

FIAP

Engenharia de Software

Edge Computing & Computer Systems

Prof. Fabio / Prof. Flavio/ Prof. Lucas / Prof. Yan

Apresentação do professor

FIAP

Yan Gabriel Coelho

Formação:

Graduado em engenharia elétrica – UNINOVE

Pós-graduado em Inteligência Artificial e Aprendizagem de Máquina - UNINOVE

Licenciatura em matemática – UNINOVE (ATUAL)

Experiência como professor:

FIAP (SI e ES)

FIAP ON (SI 4 ANO)

Técnico

ESTADO

OBJETIVO

Etc...

Experiência

Técnico em Hardware

Coordenador de cursos de TI (ATUAL)

Hobbies:

Séries, Cultura Maker, Jogos, PC, entre outros.



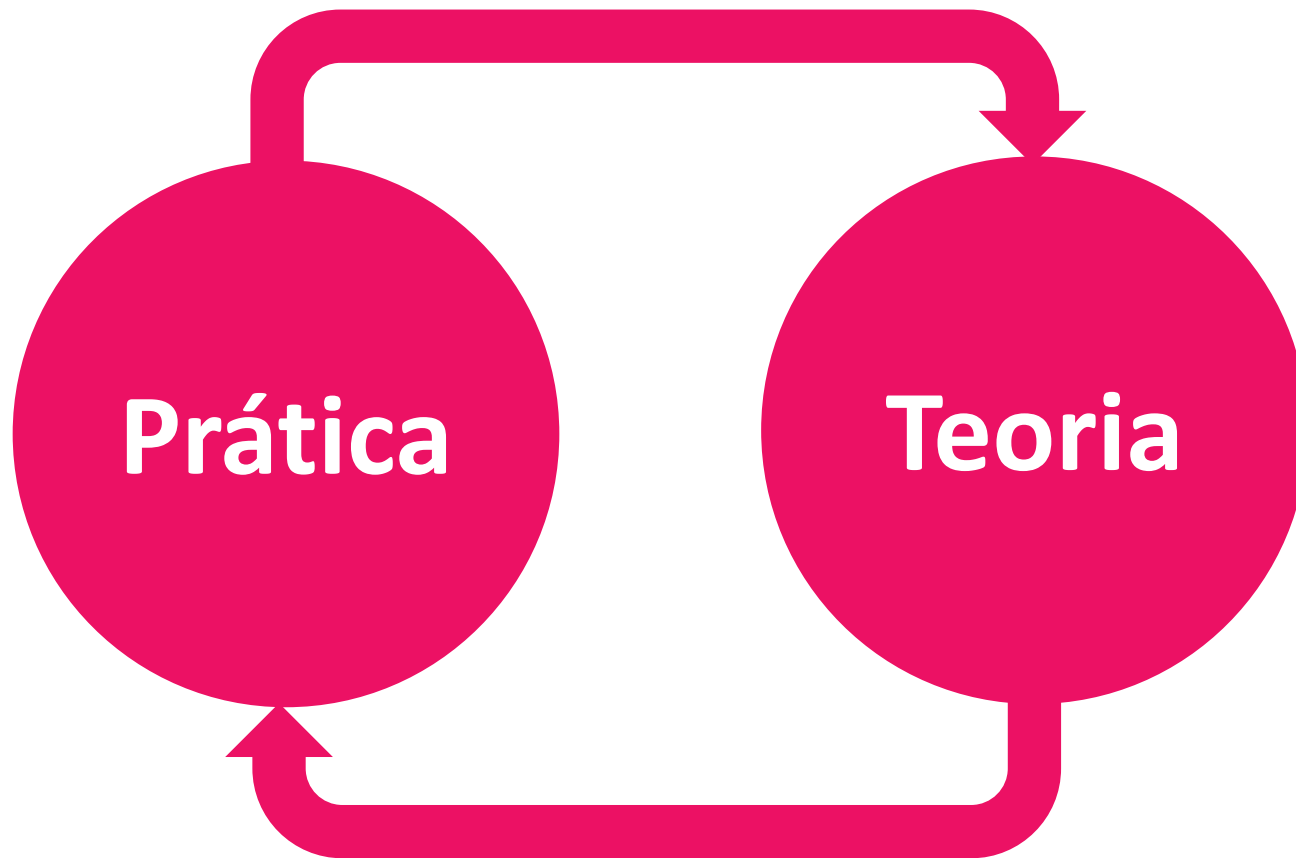
proyan.coelho@fiap.com.br



Microsoft Teams

Dinâmica do curso

- Metodologia baseada em projetos (PBL) e hands-on.



Mão na massa!!!!

1. Introdução a lógica de programação

Lógica de programação

É a técnica de encadear pensamentos para atingir um determinado objetivo.

Como fazer isso? → Algoritmo

“Sequência de ações que permite solucionar um determinado problema”



Lógica de programação

Quais são os passos necessários para se trocar uma lâmpada queimada?

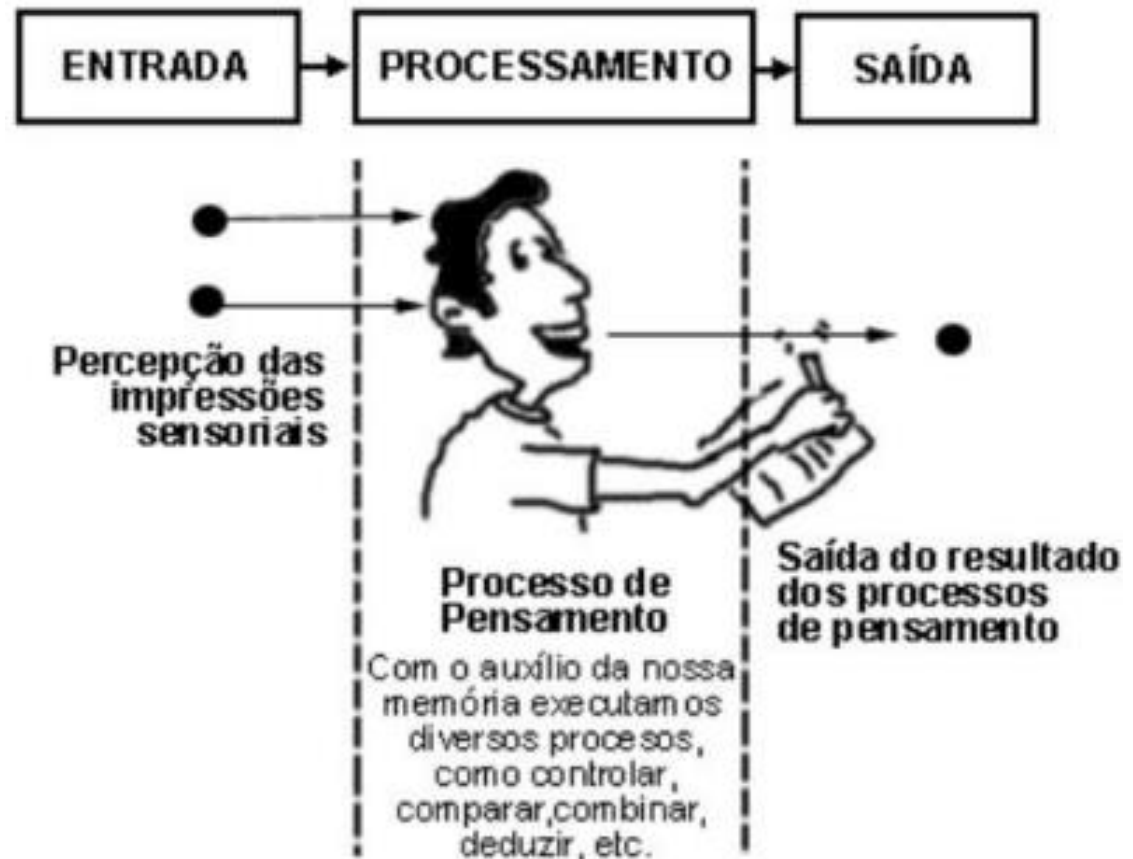
- 1 – Comprar uma lâmpada nova;
- 2 – Pegar uma escada;
- 3 – Desligar o interruptor;
- 4 – Pegar a lâmpada nova;
- 5 – Subir na escada;
- 6 – Remover a lâmpada queimada;
- 7 – Instalar a lâmpada nova;
- 8 – Descer da escada;
- 9 – Descartar a lâmpada queimada;
- 10 – Acionar o interruptor;



Lógica de programação

Processo básico de um algoritmo:

- Entrada de dados
- Processamento de dados
- Saída de dados



Lógica de programação: fluxogramas

Diagrama de blocos

Forma padronizada, organizada e eficaz para representar os passos lógicos de um determinado processo

Facilita a visualização dos passos de um processamento

Início ou fim

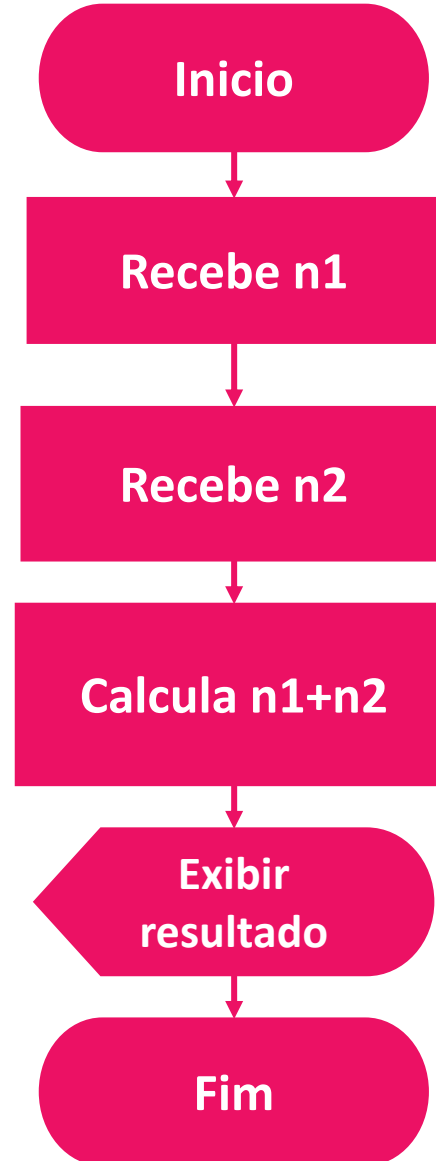
Processamento geral. Neste curso, será usado também para entrada ou saída de dados

