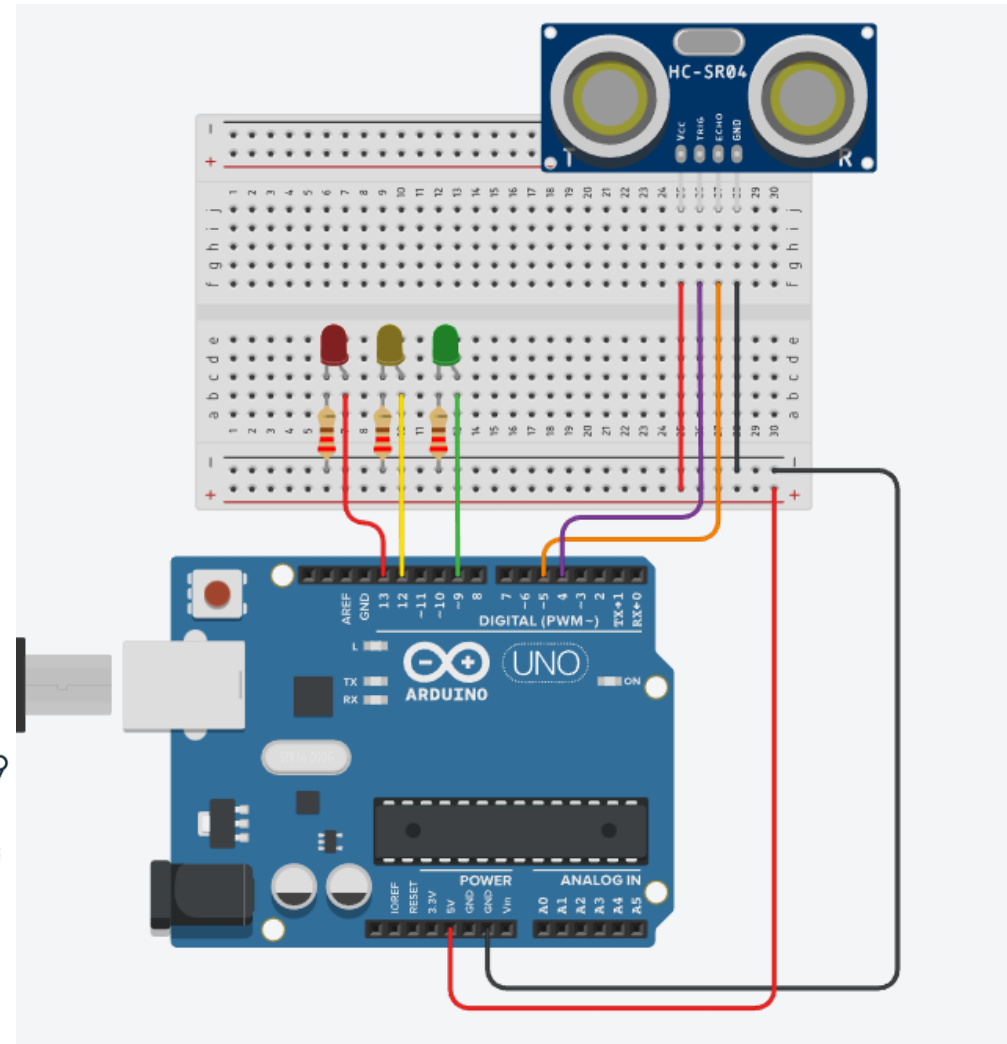




Desafio – Sensor de Proximidade

Material necessário:

- 1 Arduino;
- 1 Sensor Ultrasonico HC-SR04;
- 3 Resistores de 220 ohms
- 1 Led Verde;
- 1 Led Vermelho;
- 1 Led Amarelo;
- 1 Protoboard;
- Jumpers cables.



