

Treinamento de Microcontroladores

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?

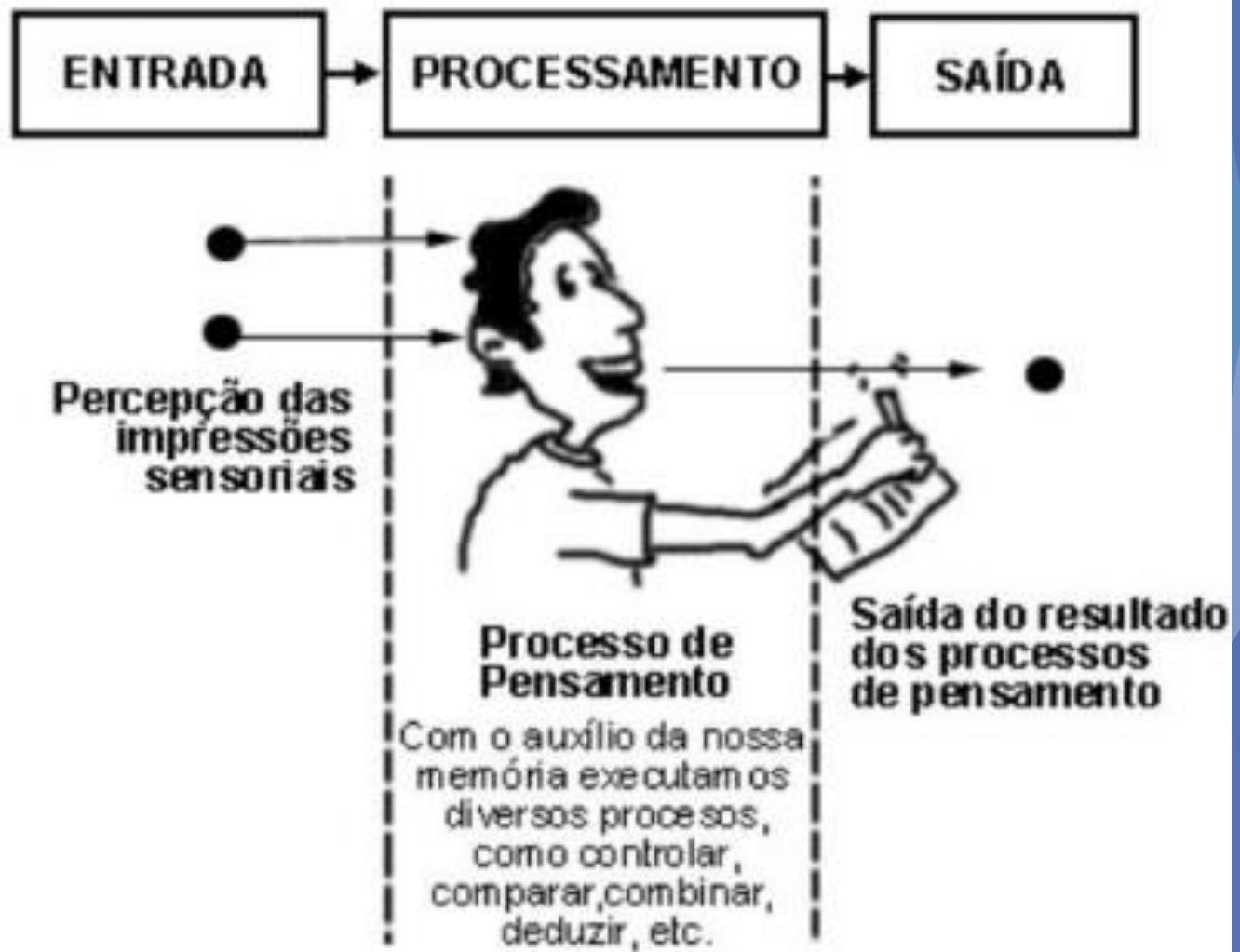
- 1 – Comprar uma lâmpada nova
- 2 – Pegar uma escada
- 3 – Desligar o interruptor
- 4 – Pegar a lâmpada nova
- 5 – Subir na escada
- 6 – Remover a lâmpada queimada
- 7 – Instalar a lâmpada nova
- 8 – Descer da escada
- 9 – Descartar a lâmpada queimada
- 10 – Acionar o interruptor



Lógica de programação

Processo básico de um algoritmo

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



Lógica de programação: fluxogramas

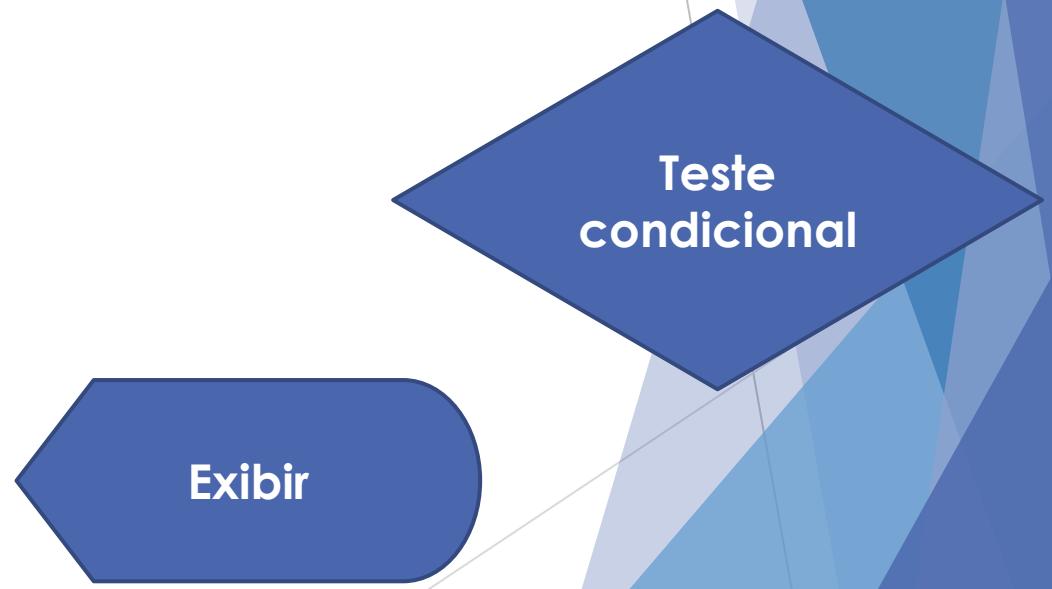
Diagrama de blocos

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

Facilita a visualização dos passos de um processamento

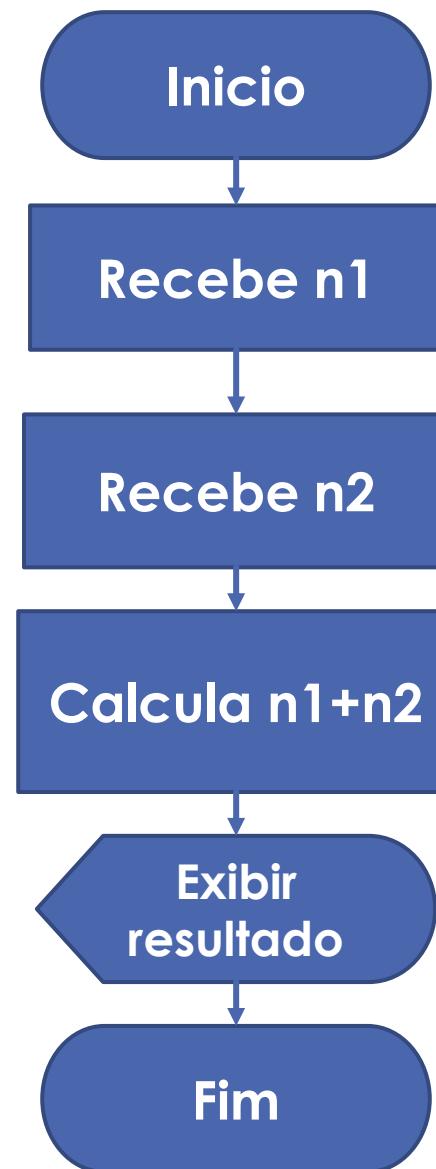


Processamento geral.
Neste curso, será usado
também para entrada
ou saída de dados



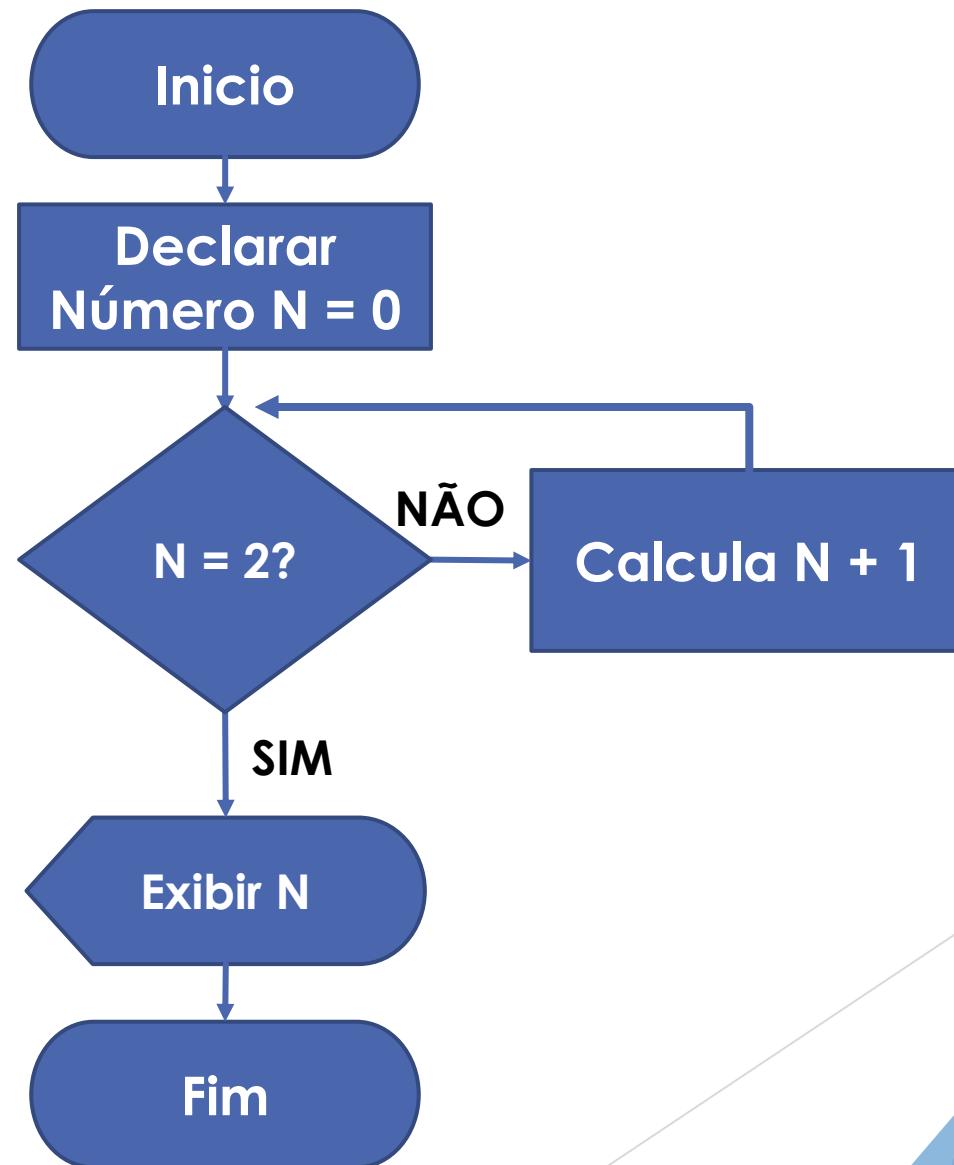
Lógica de programação: fluxogramas

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 contém 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	1	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 Parenteses ()
- 2º Multiplicação ou divisão
- 3º Soma ou subtração

Lógica de programação: operadores relacionais

Usados para comparações (decisões), retornando valores lógicos

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

Lógica de programação: operadores relacionais

Usados para comparações (decisões), retornando valores lógicos

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

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

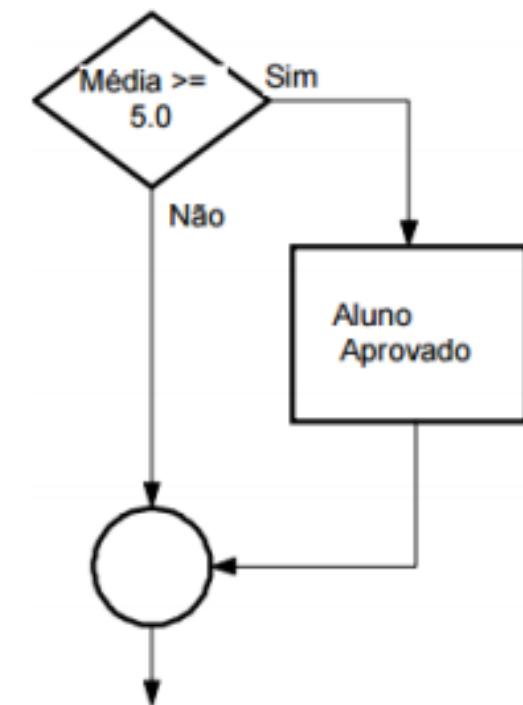
Lógica de programação: estruturas de decisão

“**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

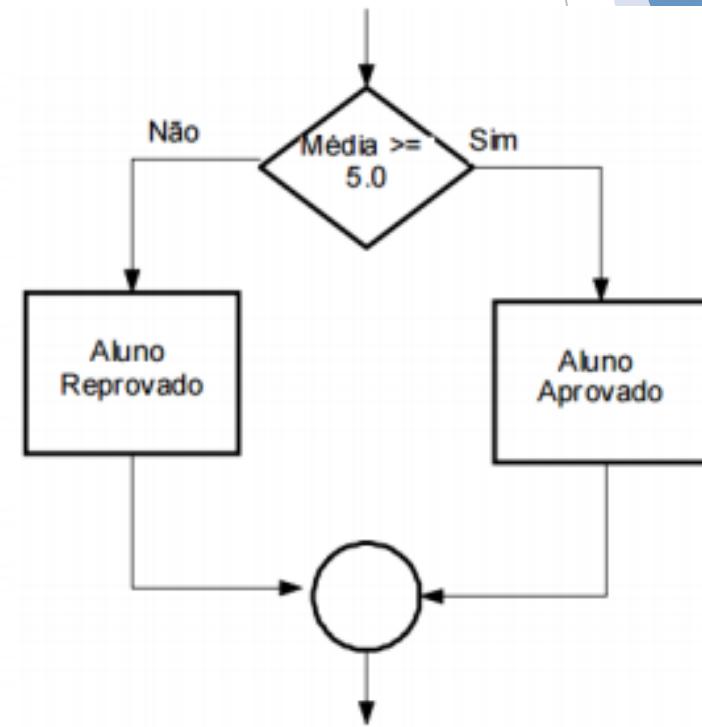


Lógica de programação: estruturas de decisão

"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

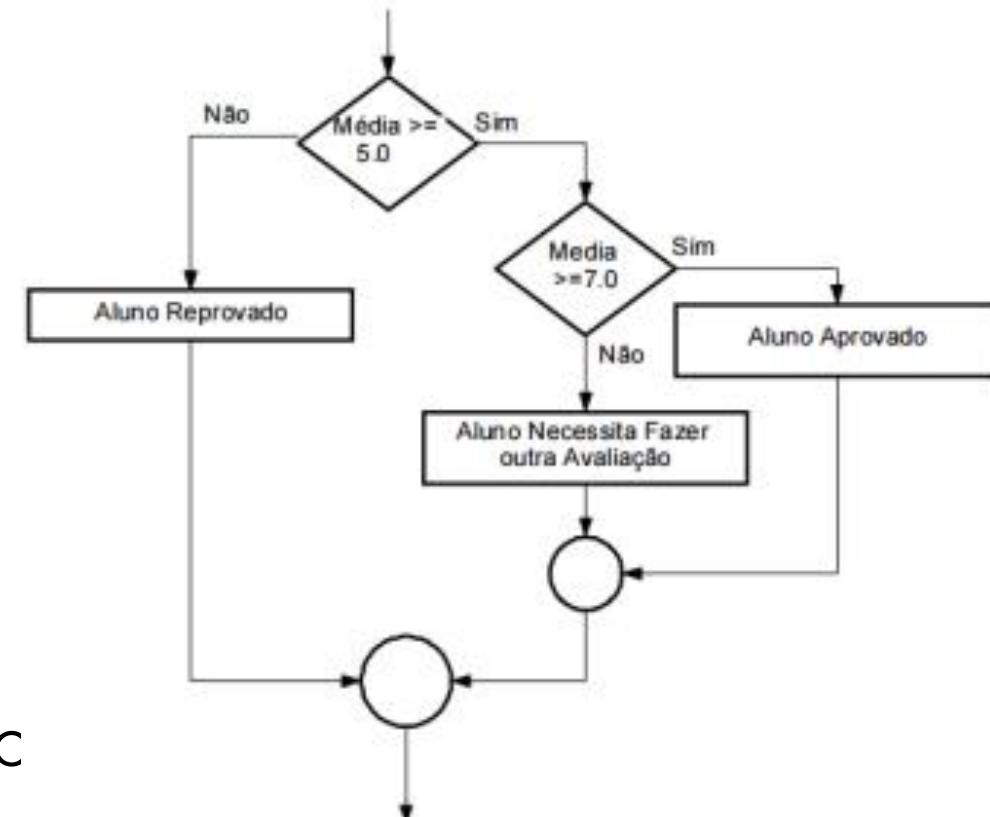
2 – **ENTÃO** faça:

3 – SE sua nota for maior ou igual a 7

4 – ENTÃO o aluno está aprovado

5 – SENÃO o aluno deve fazer recuperação

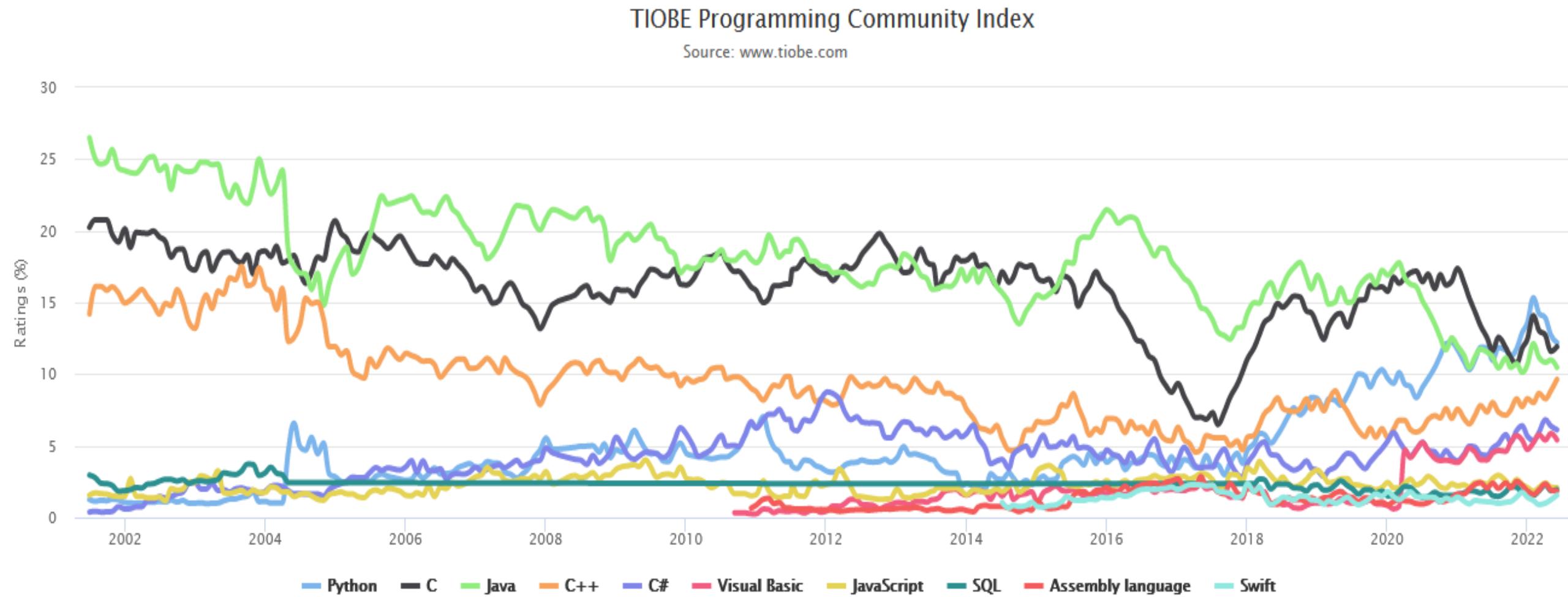
6 – **SENÃO** o aluno está reprovado



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



Linguagem C

"C is quirky, flawed, and an enormous success."

"C é peculiar, falho e um enorme sucesso."

Dennis Ritchie

Criador do Unix e da Linguagem C



Revisão de Linguagem C

- ▶ Porque C?
- ▶ Funcionalidade de projeto;
- ▶ Eficiência;
- ▶ Portabilidade;
- ▶ Flexibilidade;
- ▶ Orientado ao Programador (Fácil de converter em programas os algoritmos)

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

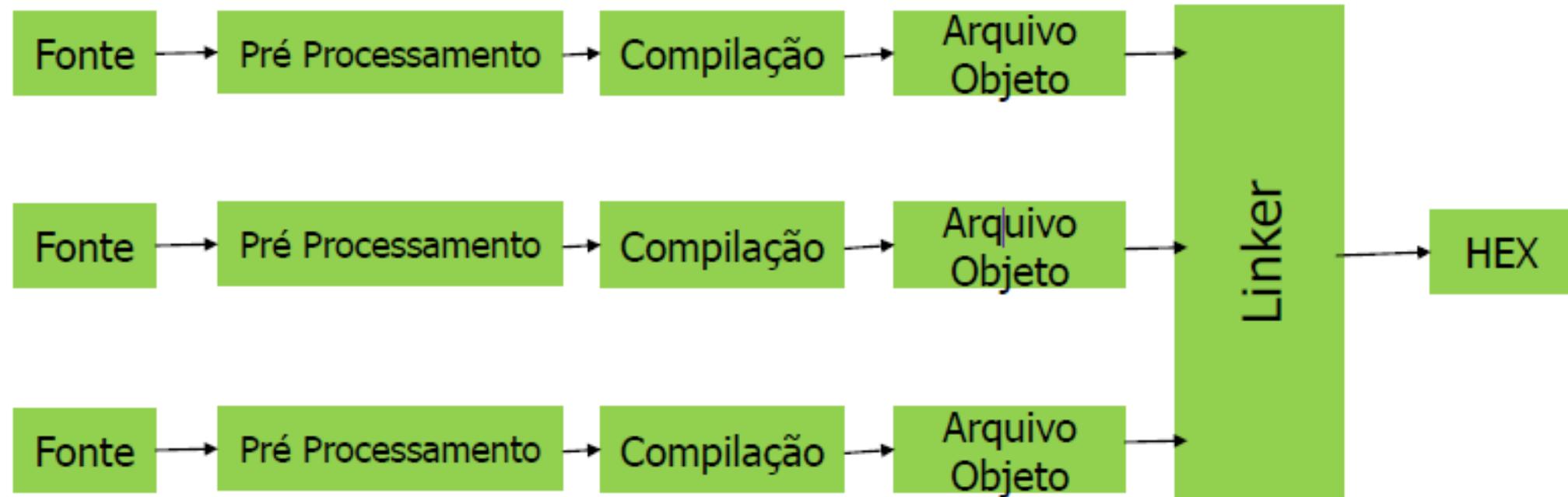
- ▶ C é orientado às necessidades do programador;
 - Permite o acesso ao HW, e permite a manipulação de bits individuais na memória;
 - Possui um grande número de operadores.

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.

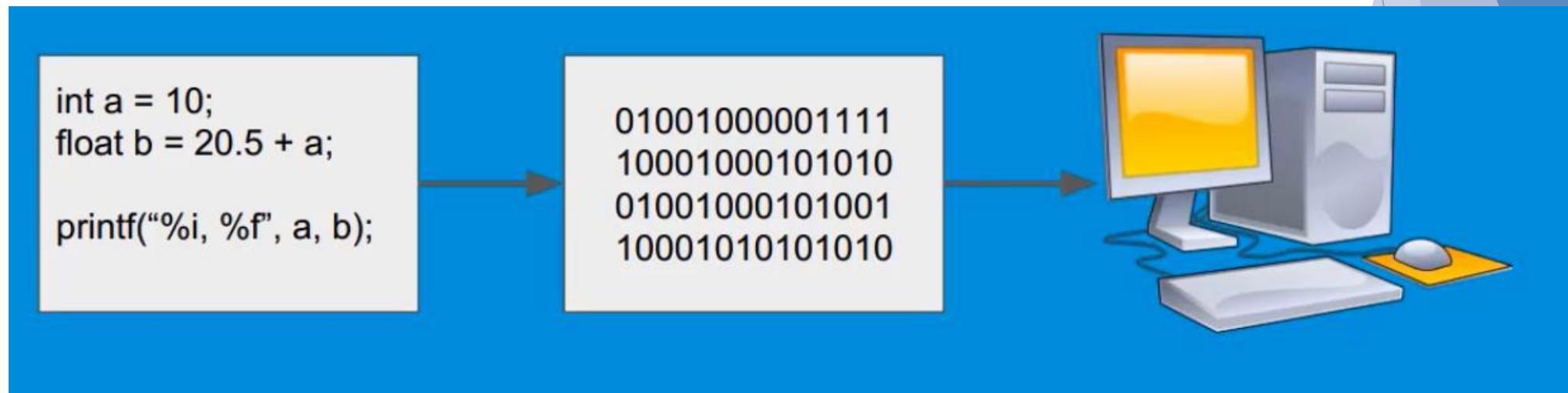
Arquivos Objeto, Executáveis e Bibliotecas

- ▶ 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



Variáveis Global e Local

► **Variáveis Globais:**

São aquelas declaradas no início de um algoritmo. São visíveis, ou seja, podem ser utilizadas no algoritmo principal e por todos os outros sub algoritmos.

► **Variáveis Locais:**

São aquelas declaradas no início de um sub algoritmo. São invisíveis, ou seja, podem ser utilizadas somente pelo sub algoritmo onde foram declaradas.

Outros sub algoritmos ou mesmo o algoritmo principal não podem utiliza-las.

Variável Global

- A variável valor pode ser alterado por diferentes funções.

```
#include <stdio.h>

int valor = 0;

void incrementa_valor(void)
{
    valor = valor + 1;
}

void decrementa_valor(void)
{
    valor = valor - 1;
}

int main (void)
{
    incrementa_valor();
    incrementa_valor();
    incrementa_valor();
    decrementa_valor();
    printf("Valor final de %i",valor);
    return 0;
}
```

Variável Local

- A variável só existe dentro do escopo da função.

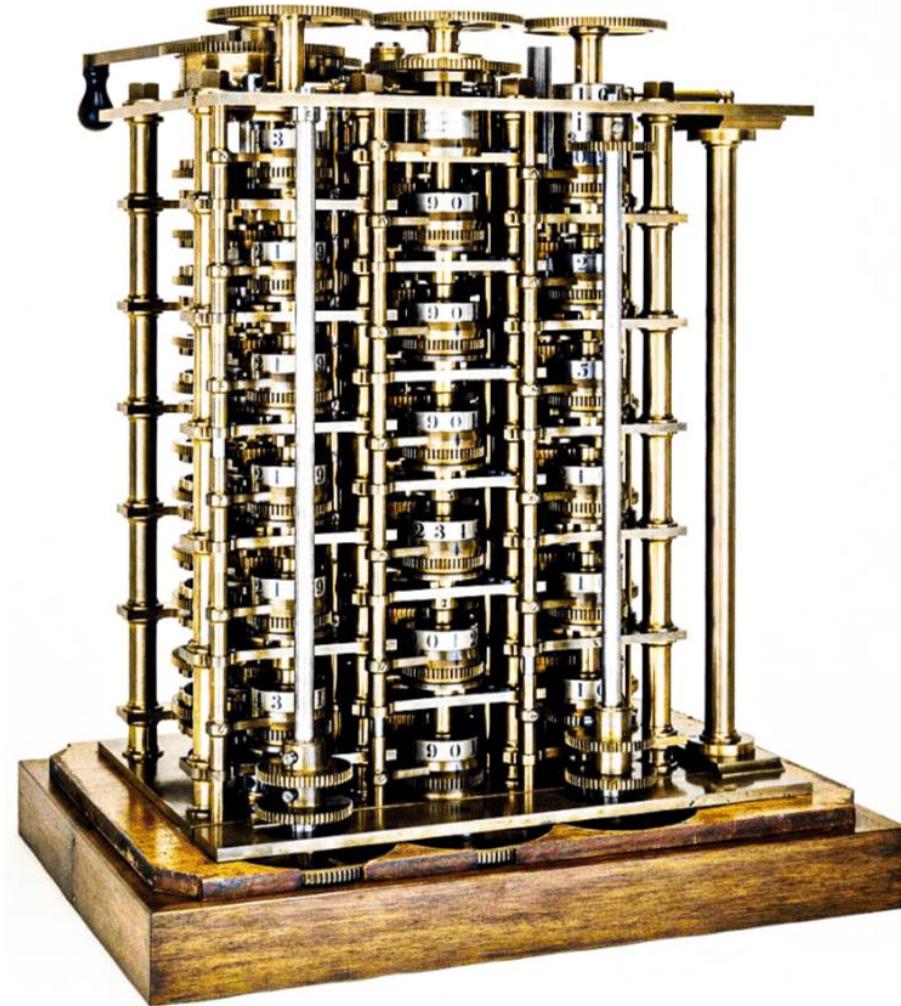
```
{  
    // Alocação de memória  
  
    // Desalocação de memória  
}
```

```
#include <stdio.h>  
  
int main (void)  
{  
    int valor = 0;  
    valor = valor + 1;  
    valor = valor + 1;  
    valor = valor + 1;  
    valor = valor - 1;  
    printf("Valor final de %i",valor);  
    return 0;  
}
```

Introdução a microcontroladores

Século XIX – Charles Babbage

- ▶ Conceito do primeiro computador é criado com a **Máquina Diferencial** e a **Máquina Analítica** criada por Charles Babbage, porém não foi implementada devido a limitações tecnológicas



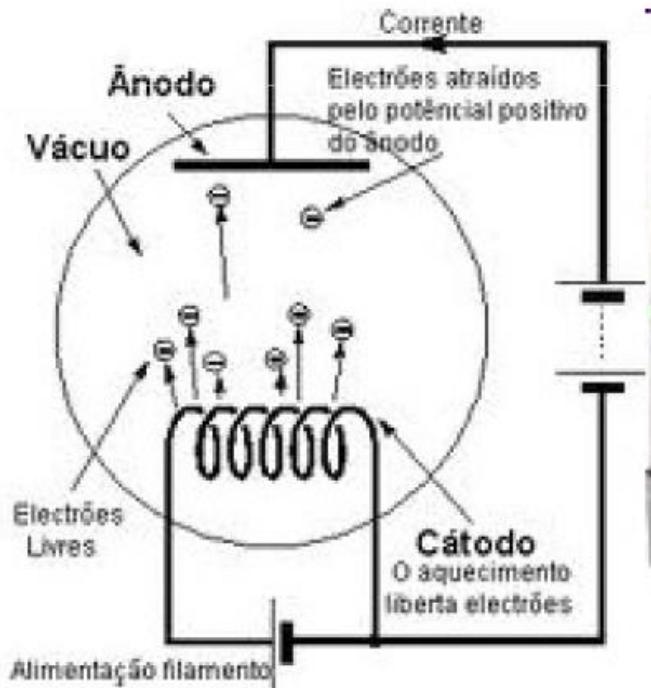
Ada Lovelace

Antes mesmo do primeiro computador, surge a primeira programadora: **Ada Lovelace**, que viu nos computadores a capacidade de ir além dos cálculos.