Teste condicional

Exibir

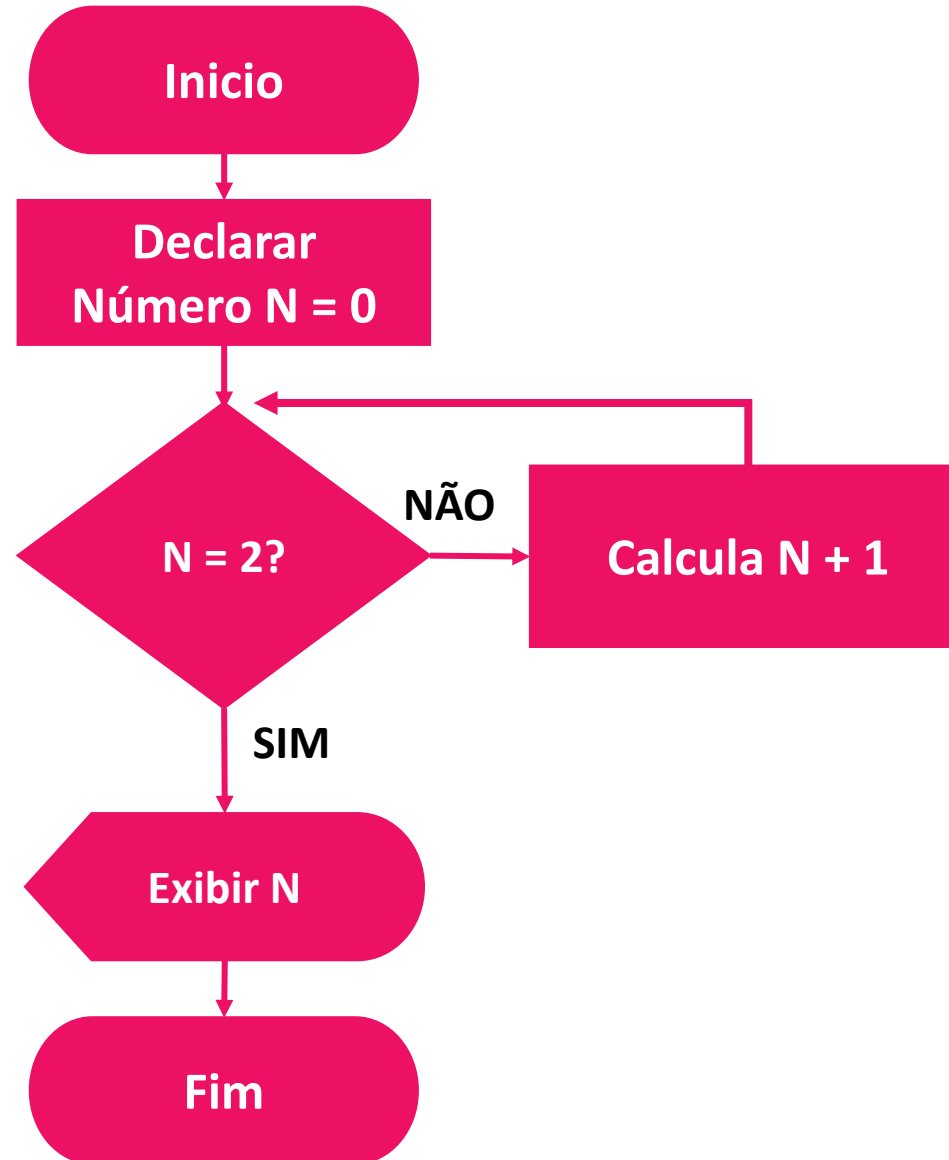
Lógica de programação: fluxogramas

Soma de dois
números



Lógica de programação: fluxogramas

Contar até dois



Lógica de programação: variáveis e constantes

“Espaços de memória reservados para armazenar um tipo específico de dado”

Constante: Espaço de memória com valor fixo ao longo da execução do programa

Variável: Espaço de memória com valor que pode ser alterado ao longo da execução do programa

Exemplo: Programa que calcula a média entre 5 números: N1, N2, N3, N4 e N5.

Constante: 5 -> quantidade de variáveis, não altera

Variáveis: N1, N2, N3, N4 e N5 -> valores alteráveis

Lógica de programação: variáveis e constantes

Tipos de dados em variáveis ou constantes

Numérico: Inteiros (int) ou reais (float), podendo ser usado para cálculos matemáticos;

Caracteres (char): Símbolos que não contêm números, como nomes;

Alfanumérico: combinação de números e letras, podendo conter só letras ou só números, mas não pode executar operações matemáticas;

Lógica / booleana: Verdadeiro ou falso;

Obs.: Um conjunto de caracteres é chamado de string;

“American Standard Code for Information Interchange” - ASCII

"Código Padrão Americano para o Intercâmbio de Informação"

Tabela ASCII

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Lógica de programação: operadores aritméticos

Usados para obtenção de dados numéricos

Operação	Símbolo
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Resto (divisão)	%

Ordem de execução:

1º Parênteses ();

2º Multiplicação ou divisão;

3º Soma ou subtração;

Lógica de programação: operadores relacionais

Usados para comparações (decisões), retornando valores lógicos.

Descrição	Símbolo
igual	==
Diferente	!=
Menor	<
Maior	>
Menor ou igual	<=
Maior ou igual	>=

Lógica de programação: operadores lógicos

Usados para comparações (decisões), retornando valores lógicos

Descrição	Símbolo
AND - E	&&
OR - OU	
Negação - Inversor	!

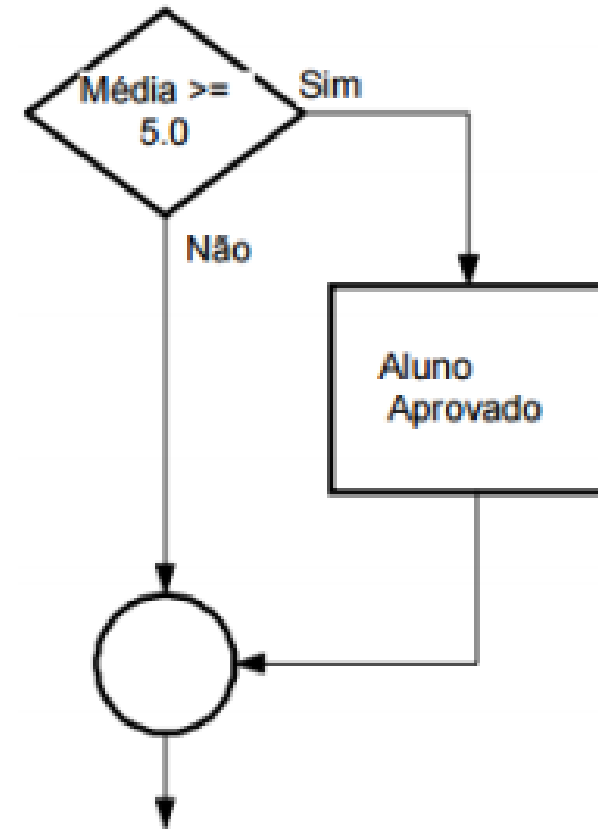
Os operados relacionais booleanos retornam valores de acordo com a respectiva tabela verdade.

Lógica de programação: estruturas de decisão

“SE x ENTÃO y” → “IF x THEN y”

Exemplo de algoritmo:

- 1 – **SE** o aluno tiver média maior que 5.0
- 2 – **ENTÃO** o aluno está aprovado

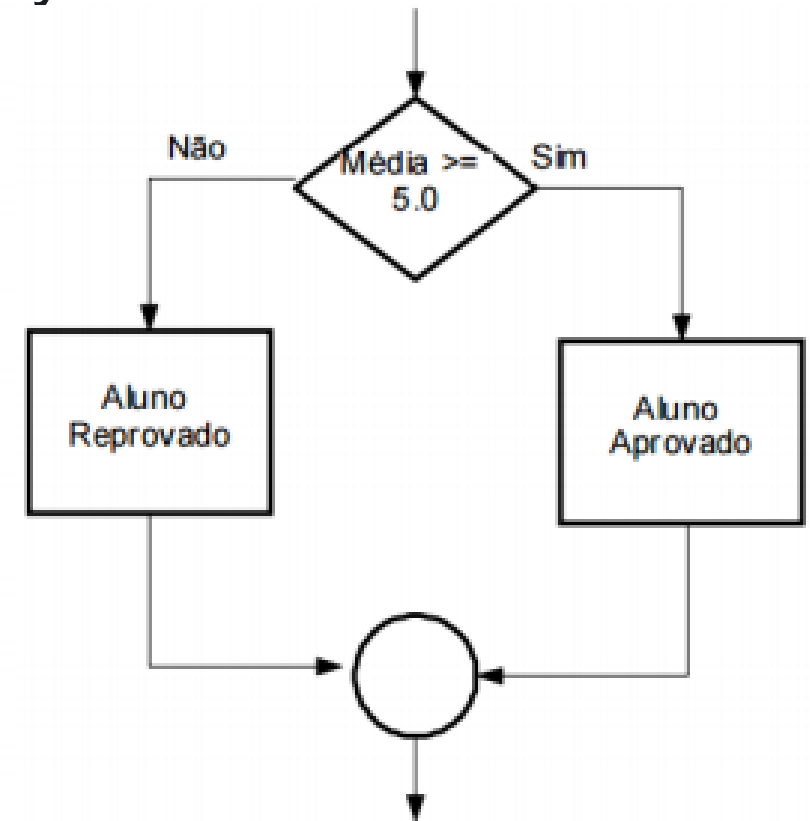


Lógica de programação: estruturas de decisão

“SE x ENTÃO y SENÃO z” → “IF x THEN y ELSE z”

Exemplo de algoritmo:

- 1 – **SE** o aluno tiver média maior que 5.0
- 2 – **ENTÃO** o aluno está aprovado
- 3 – **SENÃO** o aluno está reprovado



Lógica de programação: estruturas de decisão

“SE x ENTÃO y SENÃO z” → “IF x THEN y ELSE z”

Outro exemplo de algoritmo:

1 – **SE** o aluno tiver média maior que 5.0

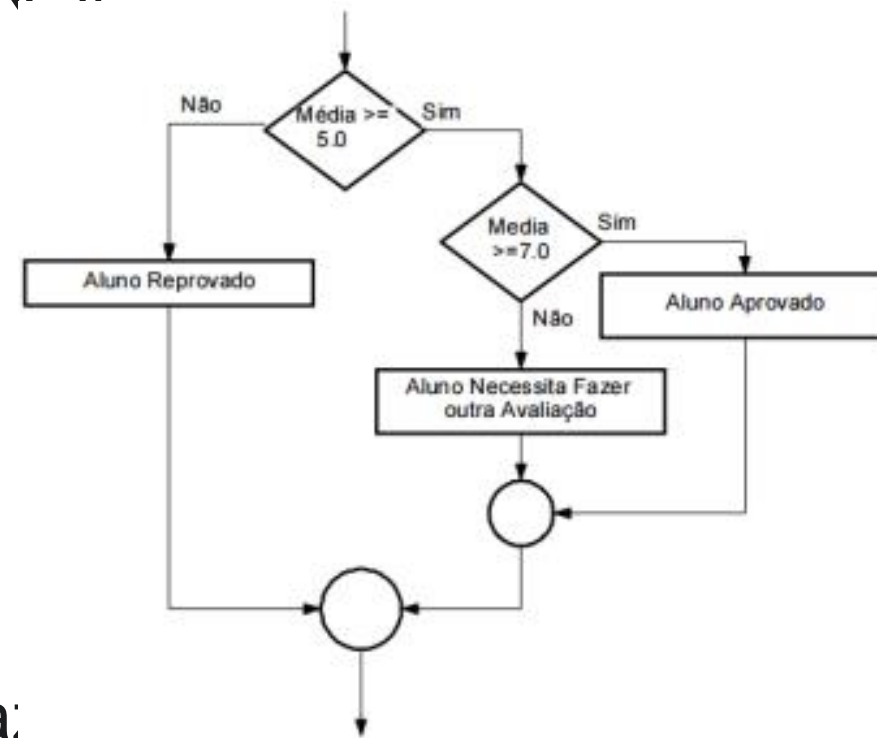
2 – **ENTÃO** faça:

3 – SE sua nota for maior ou igual a 7

4 – ENTÃO o aluno está aprovado

5 – SENÃO o aluno deve fazer recuperação

6 – **SENÃO** o aluno está reprovado



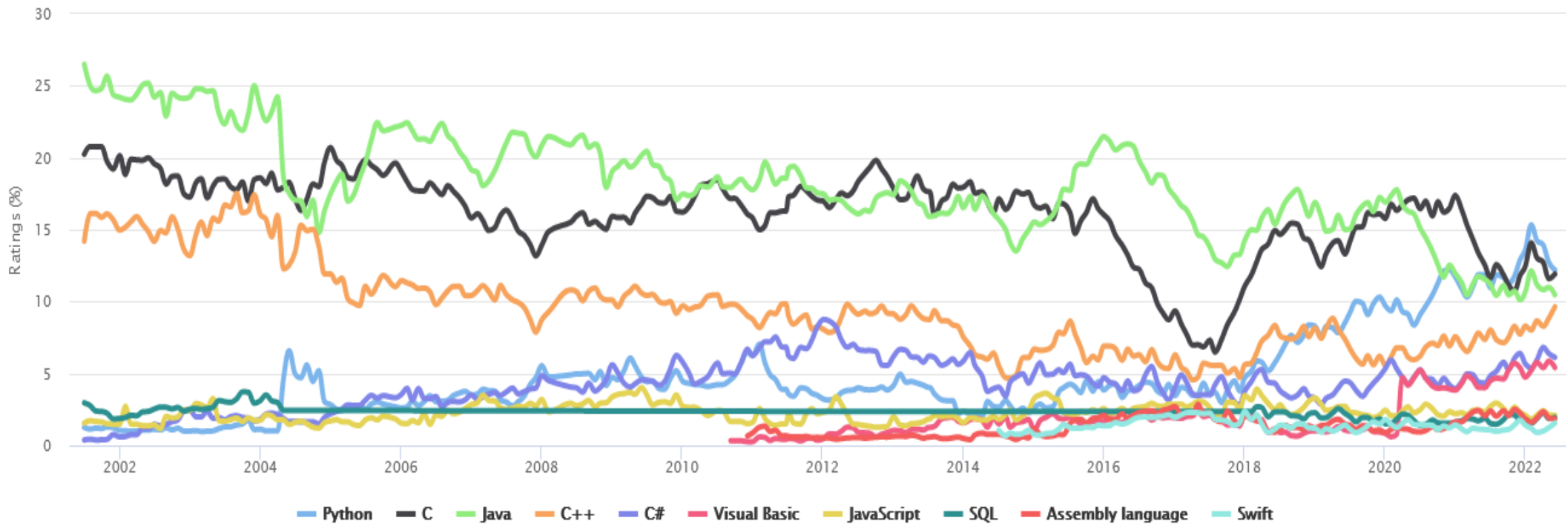
Linguagem de programação

A **linguagem de programação** é um método padronizado, com a finalidade de converter um algoritmo em instruções para execução do comportamento desejado em um computador.