1. Servo Motor PWM

PWM

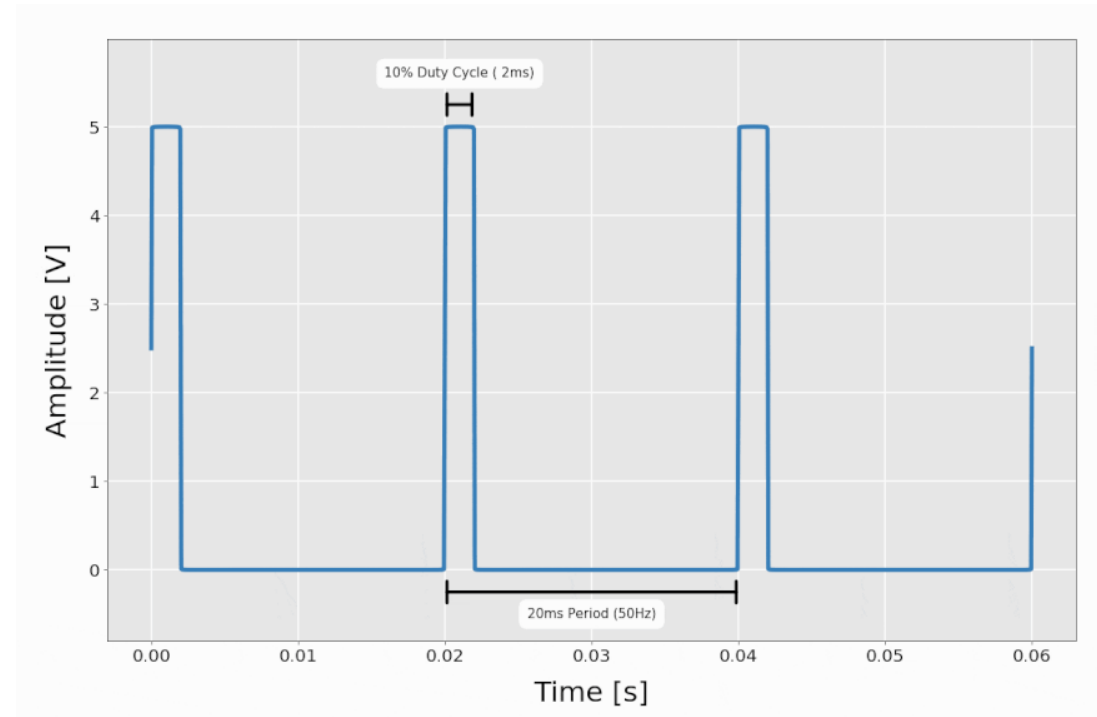
PWM é a sigla para "Pulse Width Modulation", que em português significa "Modulação por Largura de Pulso"

É uma técnica utilizada em eletrônica para controlar a quantidade média de energia fornecida a um dispositivo elétrico.

O tempo de cada pulso é dividido em duas partes:

- O período: tempo completo de um ciclo de pulso
- O ciclo de trabalho: é a fração do período em que o pulso está em nível alto (ligado)

A largura do pulso, ou seja, a duração em que o pulso está em nível alto, é variada para controlar a quantidade média de energia fornecida ao dispositivo.



Fonte: <https://makersportal.com/blog/2020/3/21/raspberry-pi-servo-panning-camera>

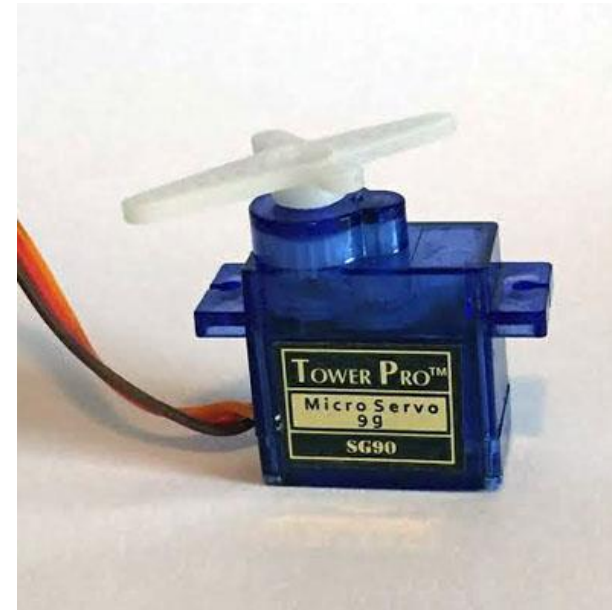
Servo Motor

É tipo especial de motor de corrente contínua (DC) projetado para controlar a posição angular de um eixo de saída com precisão. Possui três componentes principais:

- ❖ um motor de corrente contínua: responsável por fornecer a energia mecânica para girar o eixo de saída do servo motor
- ❖ um circuito de controle interno: responsável por interpretar o sinal de controle enviado pelo Arduino e ajustar a posição do eixo de saída de acordo.
- ❖ um conjunto de engrenagens: usadas para reduzir a velocidade do motor e aumentar o torque, permitindo que o servo motor seja capaz de movimentar cargas com precisão

O sinal PWM é usado para controlar a posição do servo motor. Por exemplo, um pulso com uma largura de 1 milissegundo pode indicar uma posição de 0 graus, enquanto um pulso com uma largura de 2 milissegundos pode indicar uma posição de 180 graus. A variação na largura do pulso permite o controle preciso da posição angular do servo motor.

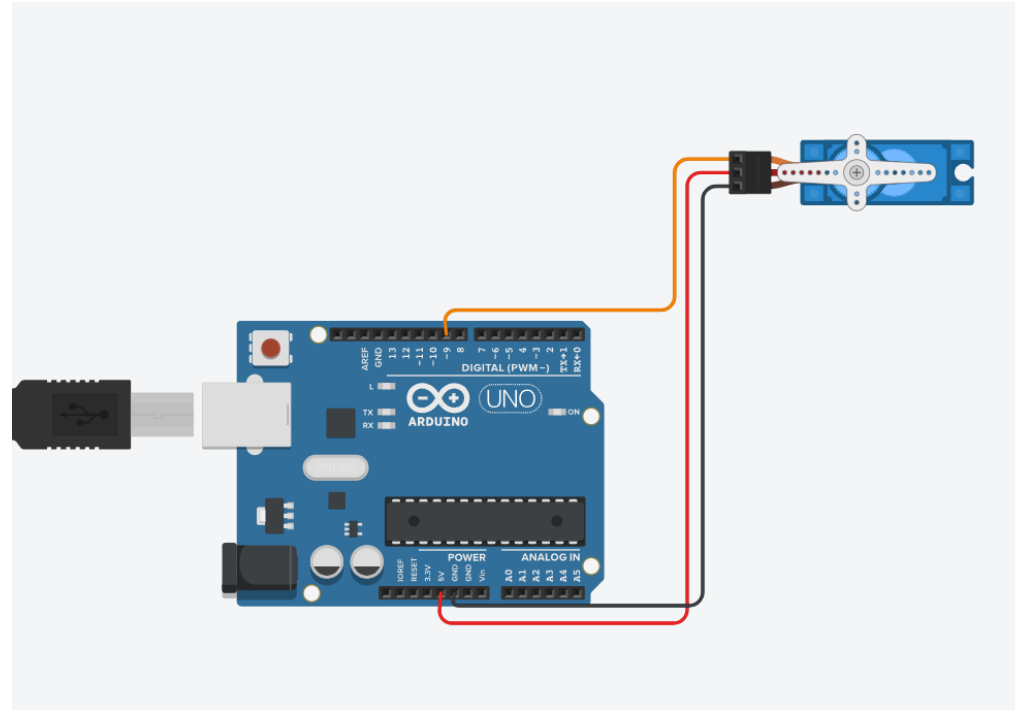
Eles são comumente usados em robótica, automação residencial, modelagem, controle de movimento de câmeras, aeromodelismo e uma ampla gama de outros projetos onde o controle preciso de posição é necessário, devido à sua precisão e facilidade de controle.



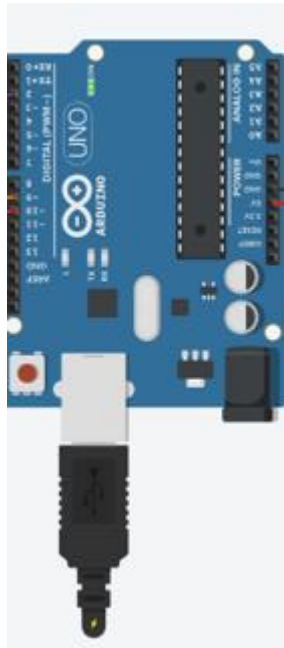
Servo Motor

Material necessário:

- 1 Arduino;
- 1 ServoMotor;
- • Jumpers cables.



DESAFIO





Copyright © 2024 Prof. Yan Coelho

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).