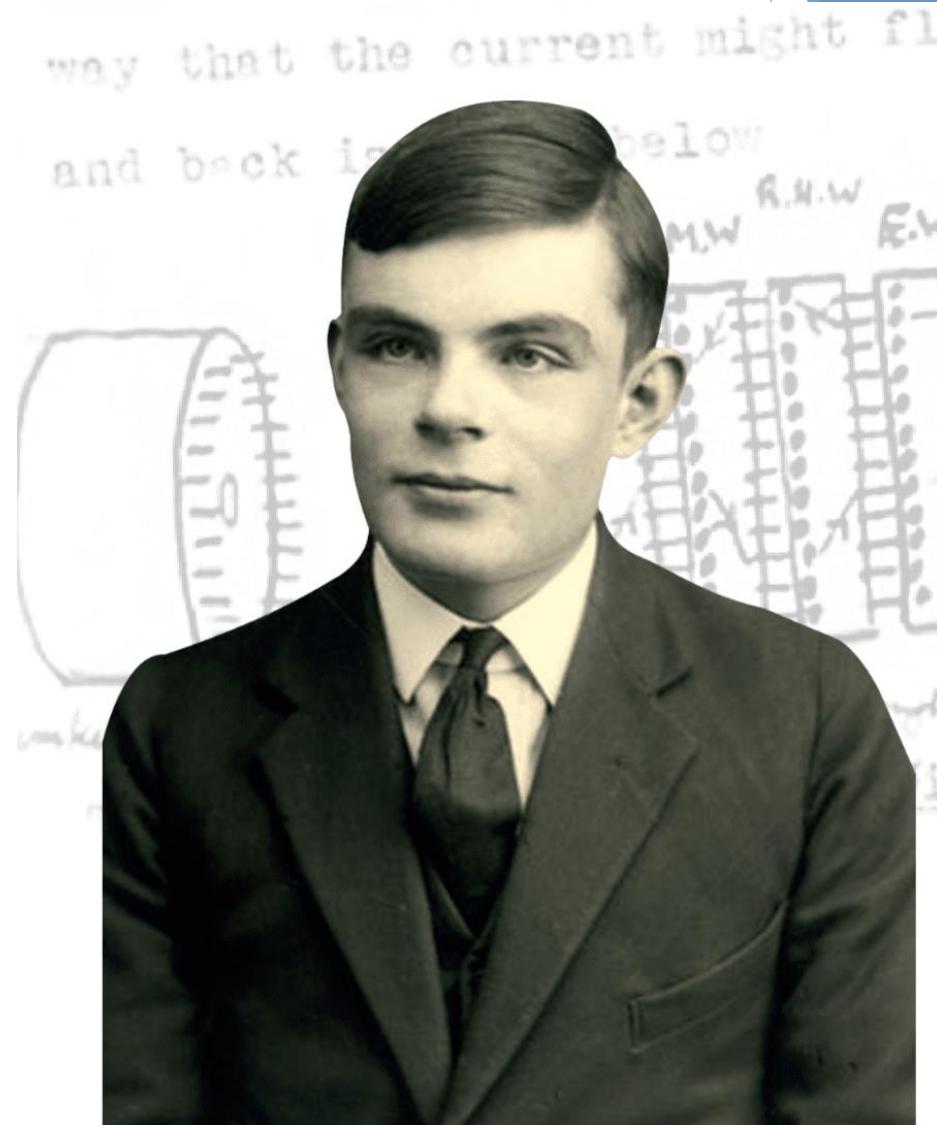
Válvulas

- ▶ Por volta de 1900 com o advento da eletricidade, começou o desenvolvimento das válvulas diodo e tríodo.



Alan Turing

- ▶ Na Inglaterra, Alan Turing cria em 1936 uma tese revelando o primeiro conceito de máquina com inteligência artificial



Bombe - code-breaking machine

Durante a segunda Guerra (1938 – 1945), Turing e sua equipe usa essa base para criar uma máquina eletromecânica para decifrar mensagens criptografadas de comunicação militar da Alemanha, o que foi decisivo para a vitória dos Aliados (Inglaterra/Estados Unidos/França) contra os países do Eixo (Alemanha/ Itália /Japão)

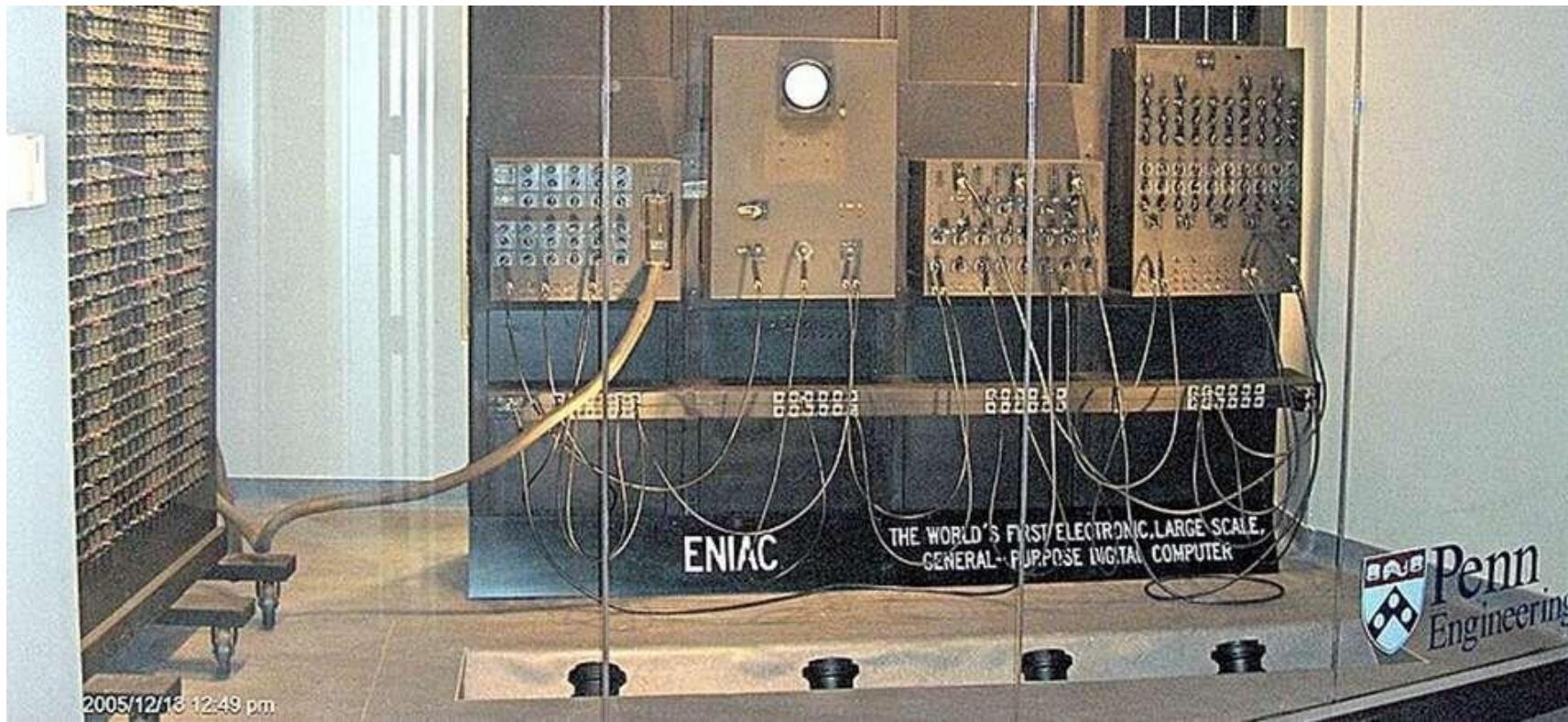


Histórico

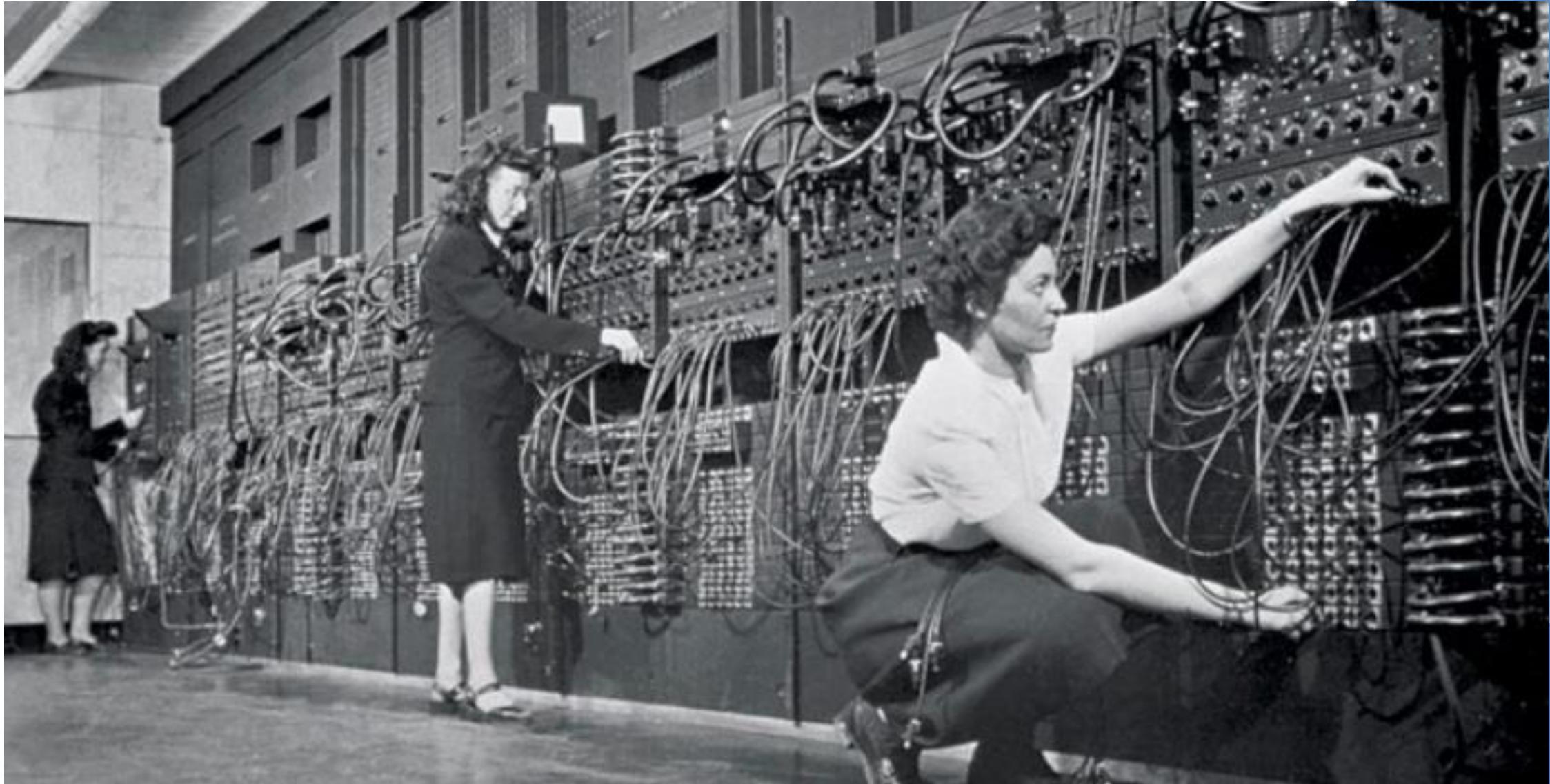
- ▶ A partir dos anos de 1920, começaram os primeiros estudos com materiais **semicondutores**.
- ▶ O objetivo era criar componentes que substituíssem as válvulas e os relés no processamento de informações, com inúmeras vantagens.