Popularização das Linguagens de Programação

TIOBE Programming Community Index

Source: www.tiobe.com



Funcionalidade de projeto

- C incorpora o controle de funcionalidades desejáveis na teoria e na prática, na área de computação;
- Sua construção à torna natural para programação estruturada e desenvolvimento modular;
- Como resultado é um programa mais confiável e inteligível.

Eficiência

- Aproveita as capacidades das máquinas atuais;
- Programas em C tendem a ser compactos e rápidos;
- De fato, C exibe algum controle fino usualmente associado com a linguagem Assembly.
 - Linguagem Assembly é um conjunto de instruções internas, específicas para cada microcontrolador.

Portabilidade

- Programas C escritos em um sistema pode funcionar em outros sistemas com pequenas ou nenhuma modificação;
- Compiladores C estão disponíveis para aproximadamente 40 sistemas, desde microcontroladores de 8 bits aos super computadores .

Flexibilidade

- C é poderosa e flexível;
- A maioria dos sistemas operacionais, como o Unix e Windows, é escrito em C;
- Muitos compiladores e interpretadores para outras linguagens, como FORTRAN, Perl, Python, Pascal, LISP, Logo, Basic e Matlab, foram escritas em C;
- C também é utilizado para resolução de problemas de física e engenharia.

Orientado ao Programador

C é orientado às necessidades do programador;

- Permite o acesso ao HW, e permite a manipulação de bits individuais na memória;
- Possui um grande número de operadores.

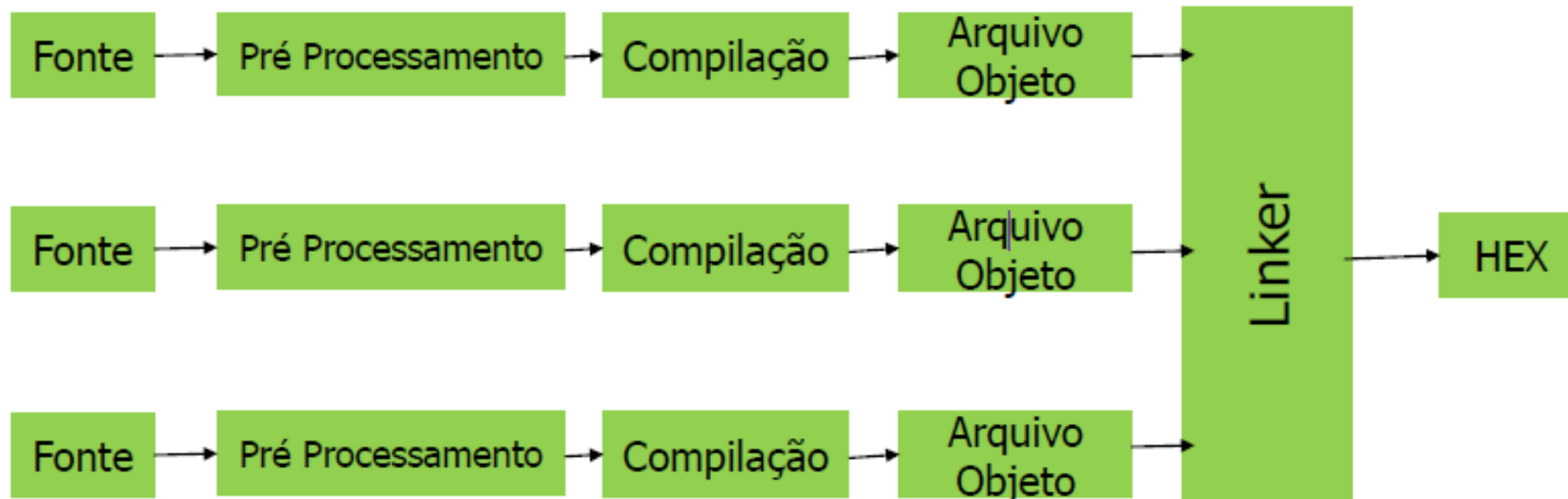
Orientado ao Programador

Essa flexibilidade é uma vantagem e um perigo;

- A vantagem é que muitas tarefas, como a conversão de dados, são mais simples;
- O perigo é que com C, é possível cometer erros que são impossíveis de cometer em algumas linguagens.

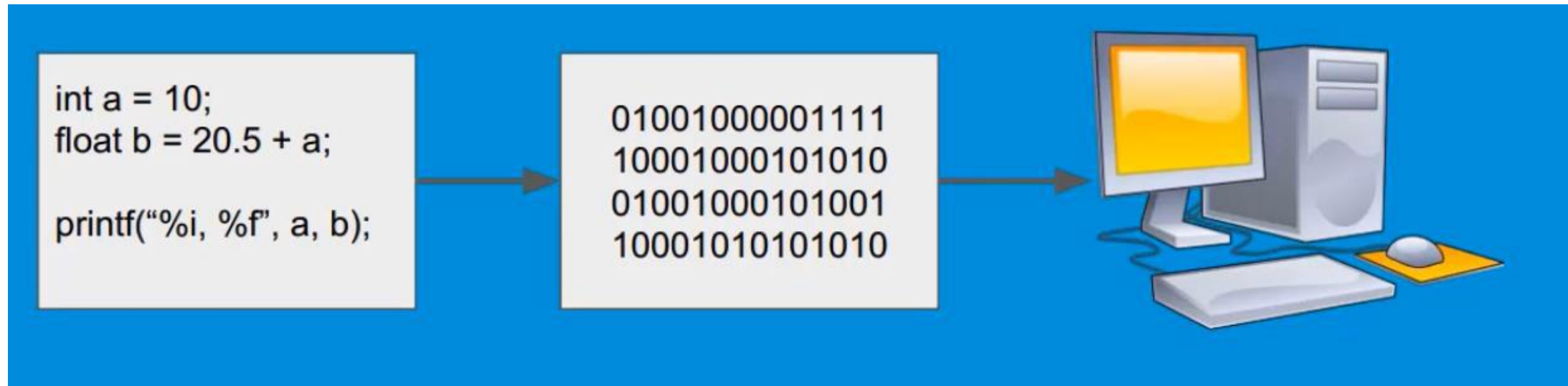
Arquivos Objeto, Executáveis e Bibliotecas

O processo de conversão dos arquivos fonte em executável, é dividido em Pré-Processamento, Compilação e Linkagem;



Arquivos Objeto, Executáveis e Bibliotecas

O resultado final é um arquivo em hexadecimal com os comandos em hexadecimal para o dispositivo executar o algoritmo desejado



3. 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



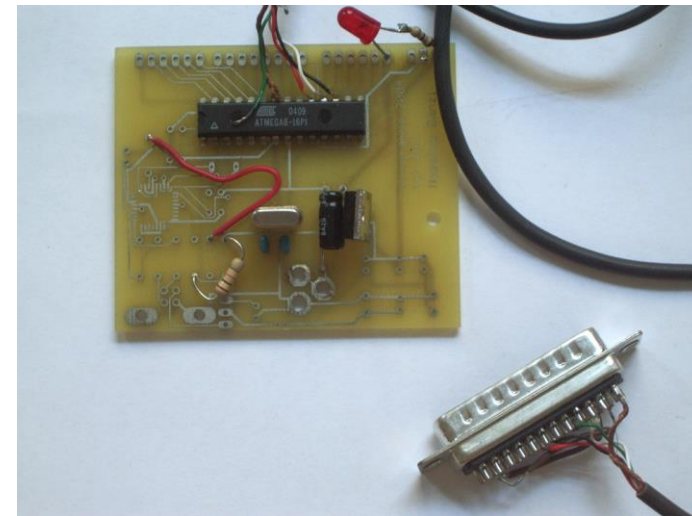
- 1 - Conector USB para o cabo tipo AB
- 2 - Botão de reset
- 3 - Pinos de entrada e saída digital e PWM
- 4 - LED verde de placa ligada
- 5 - LED laranja conectado ao pin13
- 6 - ATmega encarregado da comunicação com o computador
- 7 - LED TX (transmissor) e RX (receptor) da comunicação serial
- 8 - 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



The screenshot shows the Arduino IDE download page. The navigation bar at the top has the Arduino logo on the left and links for HOME, STORE, SOFTWARE (highlighted with a yellow circle), EDU, RESOURCES, COMMUNITY, and HELP. On the right of the bar are search, cart, and SIGN IN icons. The main heading is "Download the Arduino IDE". Below this, the page is divided into two main sections. The left section features the Arduino logo and the text "ARDUINO 1.8.9". It describes the IDE as open-source, easy to use, and compatible with Windows, Mac OS X, and Linux. It also mentions that the software can be used with any Arduino board and refers to the "Getting Started" page for installation instructions. The right section, which has a teal background, lists the download options: "Windows Installer, for Windows XP and up" and "Windows ZIP file for non admin install" (both highlighted with a yellow circle), "Windows app" (which requires Win 8.1 or 10 and includes a "Get" button with the Windows logo), "Mac OS X 10.8 Mountain Lion or newer", "Linux 32 bits", "Linux 64 bits", "Linux ARM 32 bits", and "Linux ARM 64 bits". At the bottom of this section are links for "Release Notes", "Source Code", and "Checksums (sha512)".