ENIAC

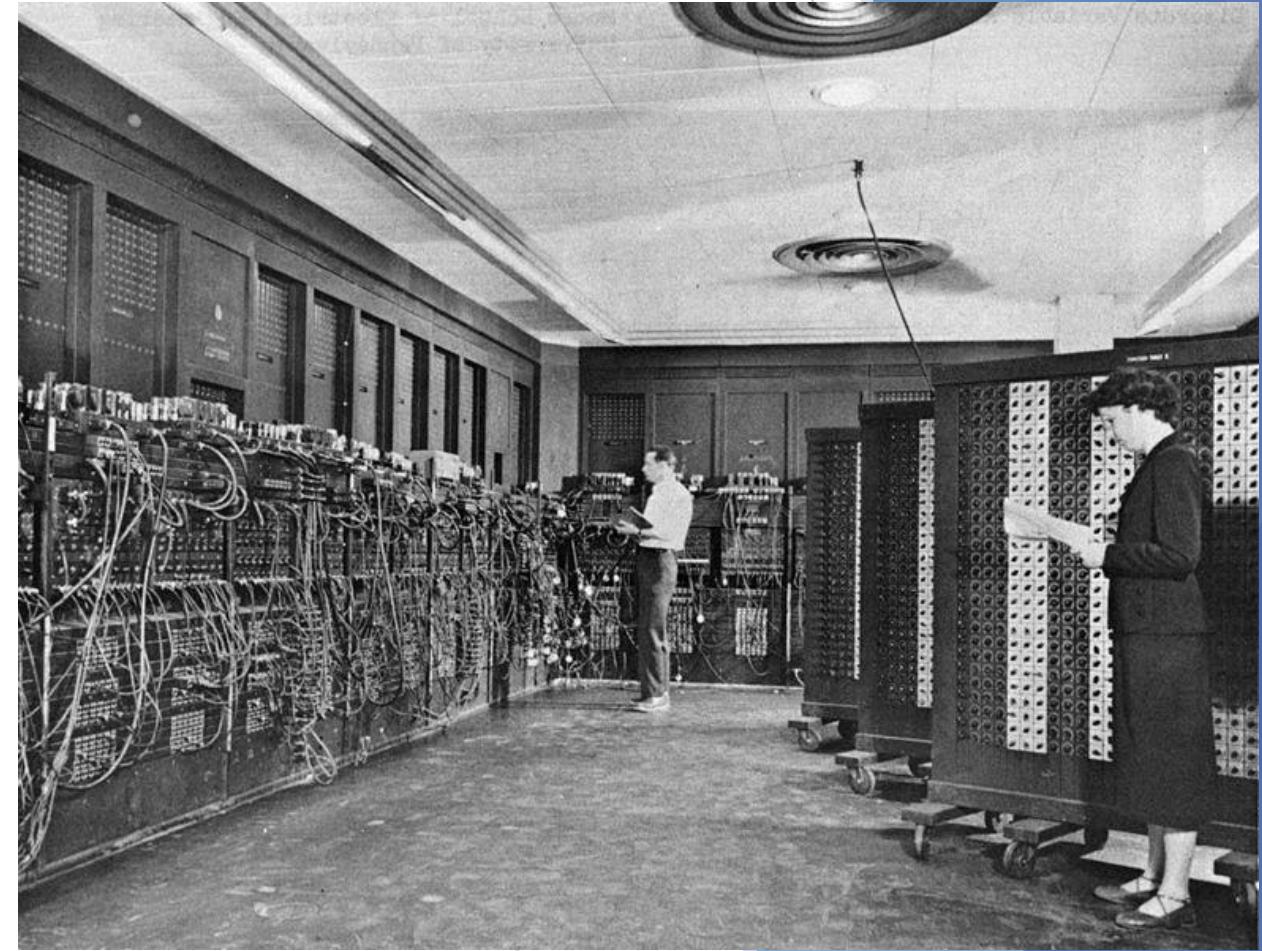
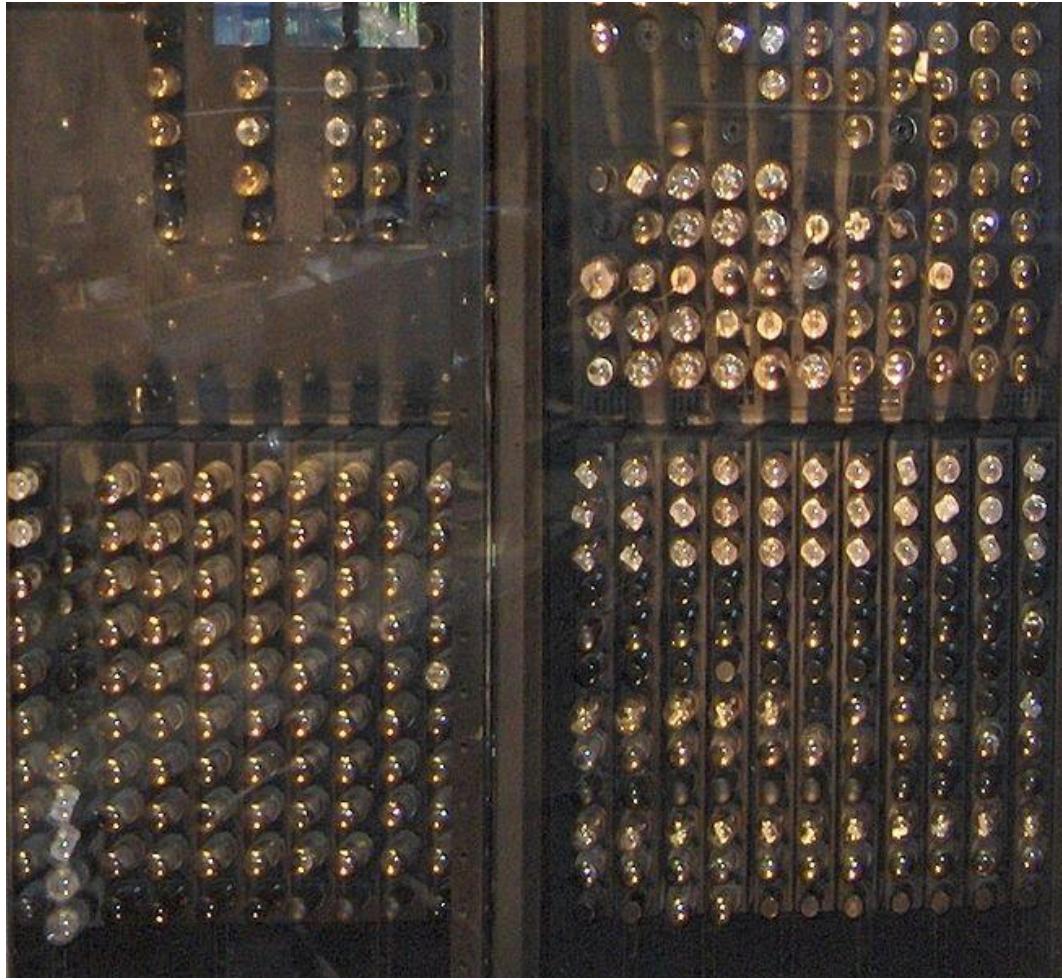
- Enquanto as pesquisas corriam, o primeiro grande computador da história era inaugurado: o ENIAC.



ENIAC



ENIAC



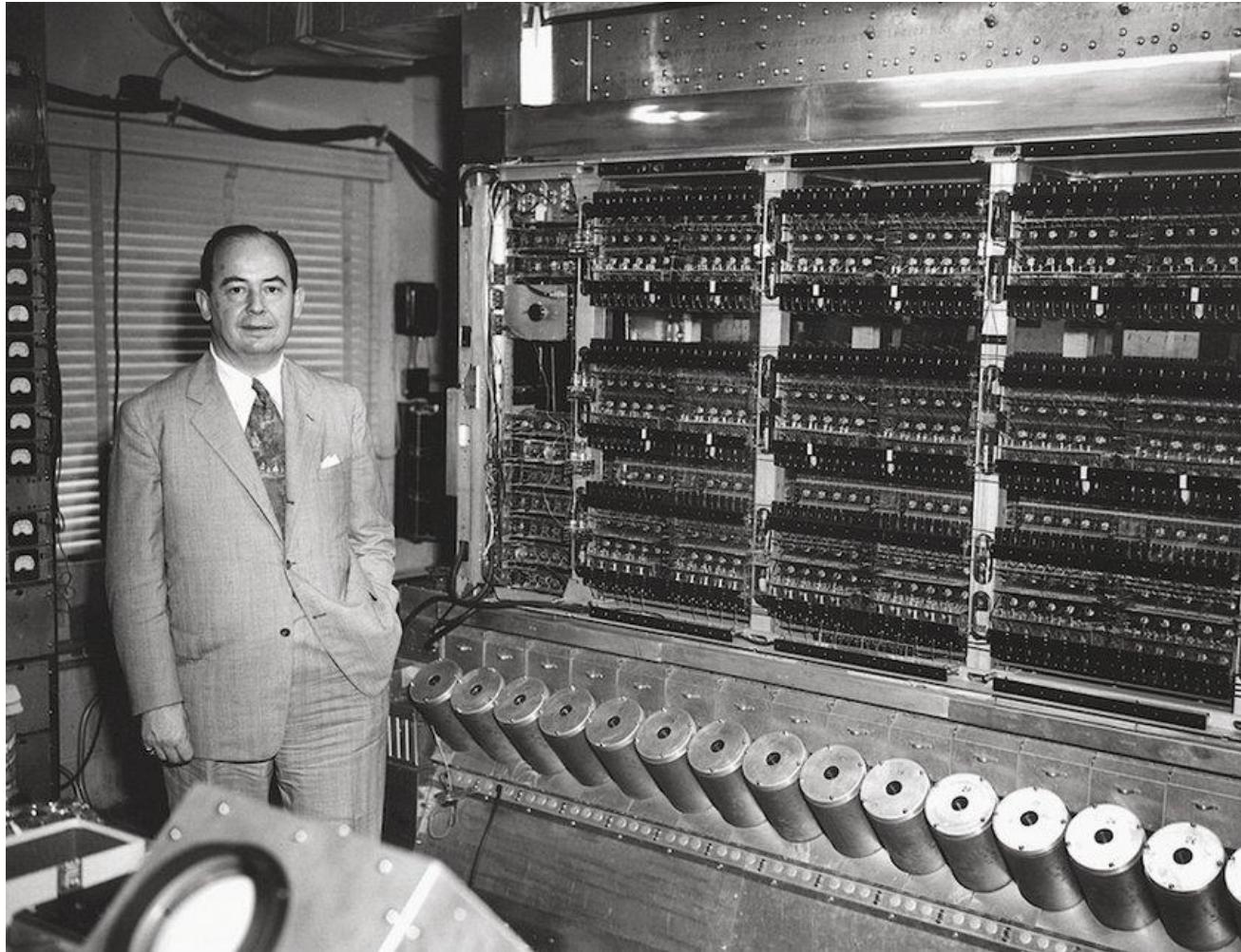
ENIAC

- ▶ Média 5,50m de altura e 25m de comprimento, totalizando uma área de 180m².
- ▶ Era composto por 70 mil resistores, 18 mil válvulas, 10 mil capacitores, 1500 relés e 6 mil chaves interruptoras.
- ▶ Consumia 200 000 W de potência.
- ▶ Quando era ligado/reiniciado, as luzes do Estado da Filadélfia piscavam.
- ▶ Foi criado com fins militares, para fazer cálculos de trajetórias de mísseis balísticos e outros cálculos de guerra.
- ▶ A sua capacidade de processamento era a de realizar 5 mil cálculos por segundo.

ENIAC

- ▶ Funcionava utilizando lógica digital (0 e 1);
- ▶ Pesava 30 toneladas (30000kg);
- ▶ Internamente, a temperatura chegava a 50°C;
- ▶ A cada 10min em média, uma válvula queimava;
- ▶ Inicialmente tinha uma equipe de 80 programadores;
- ▶ Todos os programadores eram mulheres;

EDVAC – (Electronic Discrete Variable Automatic Computer)



EDVAC – (Electronic Discrete Variable Automatic Computer)

- ▶ Diferentemente de seu predecessor ENIAC, este utilizava o **sistema binário** e possuía arquitetura de **von Neumann**.
- ▶ Esses conceitos foram fundamentais para o desenvolvimento dos computadores como conhecemos atualmente.

O Transistor

- ▶ Enquanto o ENIAC estava em pleno vapor, os cientistas John Barden, Walter Bratain e William Shockley inventam o Transistor nos laboratórios da BELL.

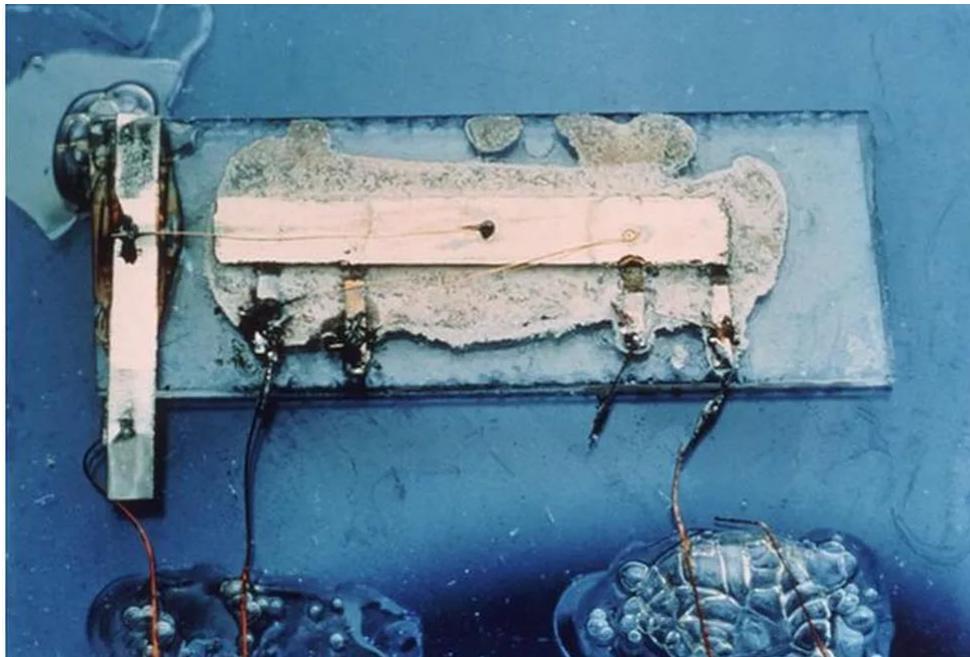


O Transistor

- ▶ Esse foi um grande marco na história. Muitos consideram o transistor como a mais importante invenção da história da humanidade.
- ▶ Isso porque o transistor substituiu as válvulas com inúmeras vantagens, tais como tamanho, consumo de energia, temperatura, velocidade de comutação, custo de produção, etc.

O primeiro CI

- ▶ Graças ao transistor, em 1959 a empresa TEXAS INSTRUMENTS cria o primeiro Circuito Integrado (CI), onde em uma mesma pastilha eram integrados vários transistores.

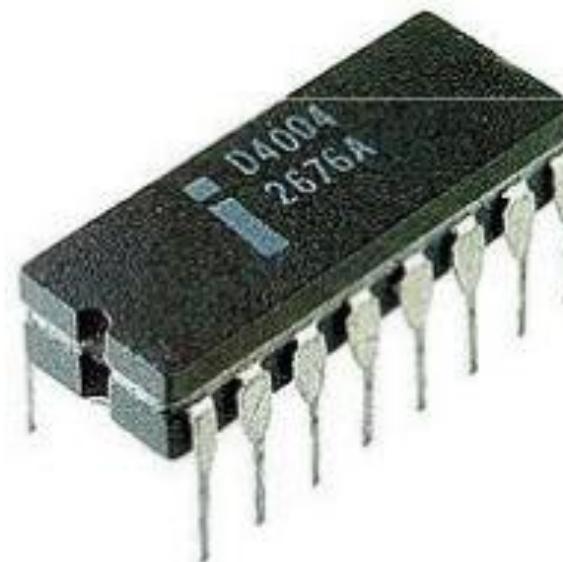
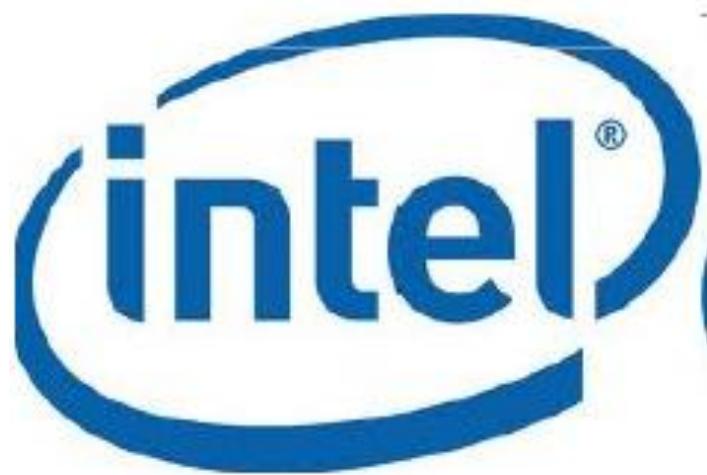


Microprocessador

- ▶ Utilizando chips (CIs), em 1964 a empresa DIGITAL começa a vender o PDP-8, o primeiro computador com preço acessível aos laboratórios
- ▶ 4 anos mais tarde, em 1968, surge a INTEL.
- ▶ Em 1971 a Intel revoluciona o mercado ao lançar o primeiro microprocessador da história, o Intel 4004.
- ▶ O 4004 foi o primeiro CI onde todo o circuito de controle e programação estava integrado em um único chip.

CI 4004

- ▶ Tinha capacidade de realizar 6 mil cálculos de soma por segundo.