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

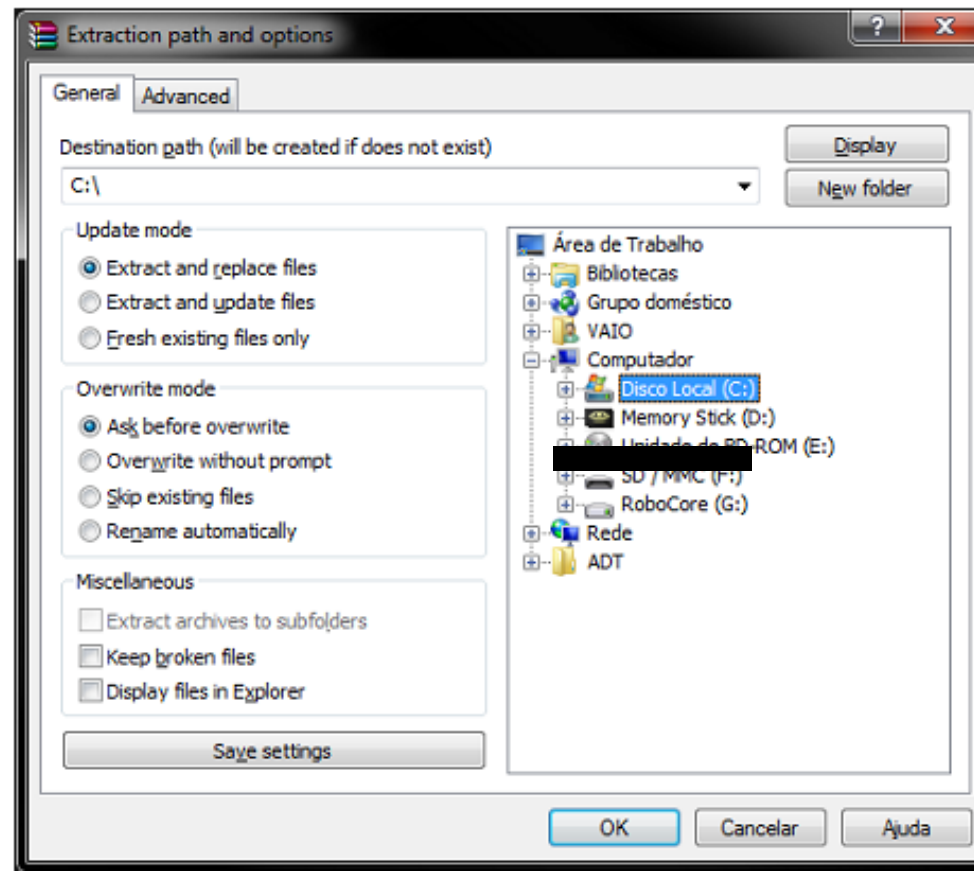
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

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

Ambiente de programação

- Ambiente integrado de Desenvolvimento (IDE)

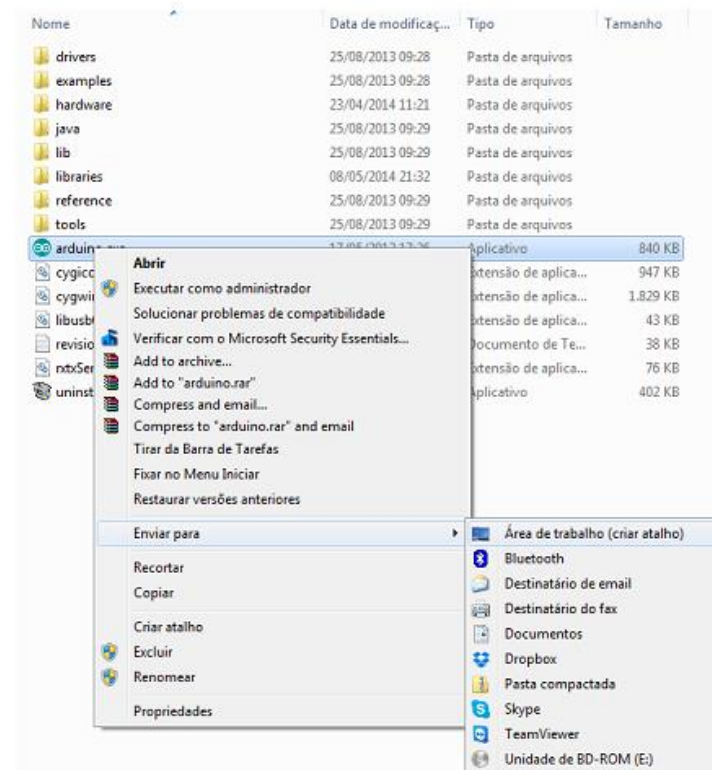
Quando finalizar o download, descompacte a pasta no diretório: **C:** conforme apresentado na figura abaixo.



Ambiente de programação

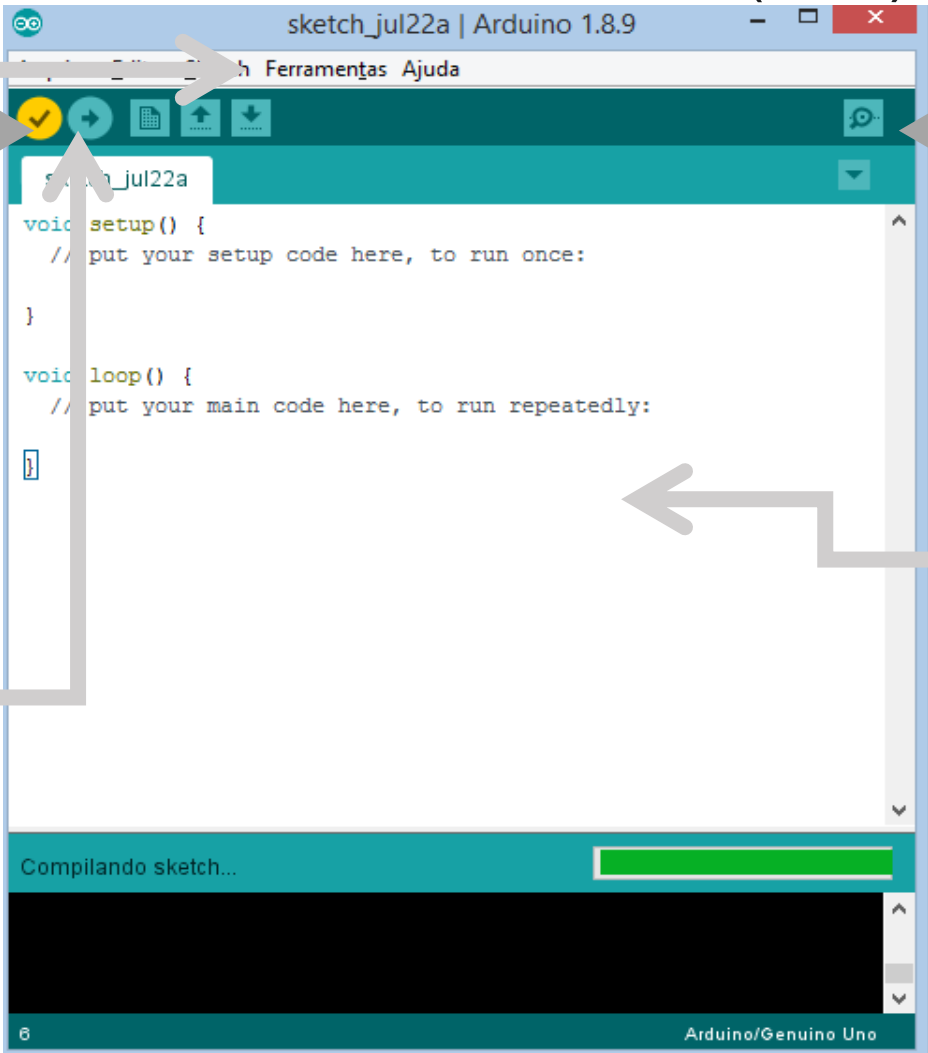
- Ambiente integrado de Desenvolvimento (IDE)

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)



The image shows the Arduino IDE interface with several components labeled:

- Tools** - seleciona o tipo de Arduino e a porta COM (points to the Tools menu)
- Verify** – compila a programação (points to the Verify button)
- Upload** – envia a programação para o Arduino (points to the Upload button)
- Serial Monitor** – monitor de dados (points to the Serial Monitor icon)
- Sketch** – palco da programação (points to the sketch editor area)

The sketch editor area contains the following code:

```
sketch_jul22a
void setup() {
  // put your setup code here, to run once:
}

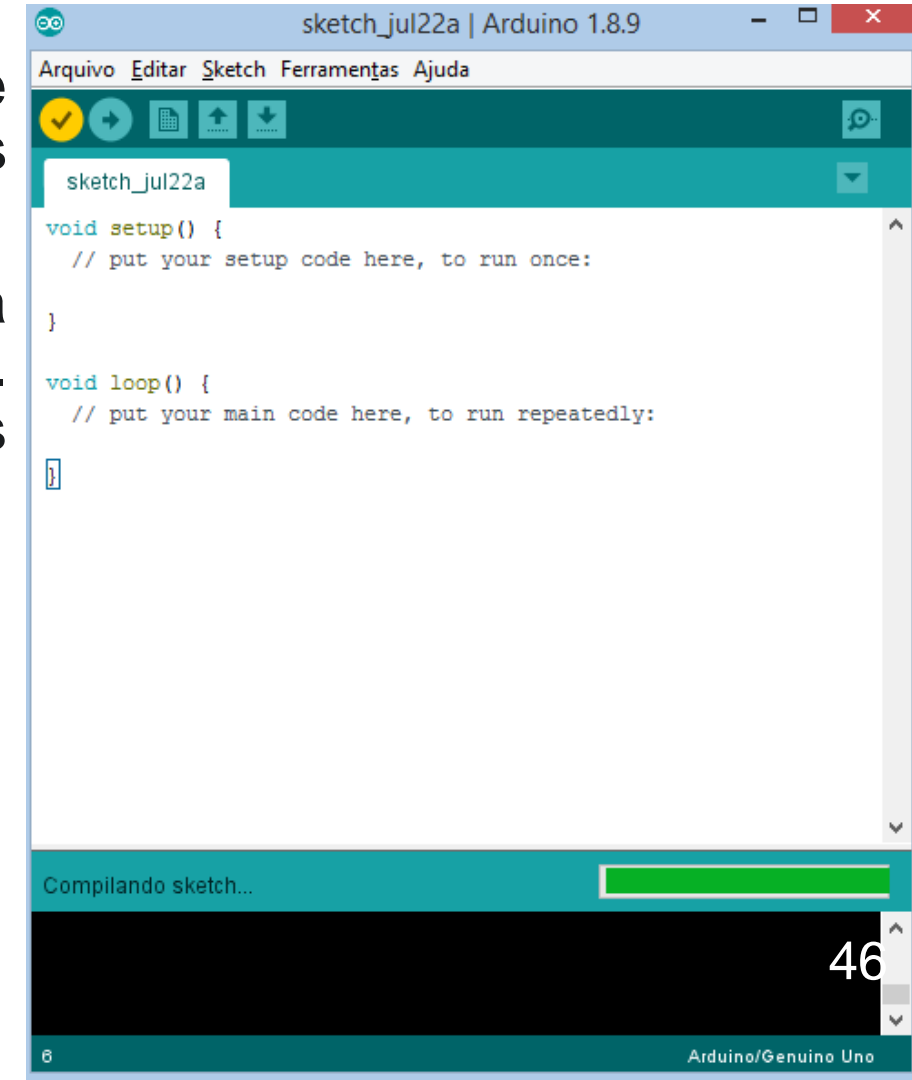
void loop() {
  // put your main code here, to run repeatedly:
}
```

The bottom status bar shows "Compilando sketch..." and "Arduino/Genuino Uno".

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.

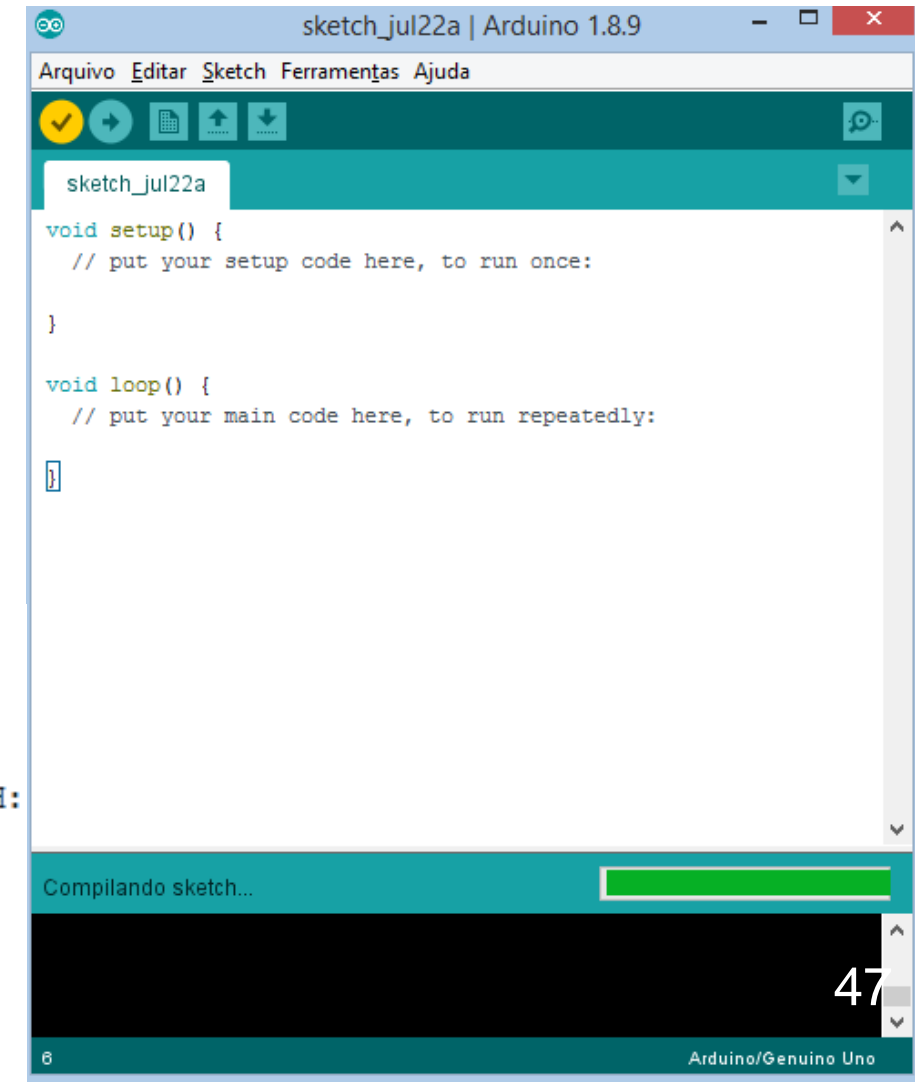
```
void setup() {  
    // initialize the LED pin as an output:  
    pinMode(ledPin, OUTPUT);  
    // initialize the pushbutton pin as an input:  
    pinMode(buttonPin, INPUT);  
}
```