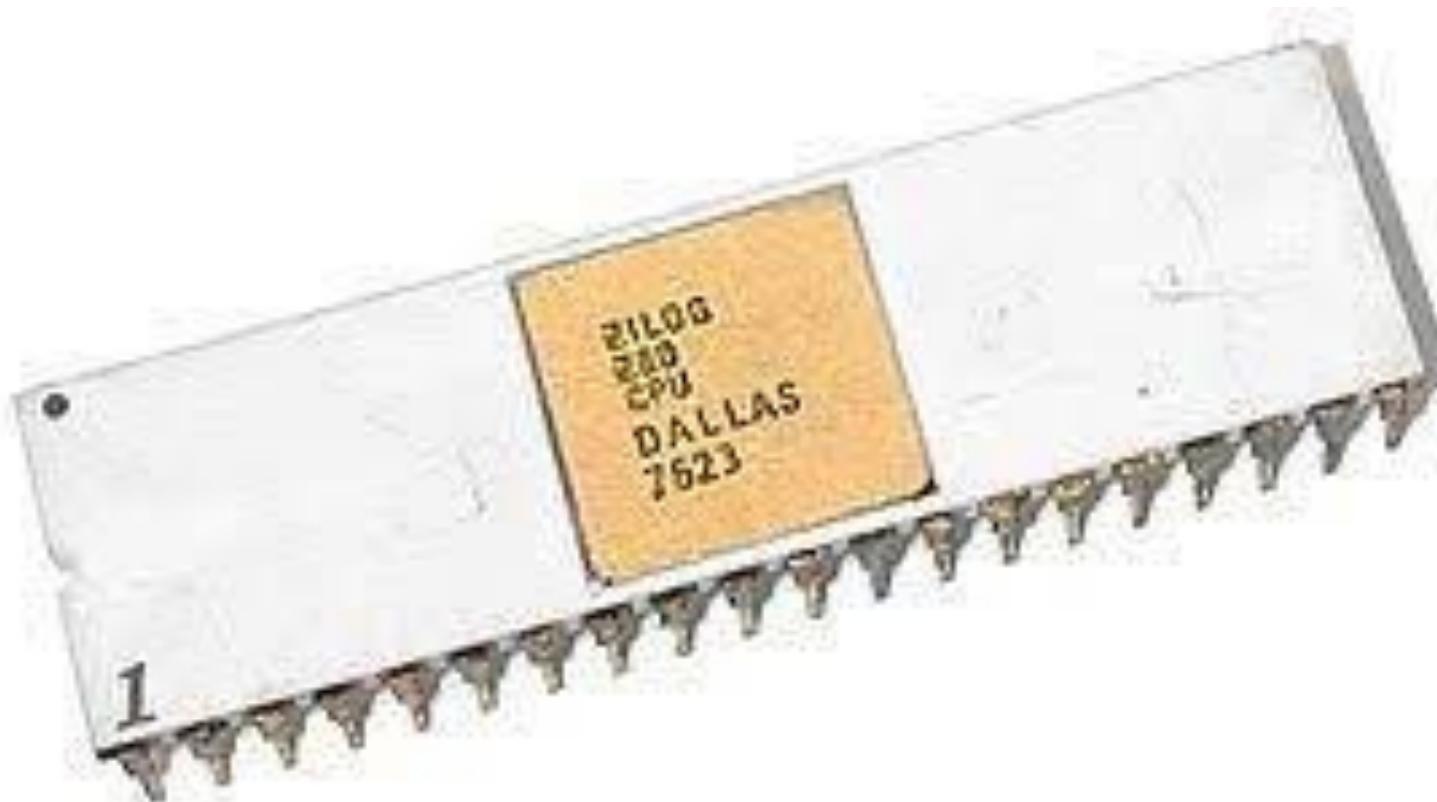
Enquanto isso no Brasil...

- ▶ O primeiro computador brasileiro é construído na Escola Politécnica da USP em 1972 e se chamava **Patinho Feio**.



Z80

- Em 1975 a empresa ZILOG fabrica o famoso microprocessador Z80.



6502 – MOS Technology

- A empresa MOS Technology lançou em 1976 o CI 6502 é um microprocessador de 8 bits projetado por **Chuck Peddle** (Ex-Intel), custando cerca de 1/6 (ou menos) do preço de dispositivos similares feitos por grandes empresas concorrentes.



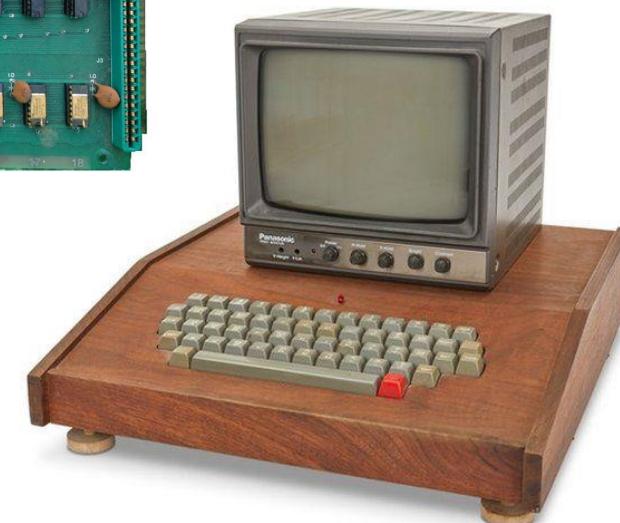
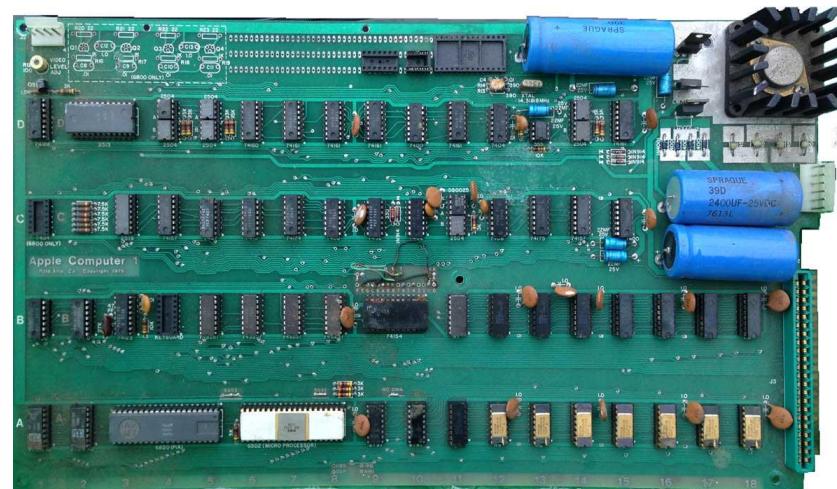
Atari 2600

- ▶ O 6502 foi o microprocessador utilizado nos videogames Atari.



Apple I

- ▶ O 6502 foi o microprocessador utilizado nos primeiros computadores a Apple



O primeiro MCU

- ▶ Em 1976 a Intel lança o primeiro MICROCONTROLADOR, o 8048.
- ▶ E 4 anos mais tarde, lança o 8051, o microcontrolador mais famoso da história.
- ▶ A partir desse momento, o desenvolvimento de **microprocessadores** e **microcontroladores** tomam caminhos diferentes.

Os MCUs e MCPs

- ▶ Todos os avanços tecnológicos existentes hoje se devem aos microprocessadores e aos microcontroladores.
- ▶ Mas o que são eles? O que eles fazem? Qual a diferença entre um e outro?

Motivação

- ▶ Desenvolvimento de Sistemas Eletrônicos (Embedded Systems);
- ▶ Mercado de microcontroladores em expansão (Novos chips e famílias);
- ▶ Estima-se que, em poucos anos, em média uma pessoa **interagirá com 500** dispositivos microcontrolados diariamente;
- ▶ Aplicações em diversas áreas.

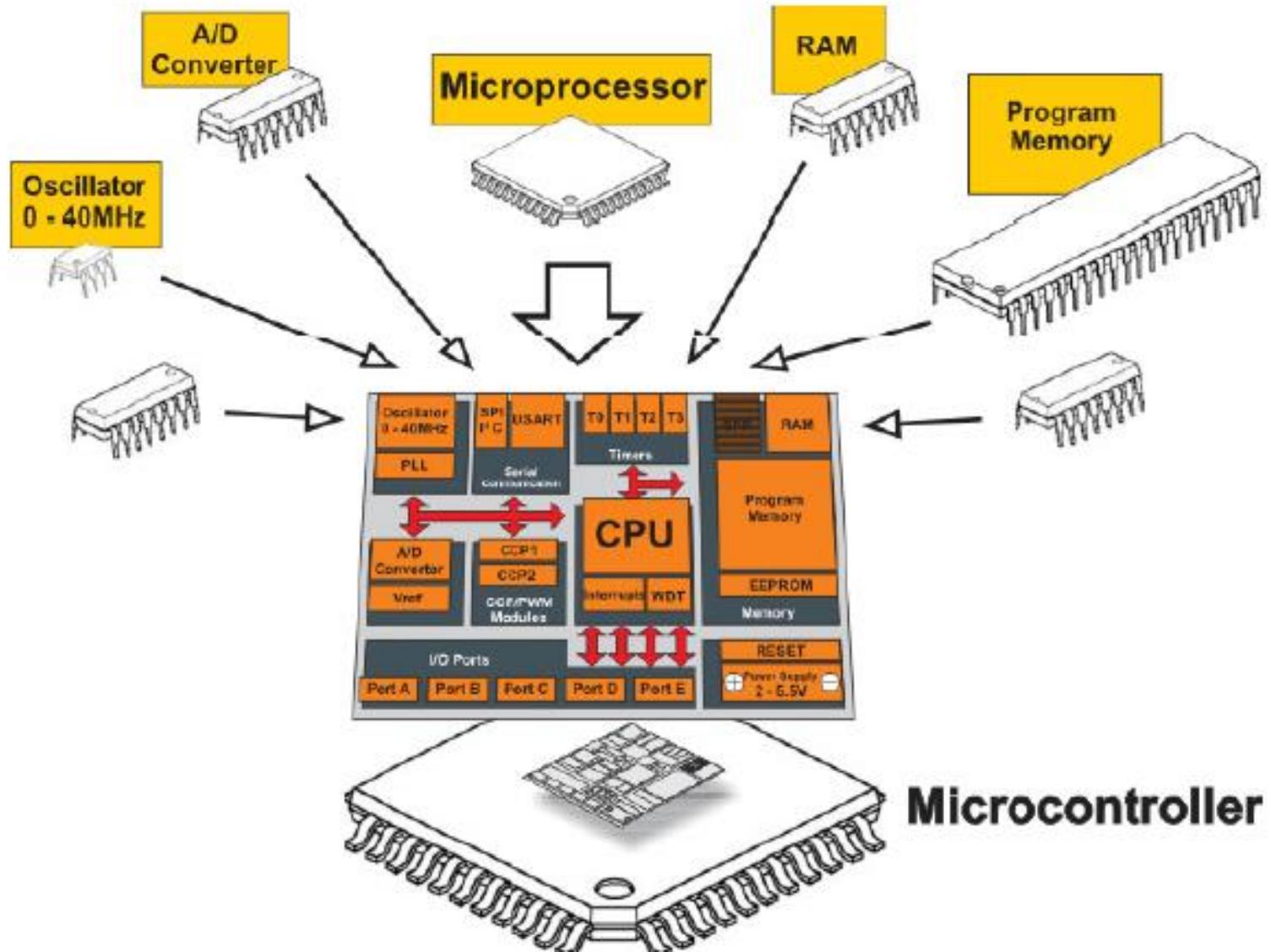
Motivação



Motivação

- ▶ Vantagens do uso de *microcontroladores*:
 - ▶ Circuitos ficam mais compactos (menos componentes) e podem ter mais funções;
 - ▶ Permite armazenamento de sinais (dados) com relativa facilidade;
 - ▶ Facilita correção/modificação das funções do circuito sem alteração de hardware (*programável*);
 - ▶ Facilita integração do circuito com computadores ou outros dispositivos.

Microcontrolador x Microprocessador



Microcontrolador x Microprocessador

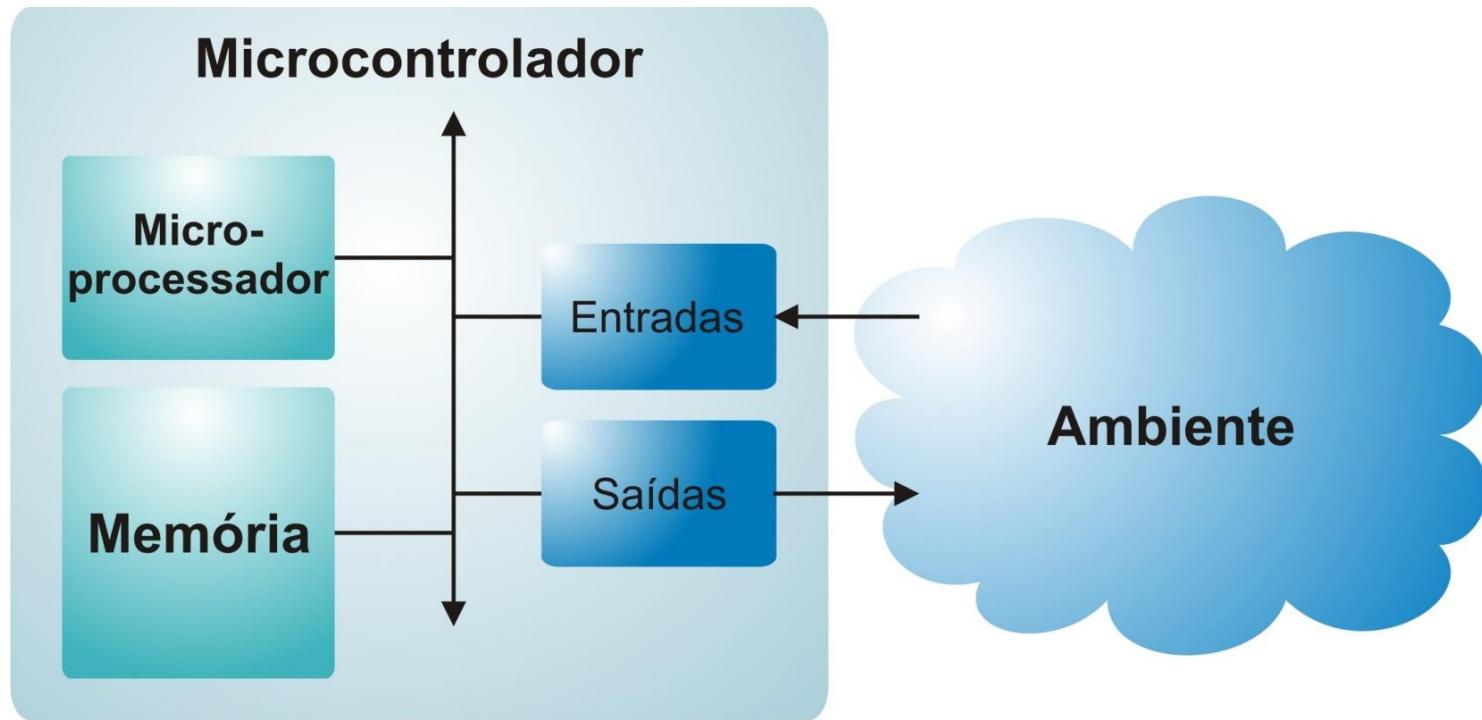
- ▶ O Microcontrolador difere de um microprocessador em vários aspectos:
- ▶ O mais importante deles, é a sua **funcionalidade**.
- ▶ Para que um **microprocessador** possa ser usado, **outros componentes devem ser adicionados**, tais como memória, chipsets e componentes para receber e enviar dados.
- ▶ Por outro lado, o **microcontrolador** foi projetado para ter todas estas funcionalidades em **uma única pastilha**. Comumente, um microcontrolador é chamado de um **computador em um único chip**.

Principais Fabricantes e Modelos

- ▶ Família 8051 (Intel ou Atmel-Microchip/NXP)
- ▶ AVR, ARM PIC16F/18F... (Microchip)
- ▶ ARM, H8 (Renesas)
- ▶ ARM, 8051 (NXP)
- ▶ ARM, STM8 (STMicroelectronics)
- ▶ MSP (Texas Instruments)

Microcontrolador

- Um **microcontrolador** é composto por processador, memória, dispositivos de entrada e saída e outros possíveis elementos, integrados em um mesmo componente (chip).



O que é um Microcontrolador?

- ▶ Um microcontrolador é um sistema computacional completo, no qual estão incluídos:
 - ▶ **Unidade Central de Processamento** (CPU);
 - ▶ **Sistema de Clock** para dar sequência às atividades da CPU;
 - ▶ **Memória** para armazenamento de instruções e manipulação de dados;
 - ▶ **Entradas** para interiorizar na CPU informações do mundo externo;
 - ▶ **Saídas** para exteriorizar as informações processadas pela CPU para o mundo externo;
 - ▶ **Programa (Firmware)** para que o sistema faça alguma coisa útil;
 - ▶ Além de outros possíveis **periféricos**, tais como:
 - ▶ Módulos de temporização, comunicação serial, conversores A/D entre outros;

Arquitetura Básica

- ▶ **Tipos de Memória**
- ▶ RAM (Random Access Memory)
 - ▶ Armazena dados dos programas;
 - ▶ Volátil;
- ▶ ROM (Read Only Memory)
 - ▶ Programa e dados fixos;
 - ▶ Geralmente programadas na fábrica e seus dados não podem ser modificados

Arquitetura Básica

- ▶ **Tipos de Memória**
- ▶ **EPROM (Erasable Programmable Read Only Memory):**
 - ▶ Similar à ROM, mas pode ser programada;
 - ▶ Possuem uma janela de vidro sobre o chip onde os dados podem ser apagados através de luz UV;
- ▶ **EEPROM (Electrically Erasable Programmable Read Only Memory)**
 - ▶ Não-volátil;
 - ▶ Podem ser apagadas ou gravadas sob comando de programa;

Arquitetura Básica

► ***Tipos de Memória***

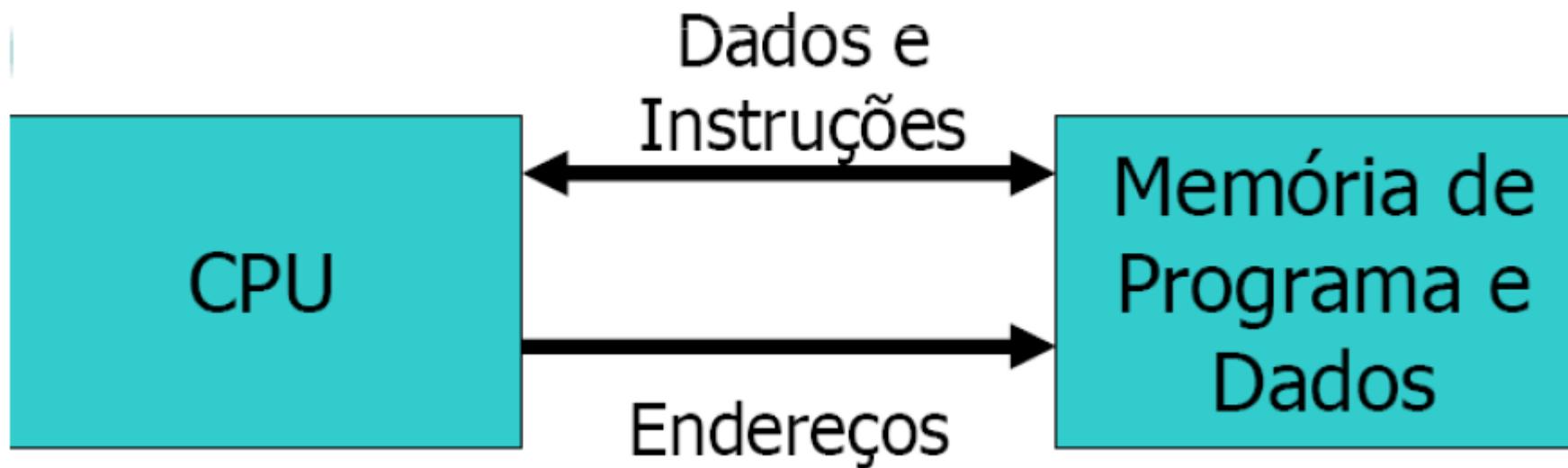
► **Flash**

- Usada para armazenar o programa de Usuário;
- Não-volátil;
- Geralmente é rápida;
- É gravada e apagada através de um dispositivo de programação;

Arquitetura Básica

► Arquitetura Von-Neumann

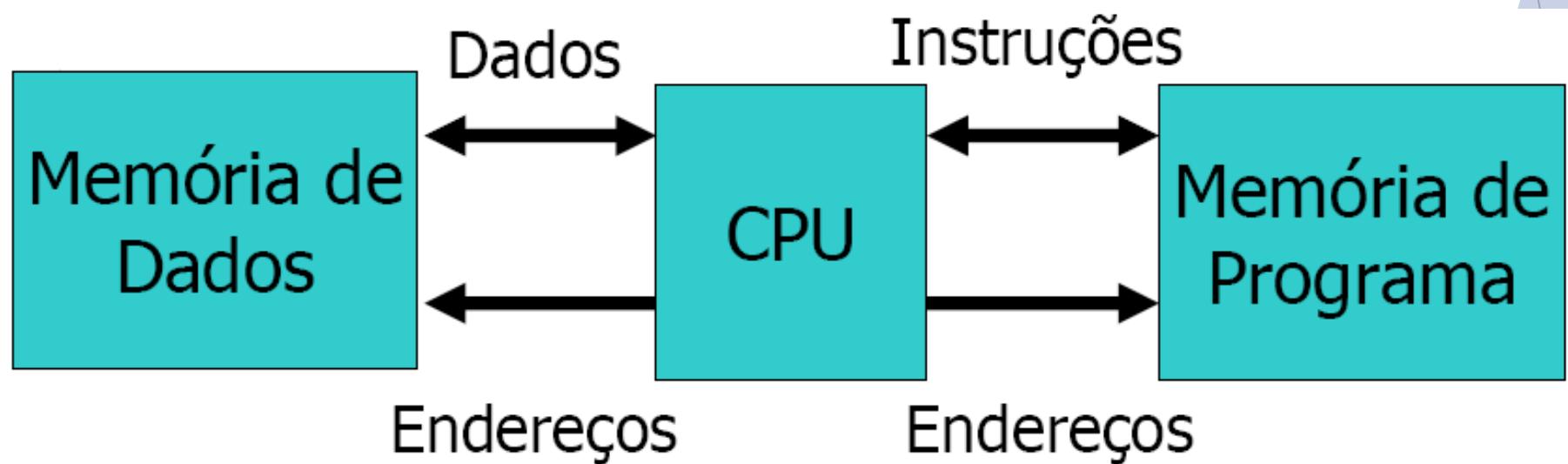
- Memória de programa e a memória de dados compartilham um único espaço de endereçamento;



Arquitetura Básica

► Arquitetura Harvard

- Existe um barramento para acessar instruções e outro para acessar dados de tal forma que as leituras de instruções e dados ocorrem paralelamente.
- Permite acessos simultâneos a memória de dados e de programa;

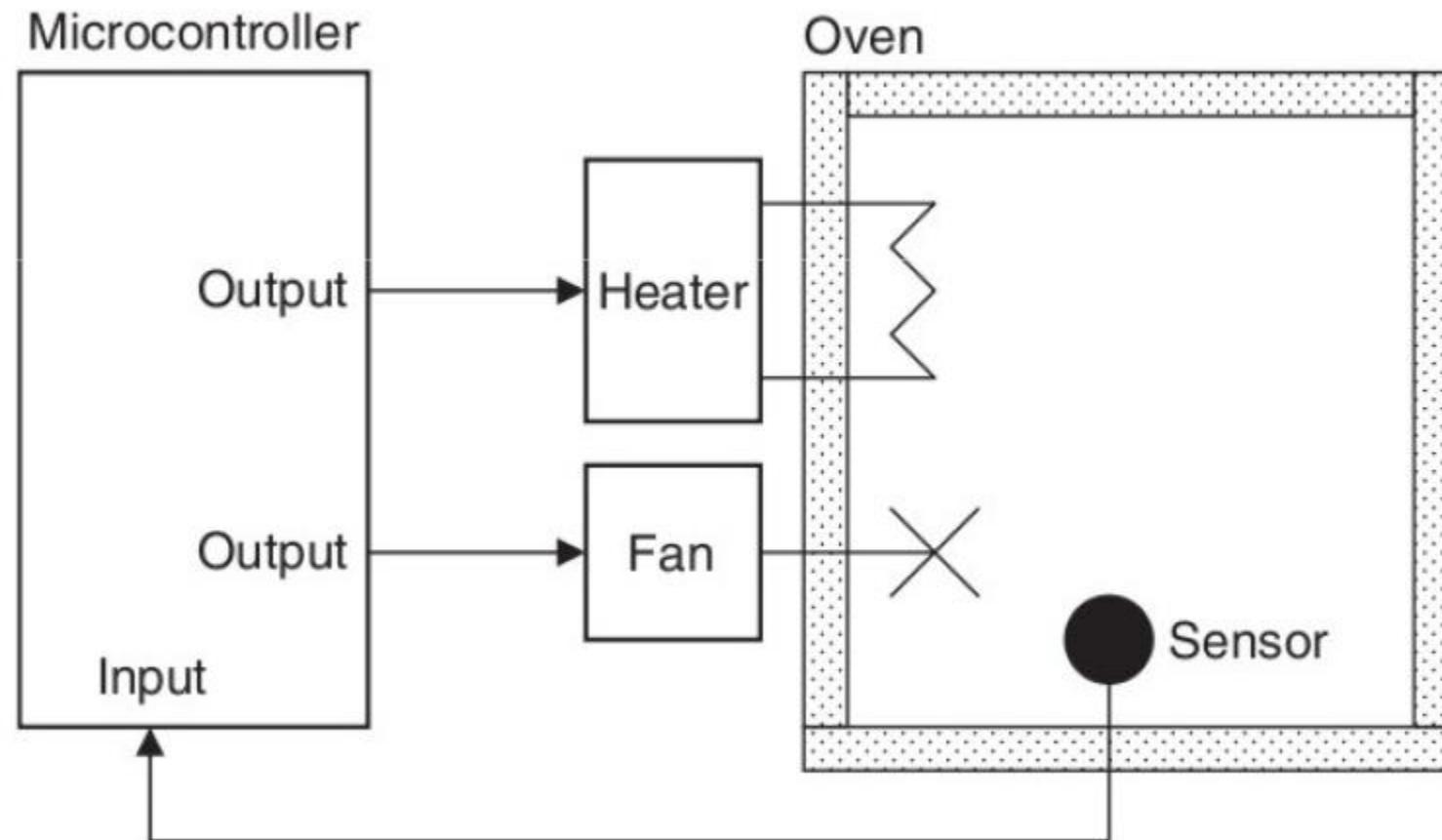


Arquitetura Básica

- ▶ **CISC (Computador com um conjunto complexo de instruções):**
 - ▶ Arquitetura Von-Neumann;
 - ▶ Grande número de instruções;
 - ▶ Menos Rápido;
 - ▶ Flexibilidade de programação;
- ▶ **RISC (Computador com um Conjunto Reduzido de Instruções):**
 - ▶ Arquitetura Harvard;
 - ▶ Pequeno número de instruções;
 - ▶ Mais Rápidas: Instruções levam um ciclo de clock interno para serem executada, exceto instruções de desvios;
 - ▶ A máquina RISC não possui geralmente hardware interno para operações de multiplicação e divisão;