Ambiente de programação

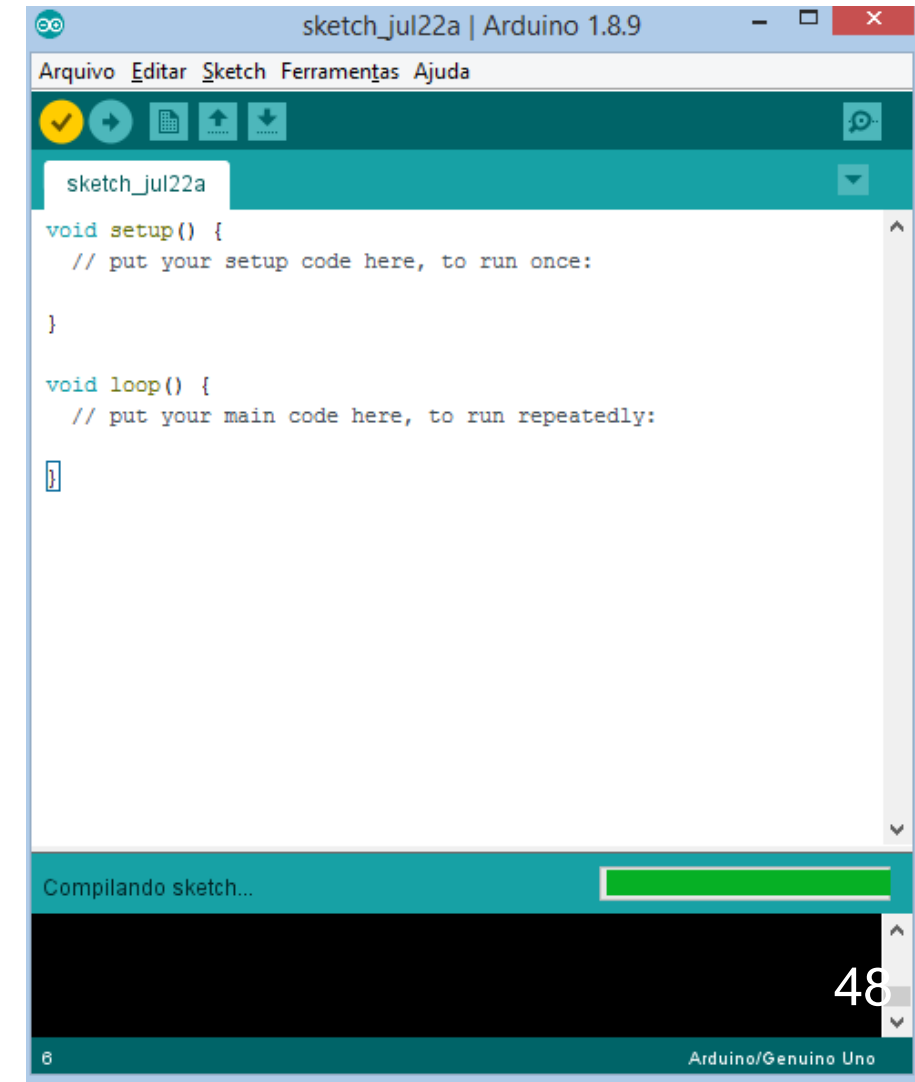
- A função **loop()** irá executar continuamente as instruções que estão lá até que outro sketch seja carregado na memória “flash” do Arduino.
- É importante notar que no Arduino é possível armazenar e executar um sketch por vez, desta forma, sempre quando transferimos um sketch esse irá substituir o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

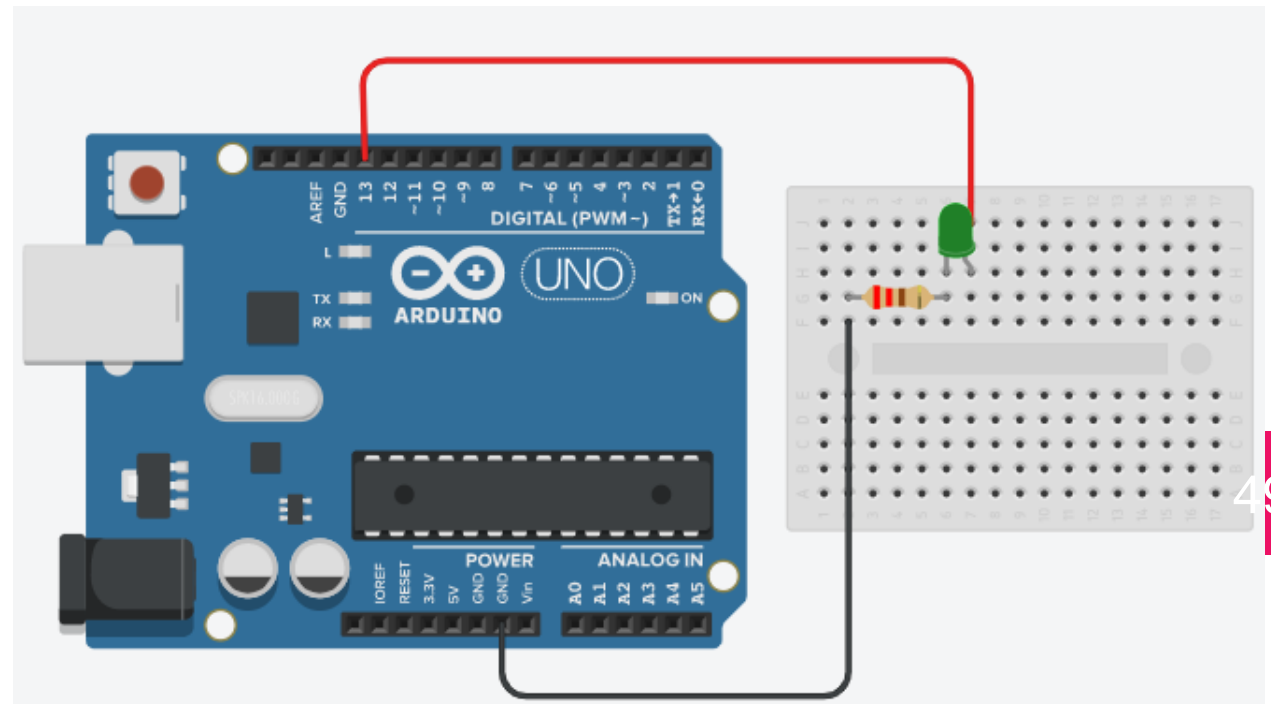
- Também observe que como o sketch fica armazenado na memória “flash”, que é permanente, mesmo quando desligamos o Arduino, o programa continua armazenado e irá entrar novamente em execução quando o Arduino for ligado novamente.
- Note também que, nestas duas funções, a palavra reservada **void** indica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



Mãos na massa! Piscar um LED

Um nível 1 (HIGH) colocado no pino irá acender o LED durante 2s, enquanto um nível 0 (LOW) vai apagar o LED.

- Material necessário:
- 1 Arduino;
- 1 Resistor de 300 ohms (laranja, preto, marrom);
- 1 Led (qualquer cor);
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Piscar um LED