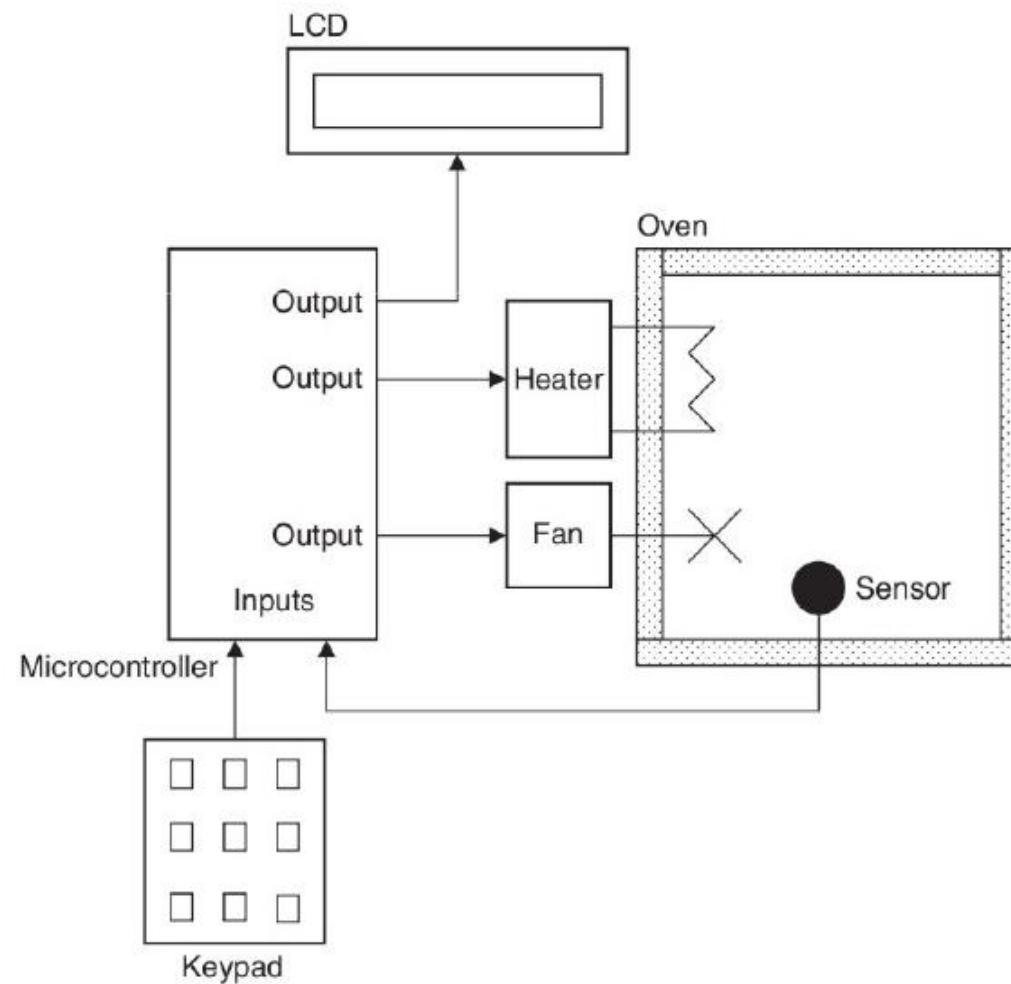
Exemplo de Aplicação

- ▶ Sistema de controle de temperatura de um forno



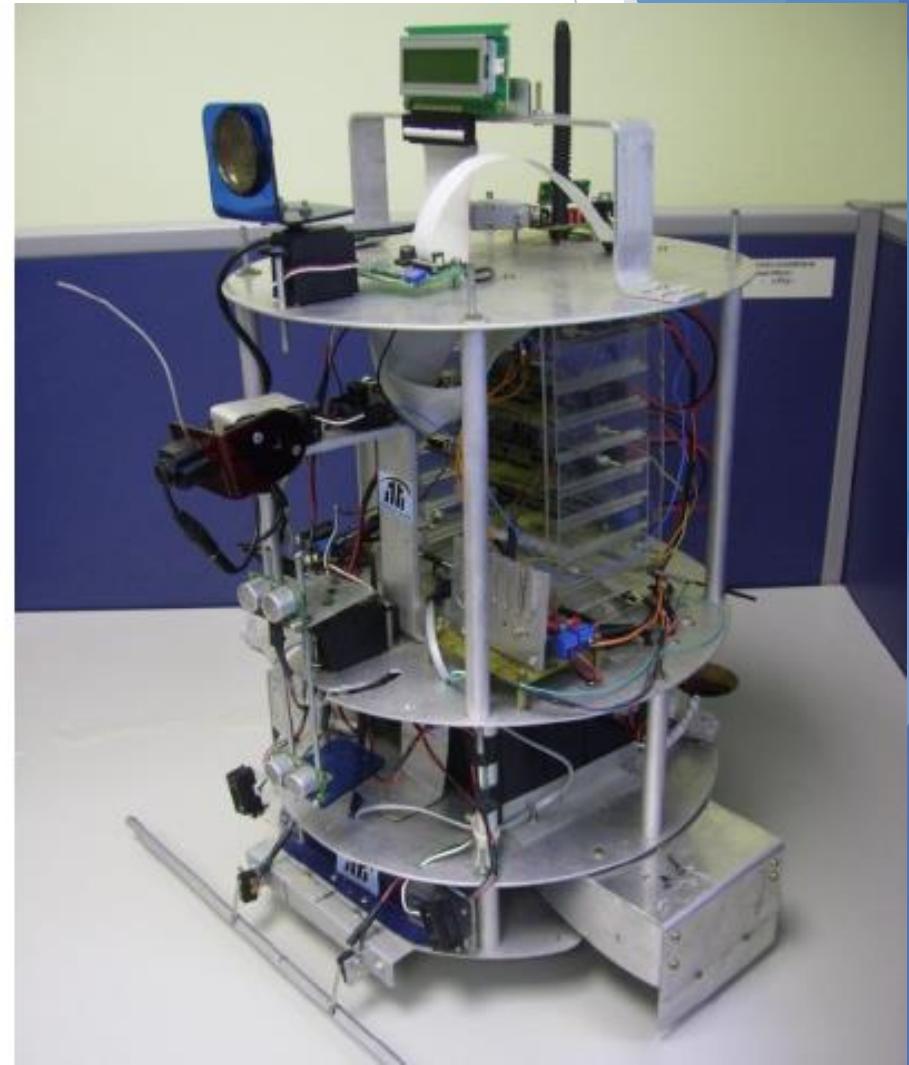
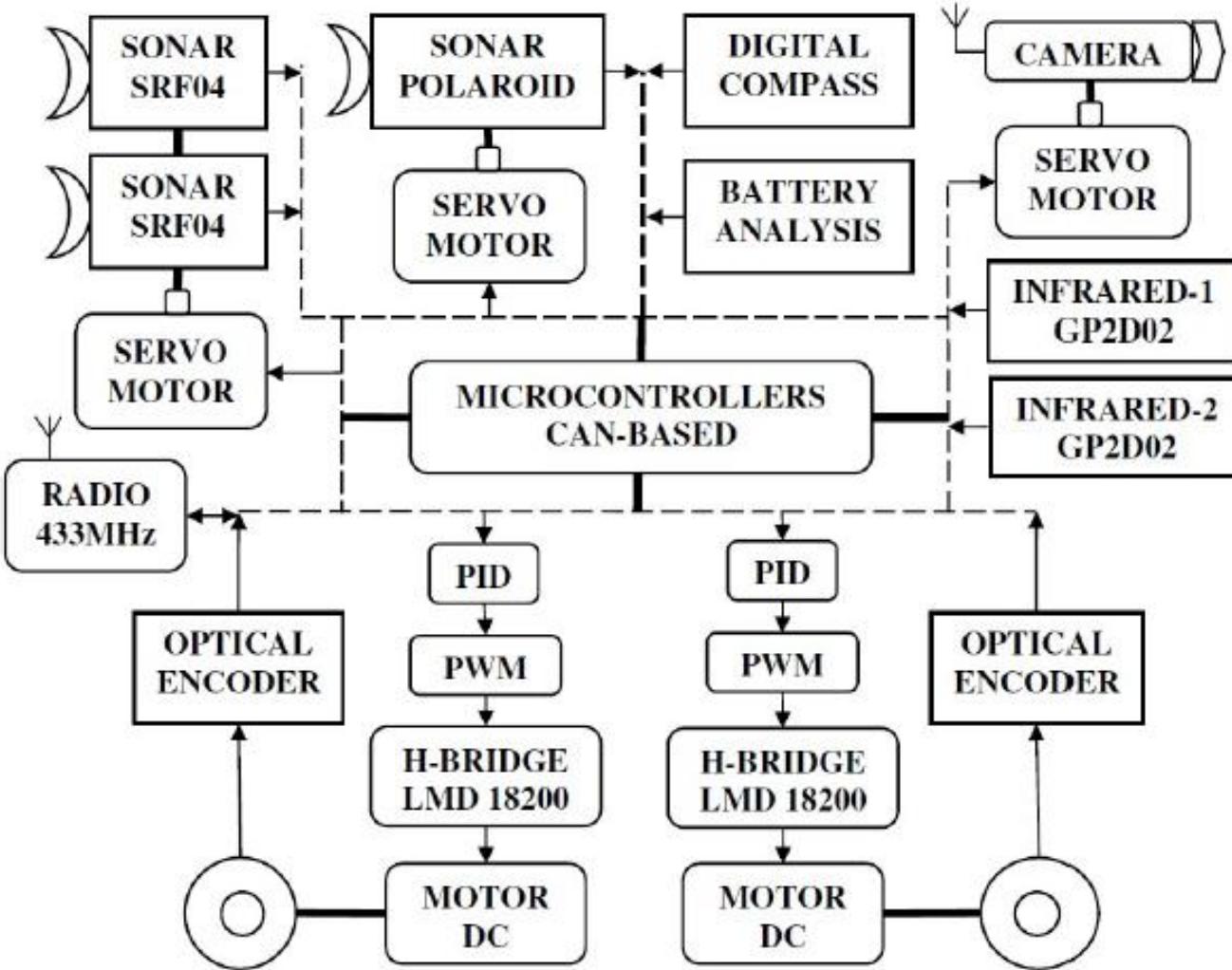
Exemplo de Aplicação

- ▶ Sistema de controle de temperatura de um forno

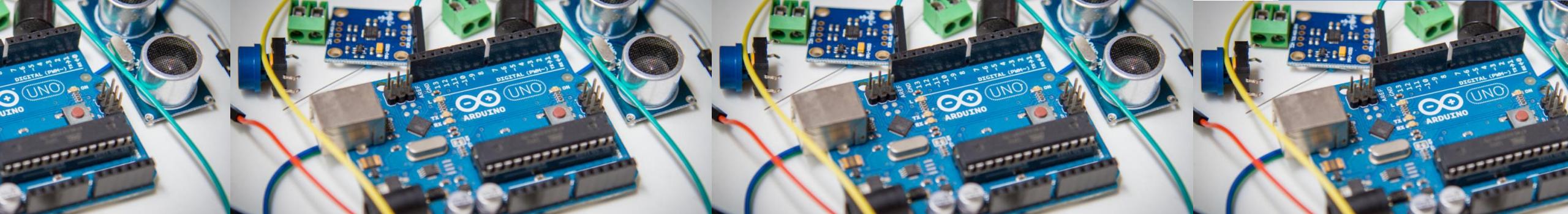


Exemplo de Aplicação

► Robótica



ARDUINO



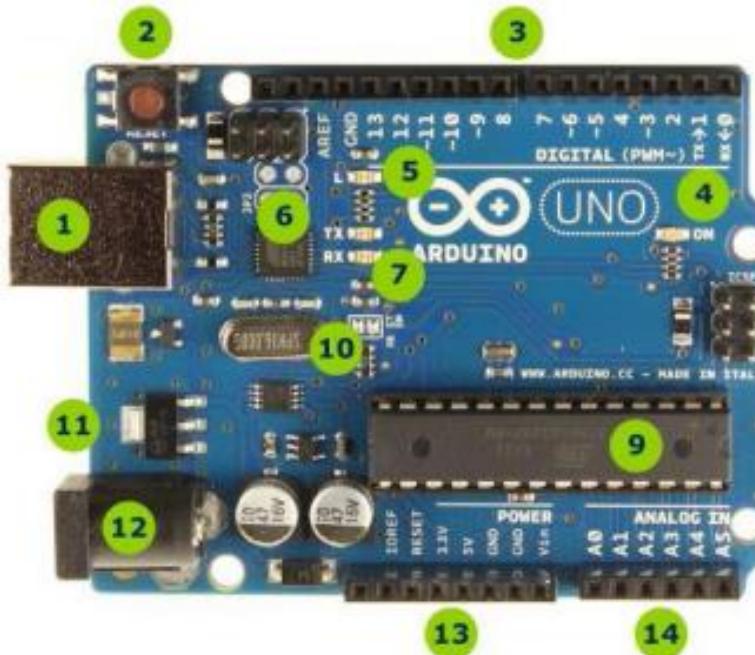
Introdução: O ARDUINO

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



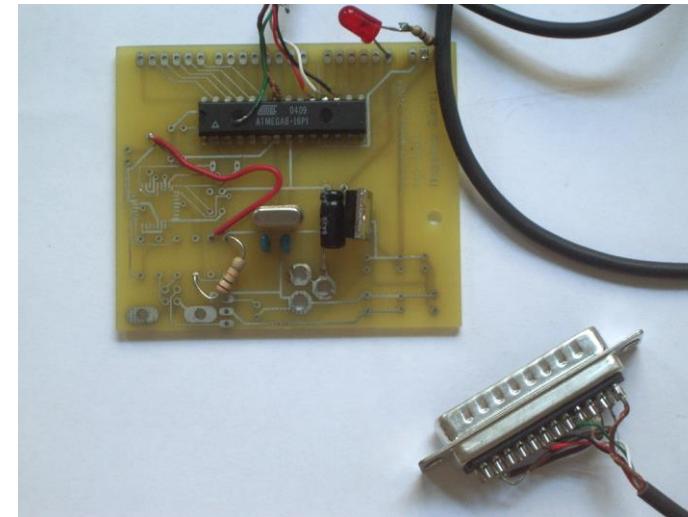
Introdução: Projeto ARDUINO – arquitetura e história



- 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.



Primeiro protótipo 2005

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

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



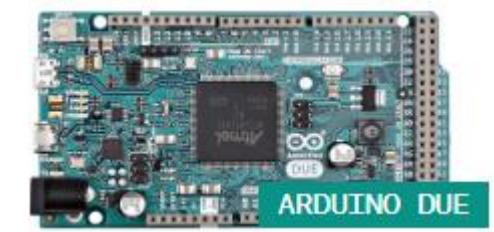
Microcontrolador	ATmega32U4	ATmega328
Tensão de operação	5V	5V
Tensão de alimentação	7-12V	
Pinos I/O digital	20	22
Pinos I/O PWM digital	7	6
Pinos analógicos	12	8
Corrente DC por pino I/O	20mA	40mA
Corrente DC por pino I/O de 3,3V	50mA	

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	<u>ATmega2560</u>	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



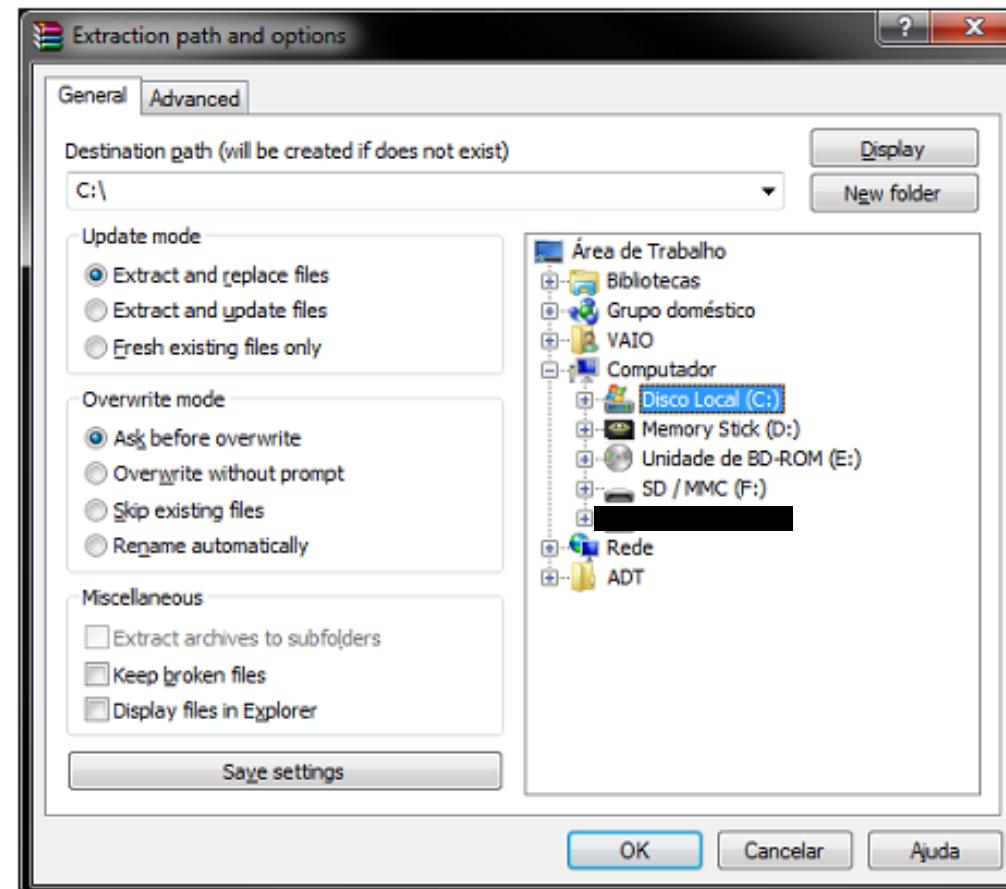
Download the Arduino IDE

A screenshot of the Arduino Software (IDE) download page. On the left, there is a large teal circle containing the Arduino logo (-∞+). To its right, the text "ARDUINO 1.8.9" is displayed. Below this, a paragraph describes the software as open-source and compatible with Windows, Mac OS X, and Linux. It also mentions that it runs on Java and is based on Processing. A note states that the software can be used with any Arduino board and refers to the "Getting Started" page for installation instructions. On the right side of the page, there is a vertical column of download links. The first two links, "windows Installer, for Windows XP and up" and "Windows ZIP file for non admin install", are circled in yellow. Below these, there are links for "Windows app Requires Win 8.1 or 10" (with a "Get" button), "Mac OS X 10.8 Mountain Lion or newer", and several Linux options: "Linux 32 bits", "Linux 64 bits", "Linux ARM 32 bits", and "Linux ARM 64 bits". At the bottom of this column are links for "Release Notes", "Source Code", and "Checksums (sha512)".