- Programa

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

void loop()
{
  digitalWrite(13, HIGH); // envia sinal 1 para o pino
  delay(2000); // aguarda 2 segundos
  digitalWrite(13, LOW); // envia sinal 0 para o pino
  delay(1000); // aguarda 2 segundos
}
```

Após **salvar** o sketch (programa), faça a **compilação** e, em seguida, conecte o Arduino à porta USB do computador. Finalizando, pressione o botão **Carregar** (Transferir) para fazer a transferência do sketch para o Arduino.



Laboratório – TinkerCAD

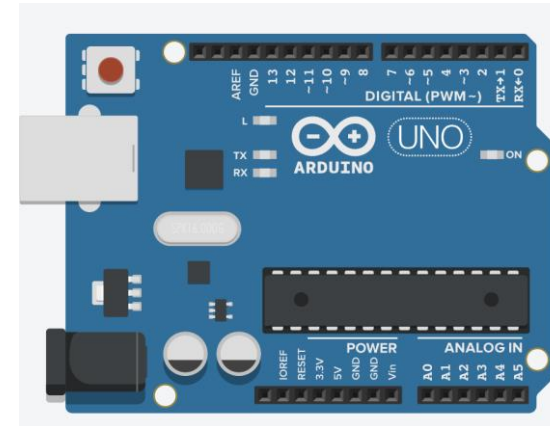
O Objetivo deste laboratório é conhecer o aplicativo

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!

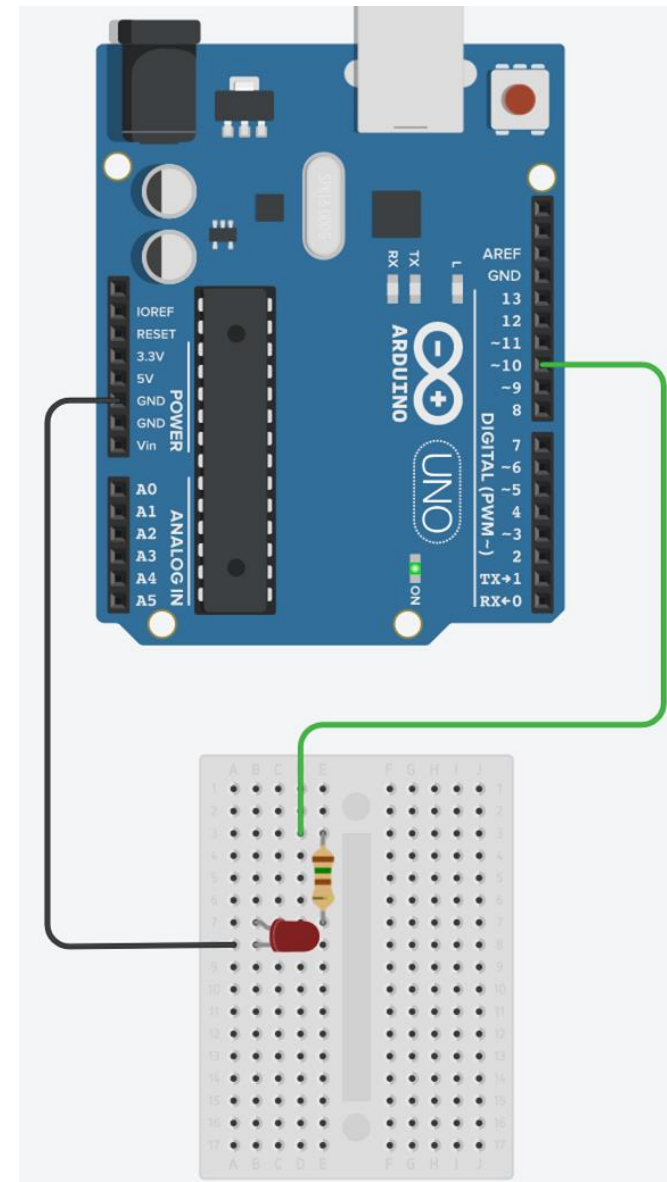
Laboratório – Piscando LED

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

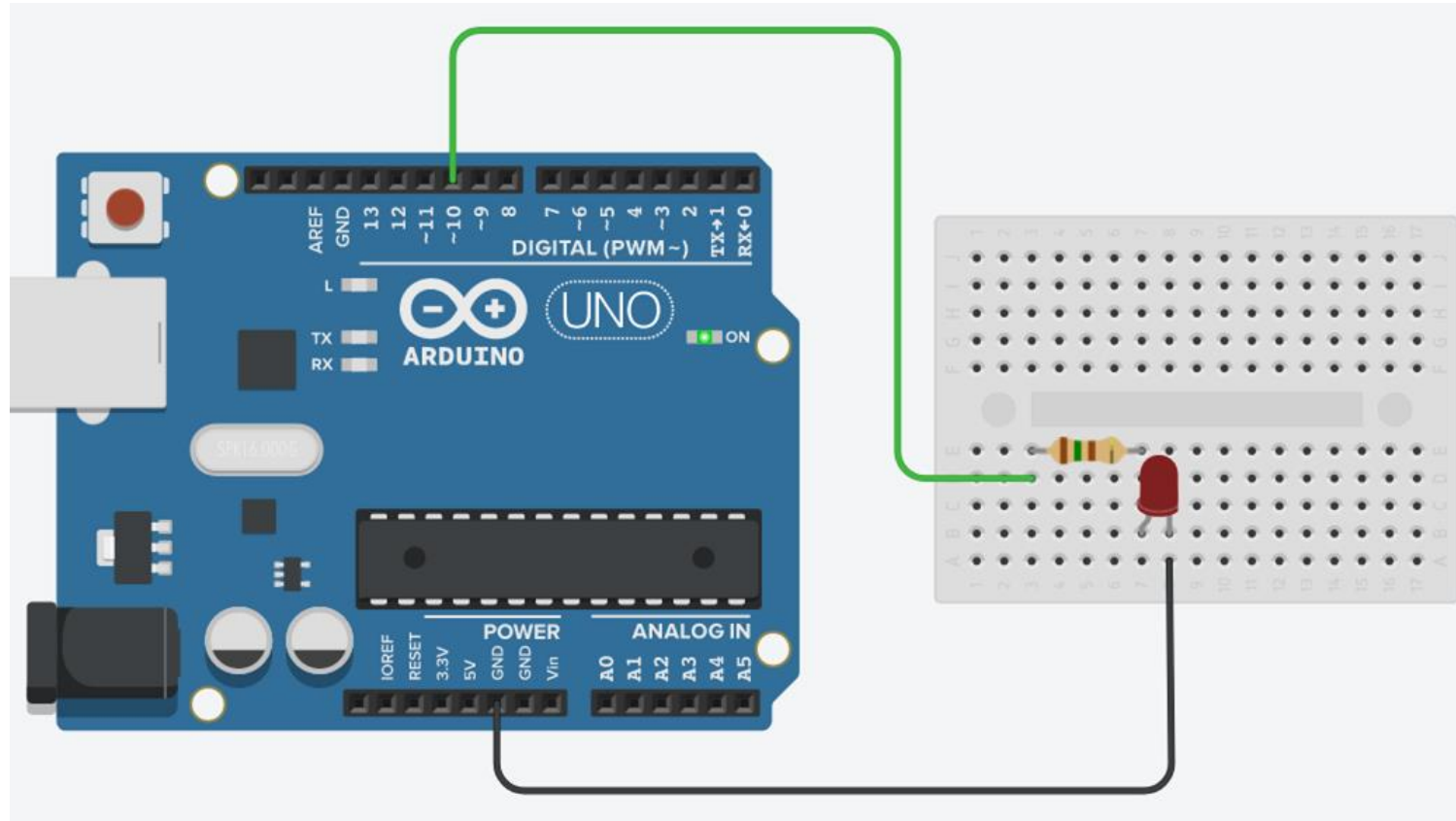
Nesse projeto vamos conhecer a interface de programação do Arduino e entender um poquinho como o hardware de prototipagem funciona.

Material necessário:

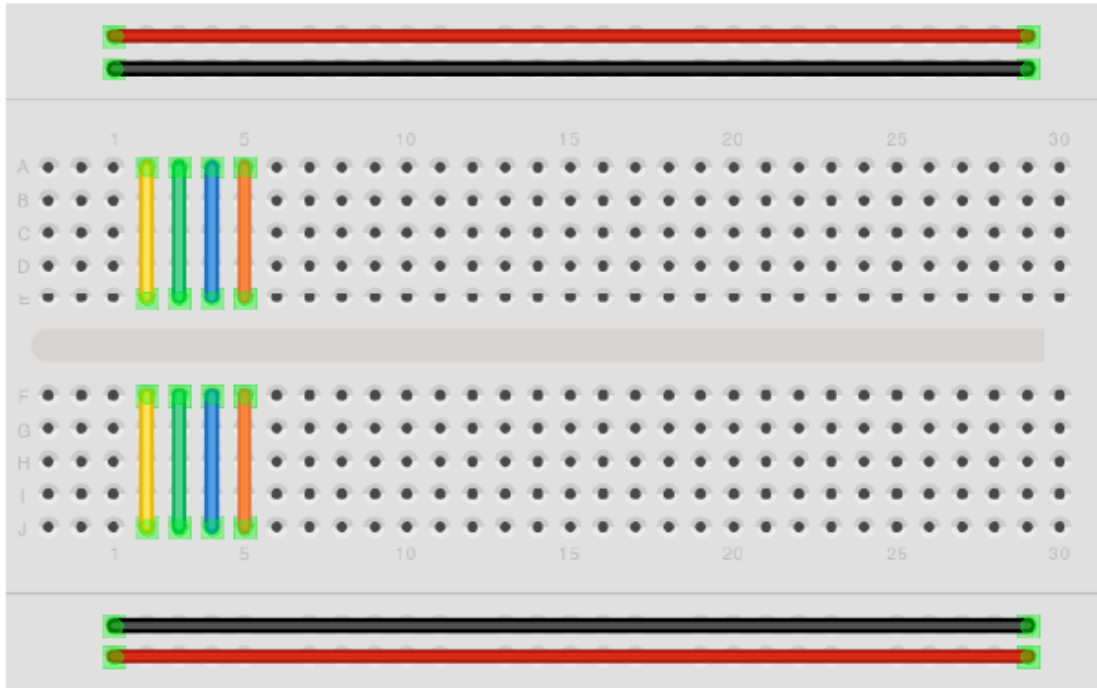
- 1 Arduino;
- 1 Resistor de 150;
- 1 Led (qualquer cor);
- 1 Protoboard;
- Jumpers cables.



Conhecendo o Hardware



Conhecendo o Hardware – Protoboard – Matriz de Contatos Elétricos



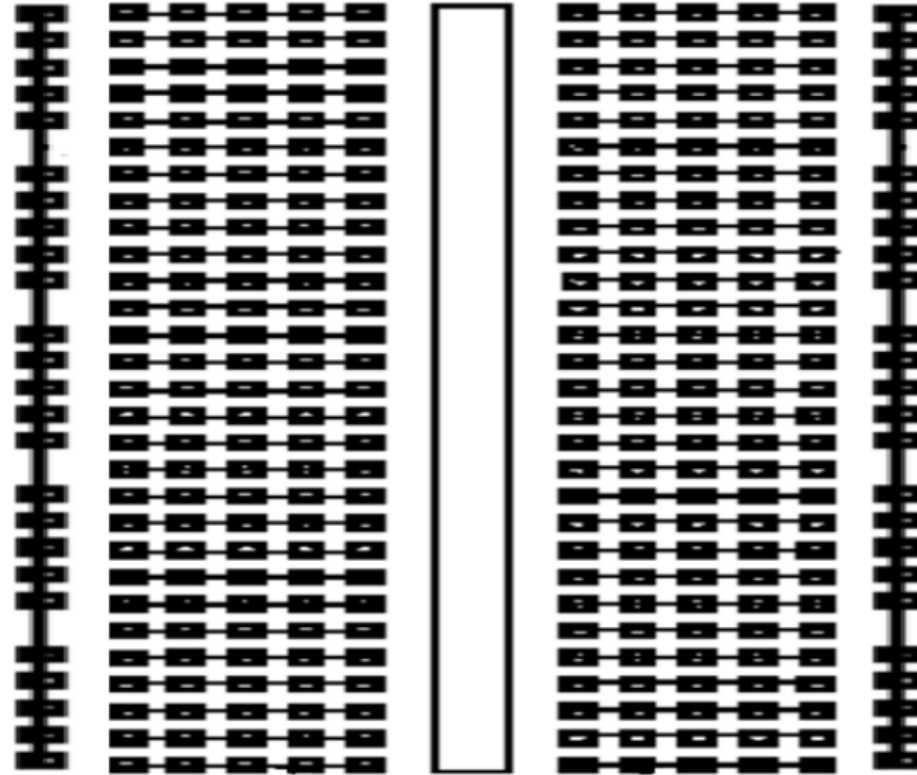
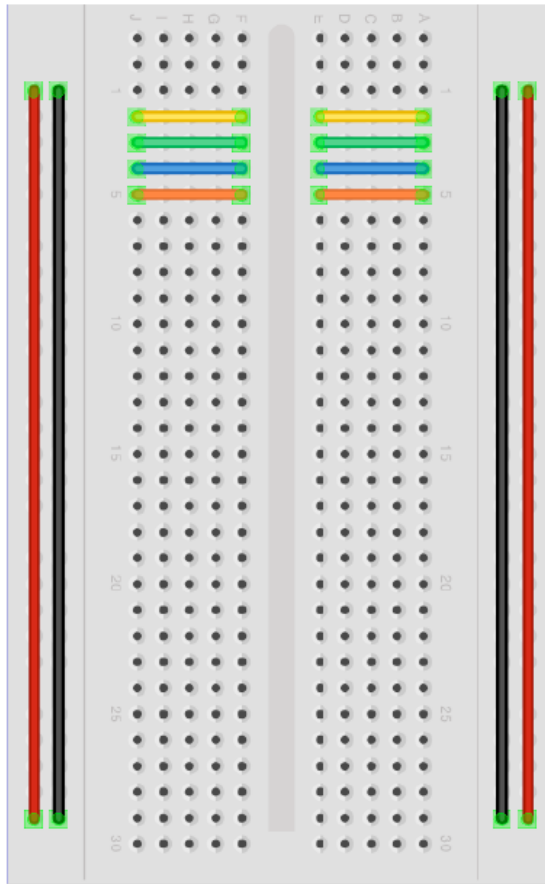
A linha **Vermelha** é toda interligada e serve para ligar o **Positivo** da fonte de alimentação: VCC, VDD, 3.3V, 5V, 12V, +

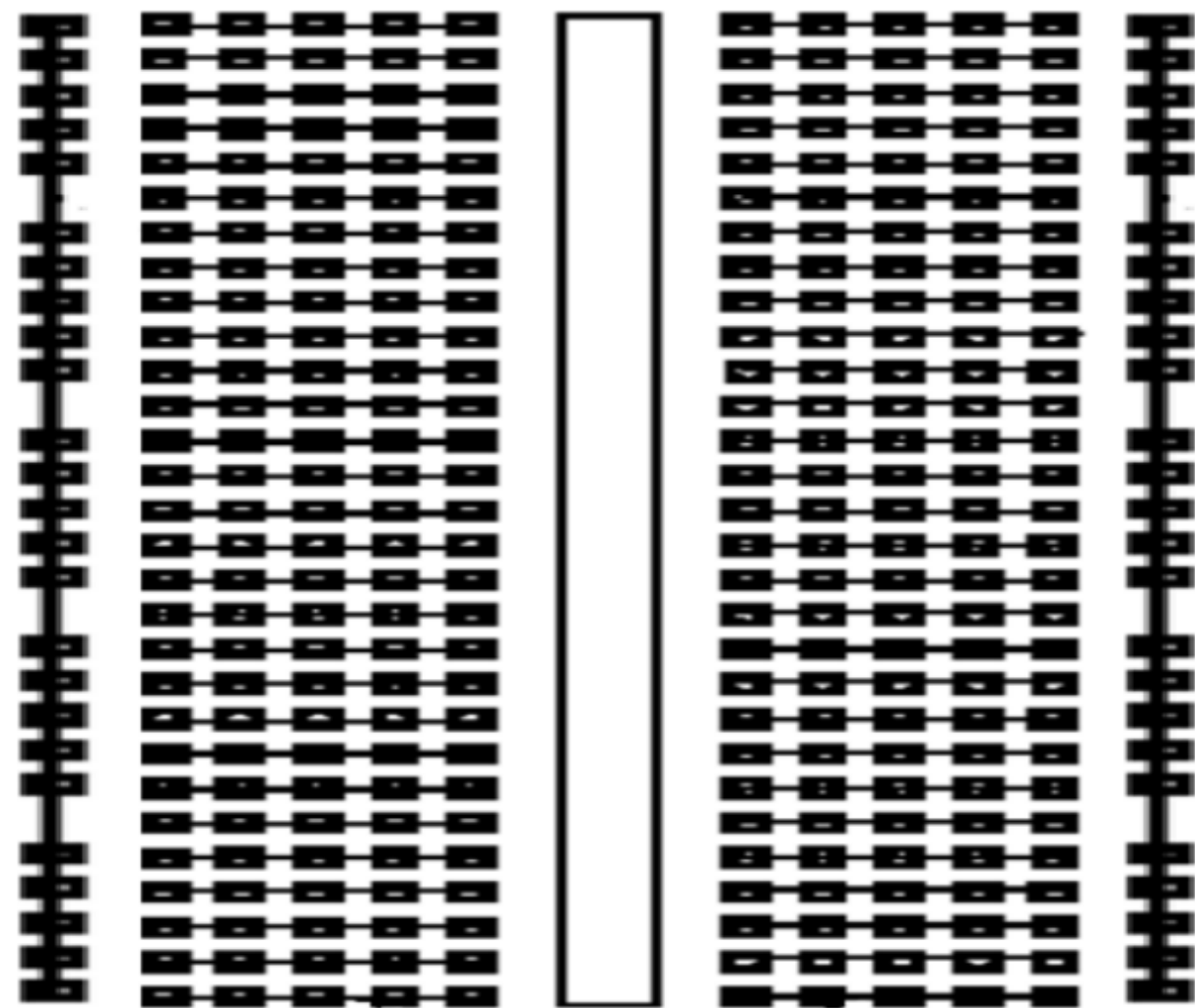
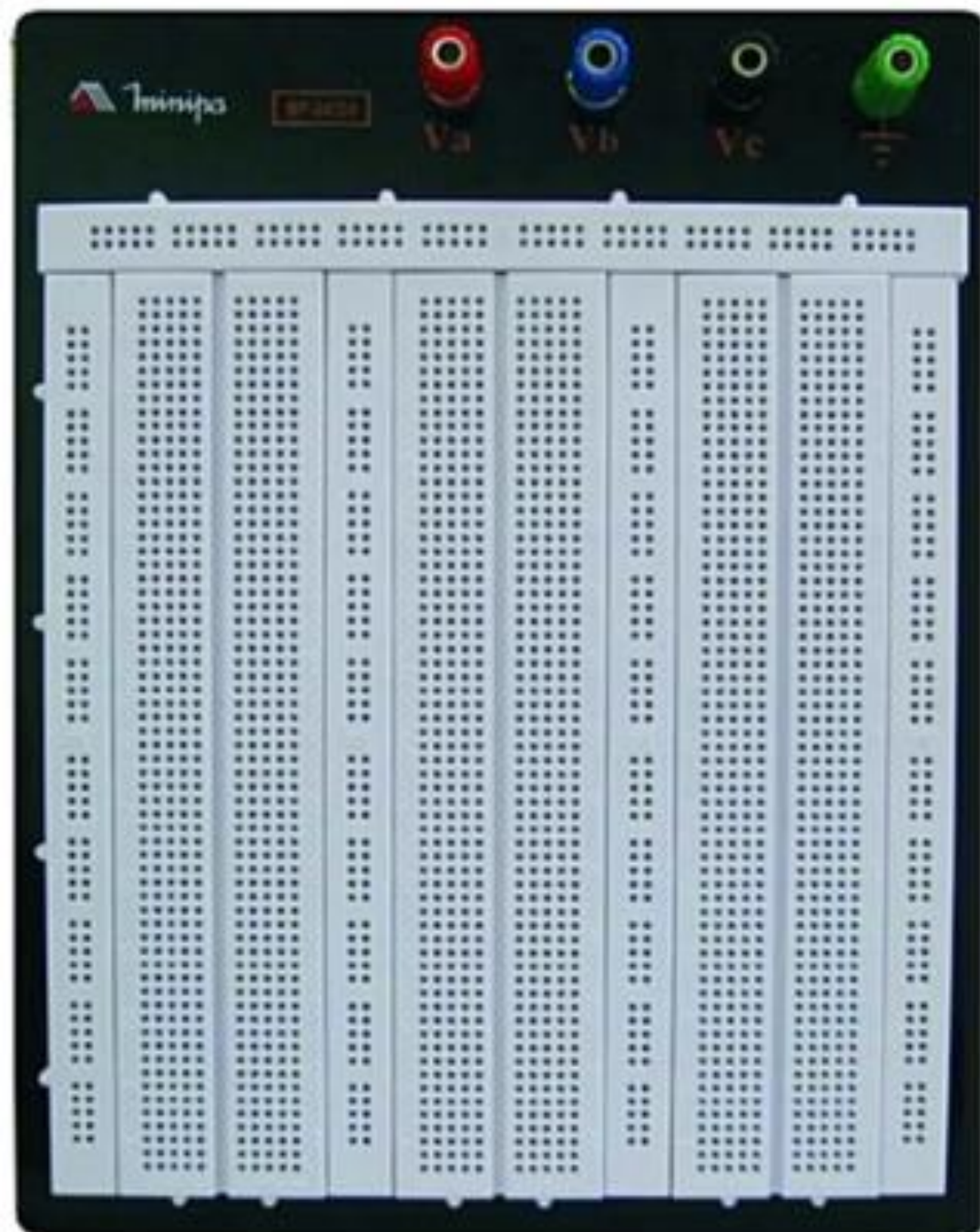
A linha **Preta** é toda interligada e serve para ligar o **Negativo** da fonte de alimentação: GND, VSS, 0V, Terra, -

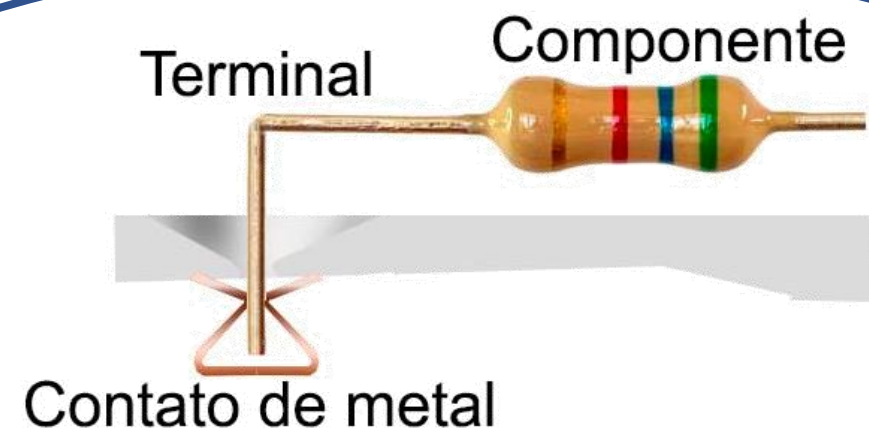
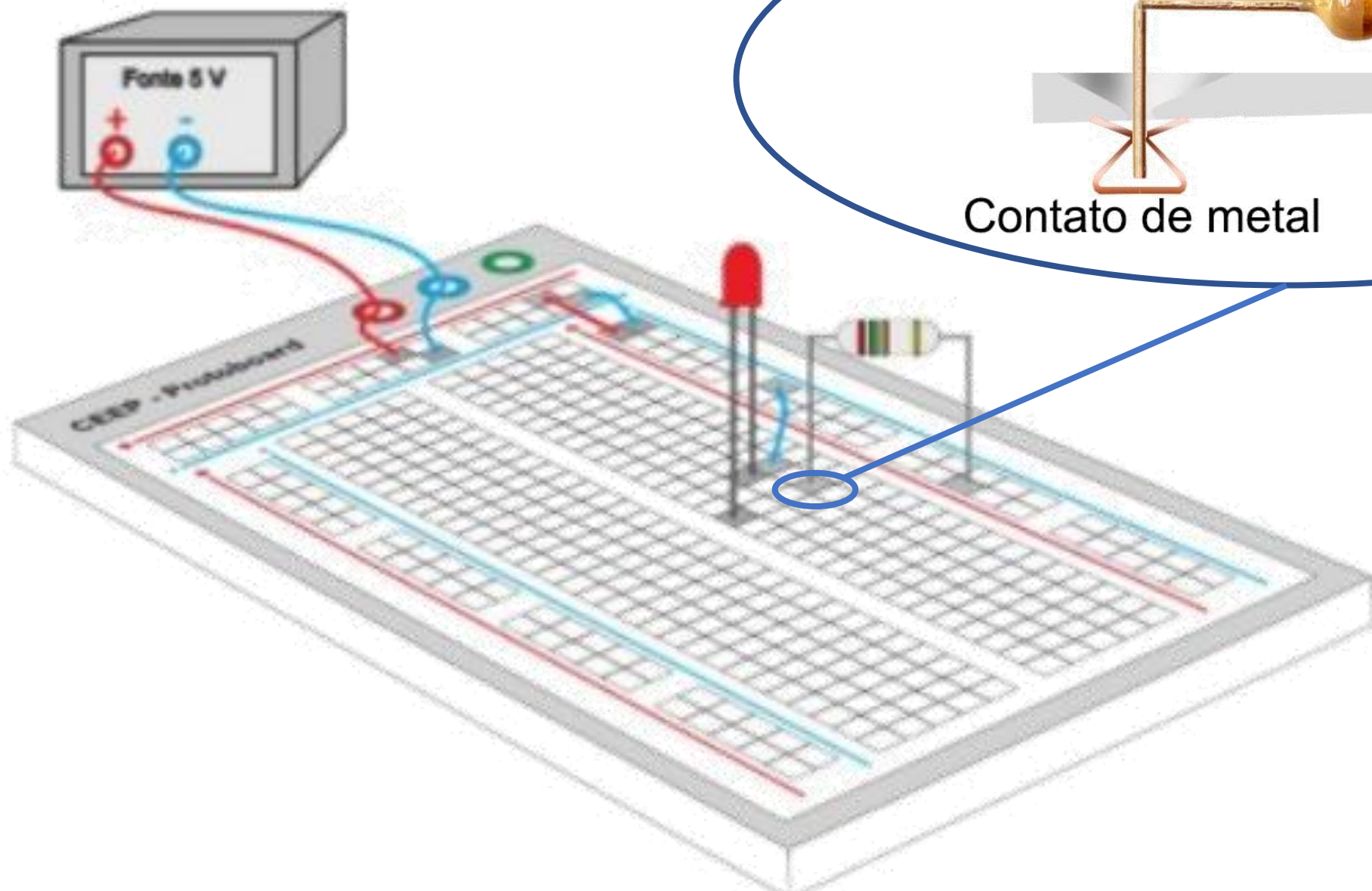
As linhas A, B, C, D e E estão ligadas na **VERTICAL**, em forma de colunas, e uma **coluna não fala com a outra**.

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

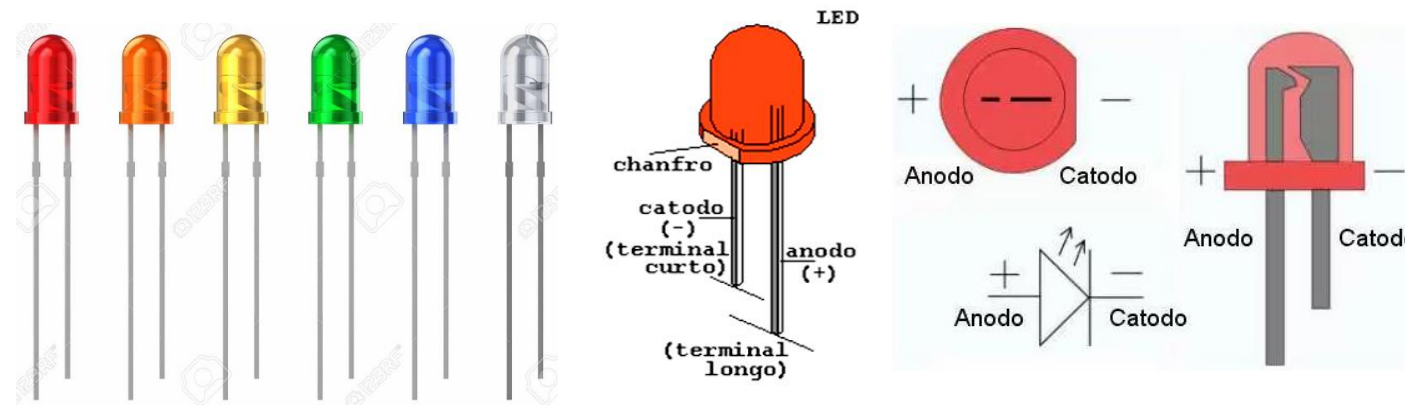






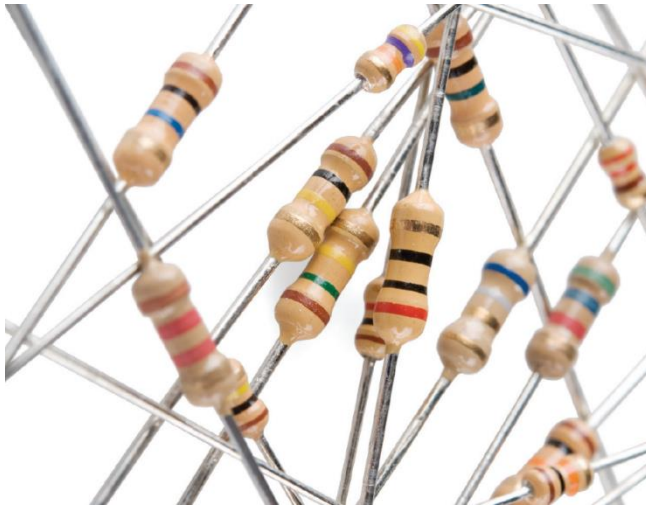
Conhecendo o Hardware – LED

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);

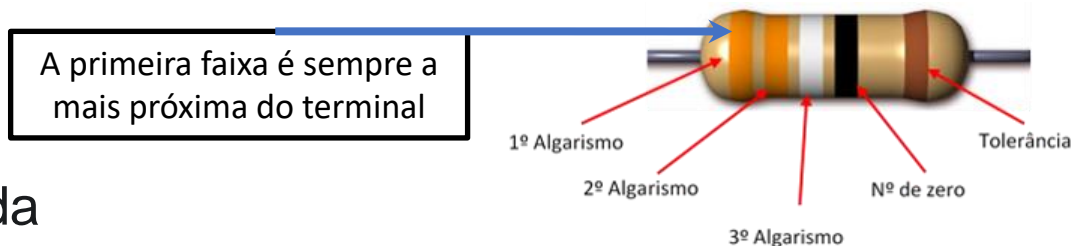
Conhecendo o Hardware – Resistor

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%

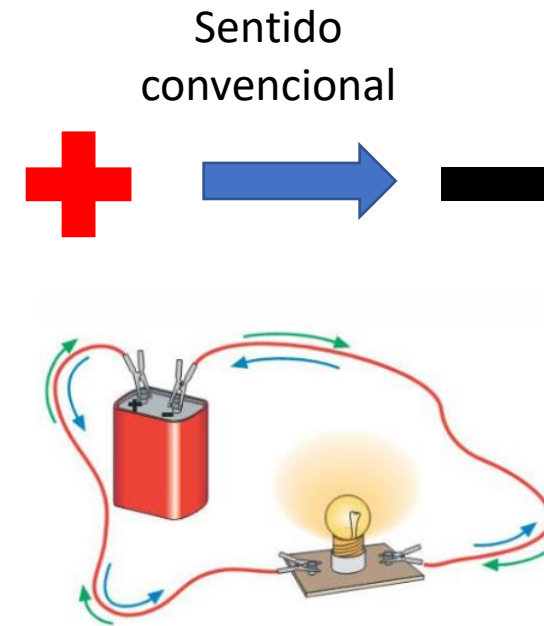
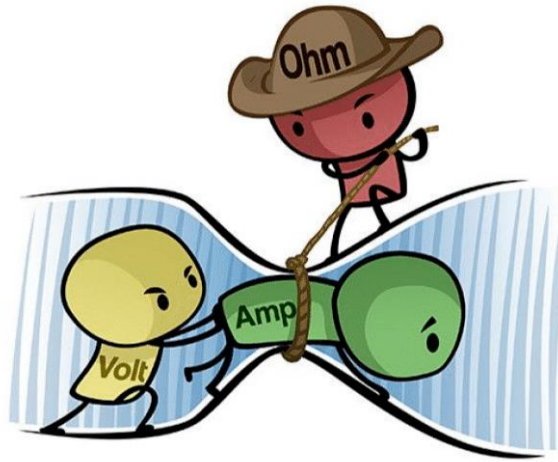
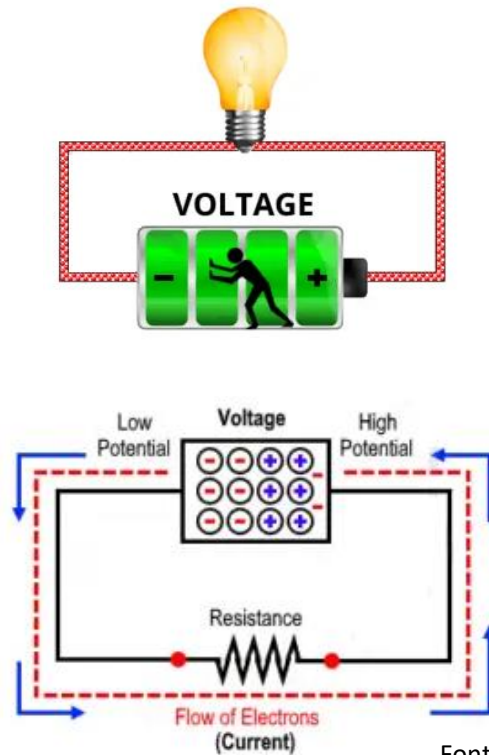
Resistor = 339 x 1, 1%
Resistor = 339 Ohms +/- 1%



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%

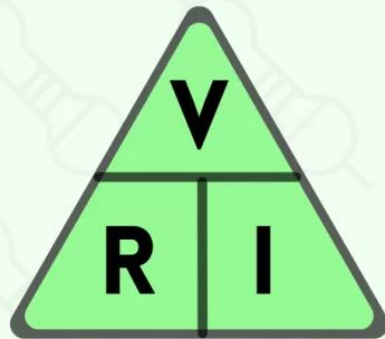
Fonte: <https://aprendendoeletrica.com/codigo-de-cores-para-resistores/>

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

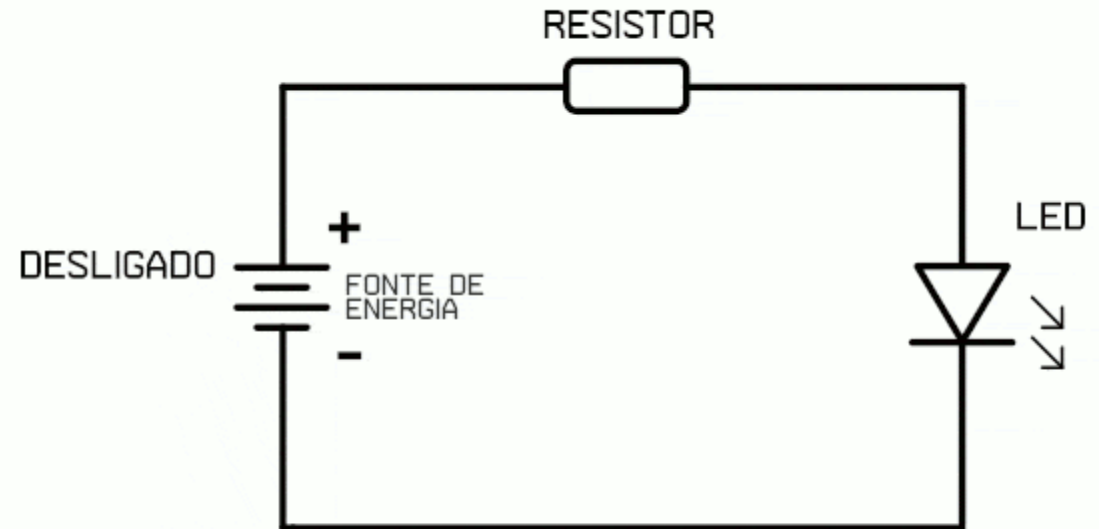


@Trick.Drawing


V = Tensão elétrica [V];


R = Resistência elétrica [Ω];


I = Corrente elétrica [A].



Conhecendo o Software

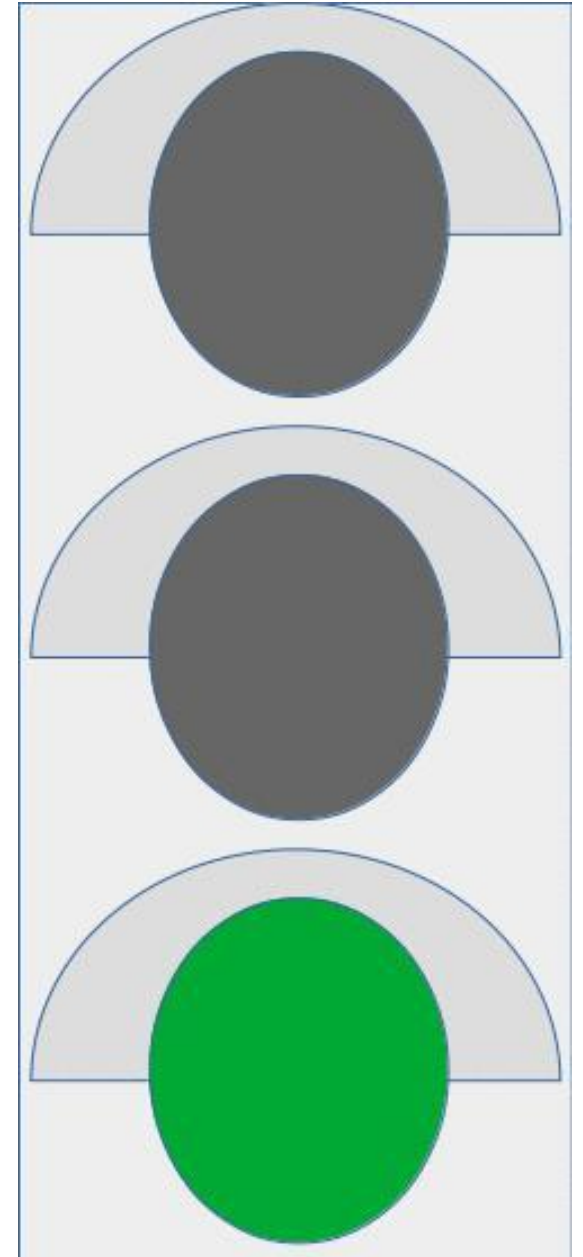
 Primeiro declaramos em qual pino vamos ligar o Led, nesse caso, no pino 10;

 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





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).