Ambiente de programação

- Ambiente integrado de Desenvolvimento (IDE)

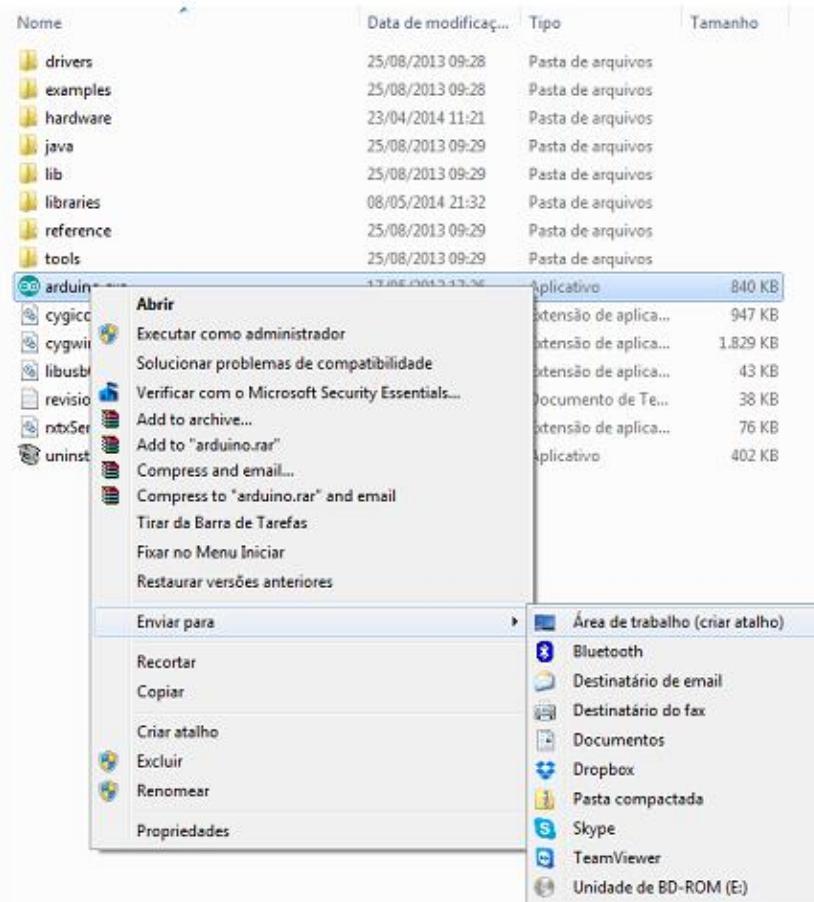
Quando finalizar o download, descompacte a pasta no diretório: **C:** conforme apresentado na figura abaixo.



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!



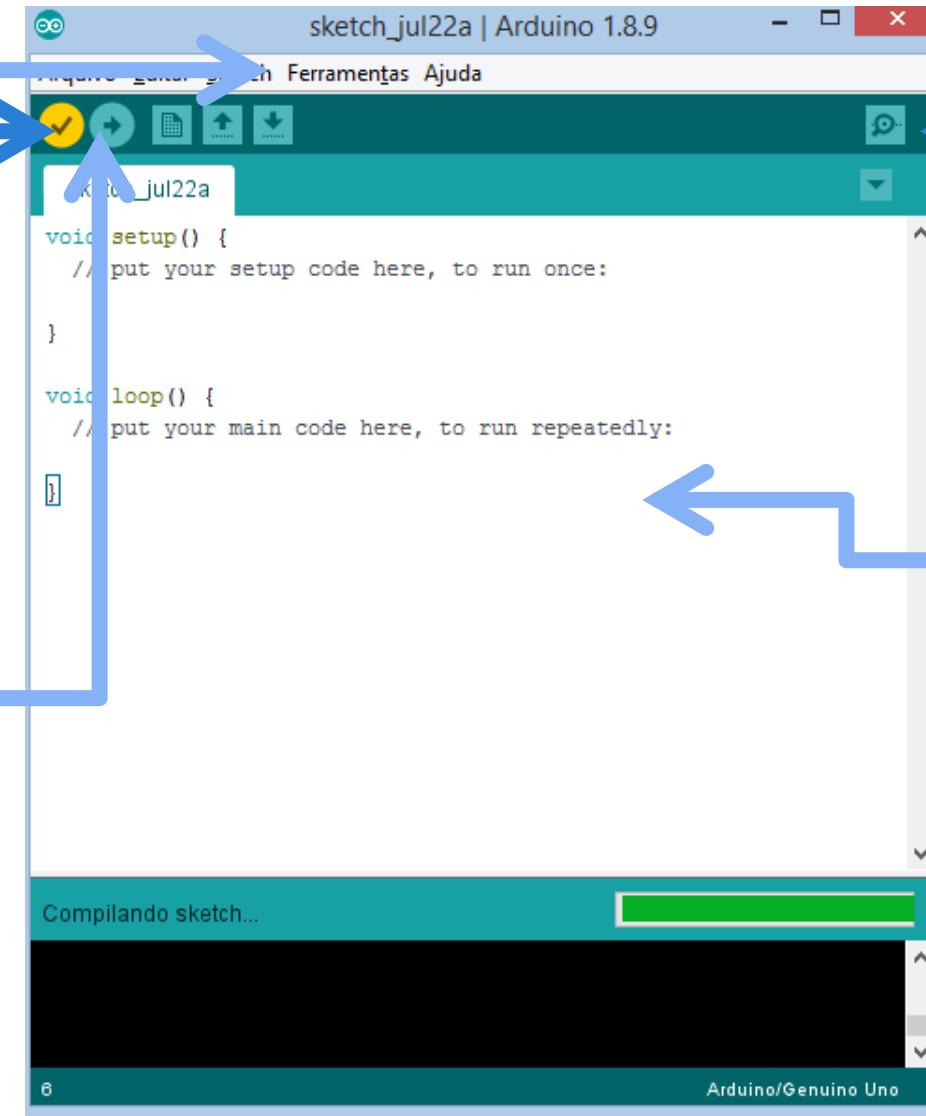
Ambiente de programação

► Ambiente integrado de Desenvolvimento (IDE)

Tools - seleciona o tipo de Arduino e a porta COM

Verify – compila a programação

Upload – envia a programação para o Arduino



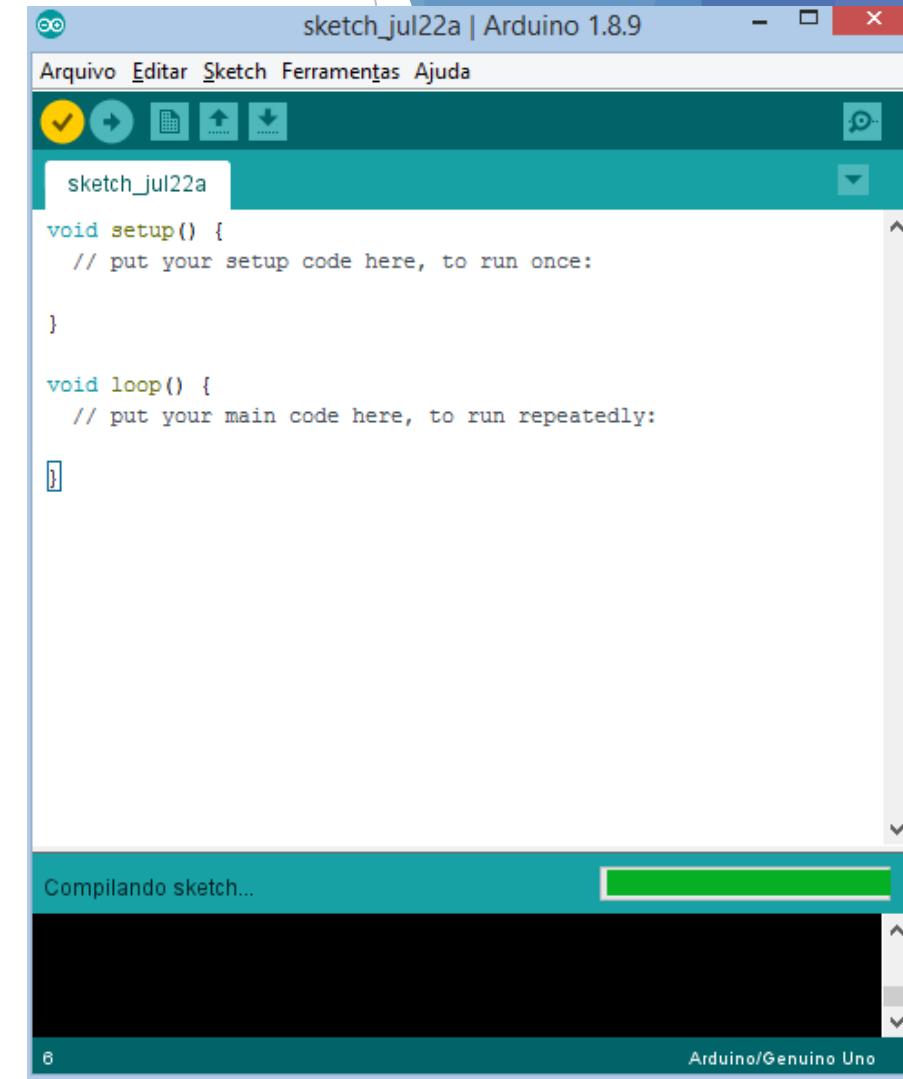
Serial Monitor – monitor de dados

Sketch – palco da programação

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.

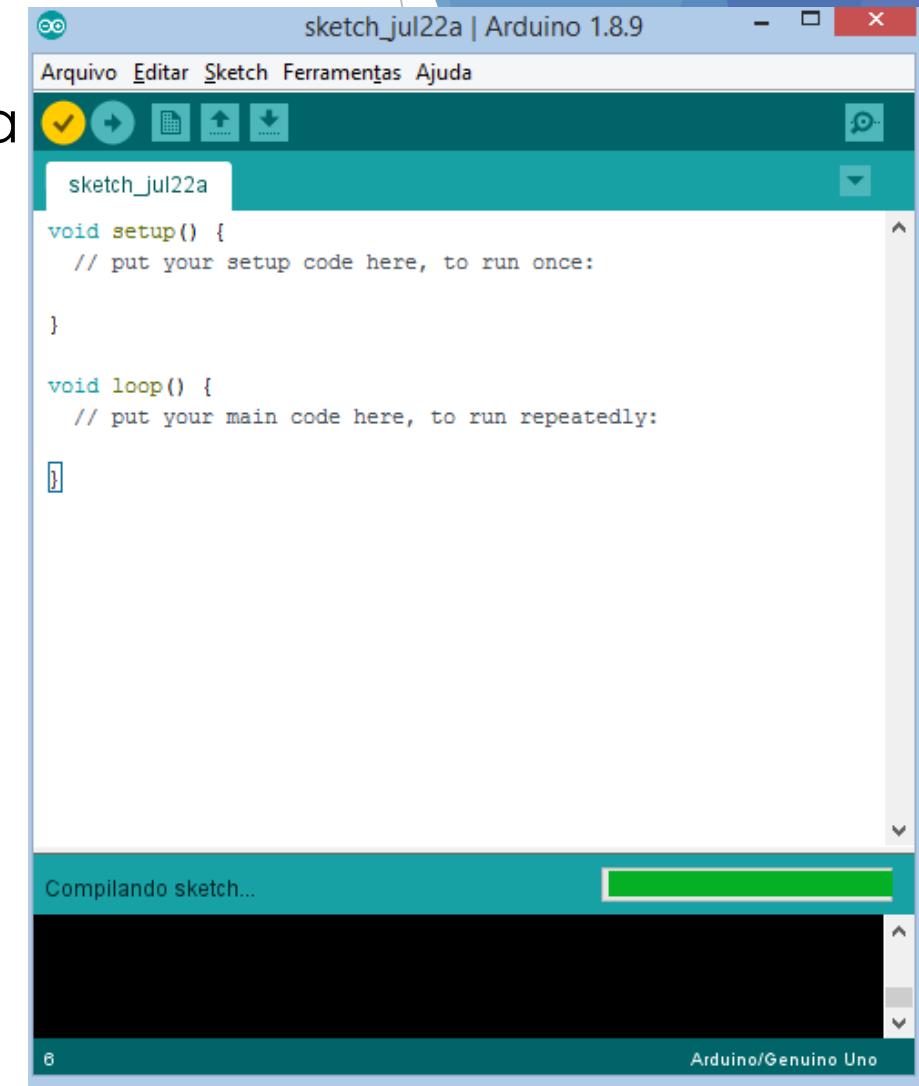
```
void setup() {  
    // initialize the LED pin as an output:  
    pinMode(ledPin, OUTPUT);  
    // initialize the pushbutton pin as an input:  
    pinMode(buttonPin, INPUT);  
}
```



Ambiente de programação

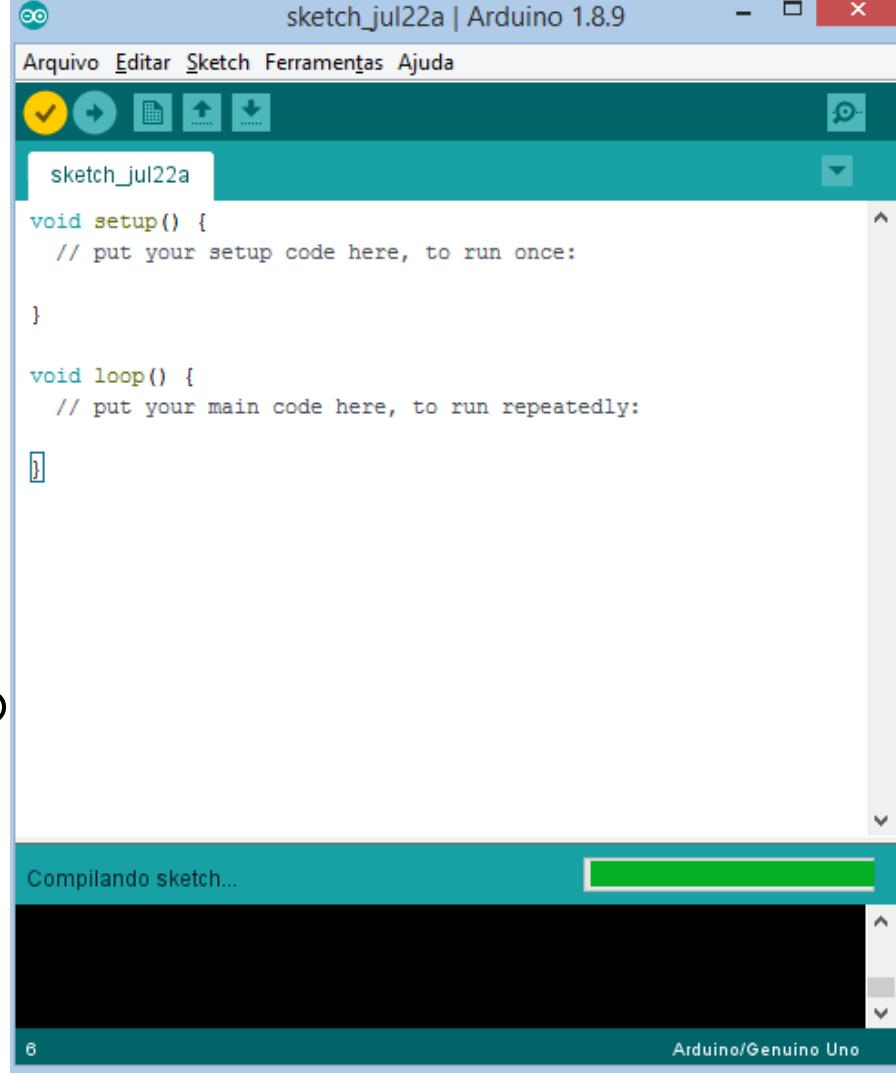
- ▶ A função **loop()** irá executar continuamente as instruções que estão lá até que outro sketch seja carregado na memória “flash” do Arduino.
- ▶ É importante notar que no Arduino é possível armazenar e executar um sketch por vez, desta forma, sempre quando transferimos um sketch esse irá substituir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



```
sketch_jul22a | Arduino 1.8.9
Arquivo Editar Sketch Ferramentas Ajuda
sketch_jul22a
void setup() {
  // put your setup code here, to run once:
}

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

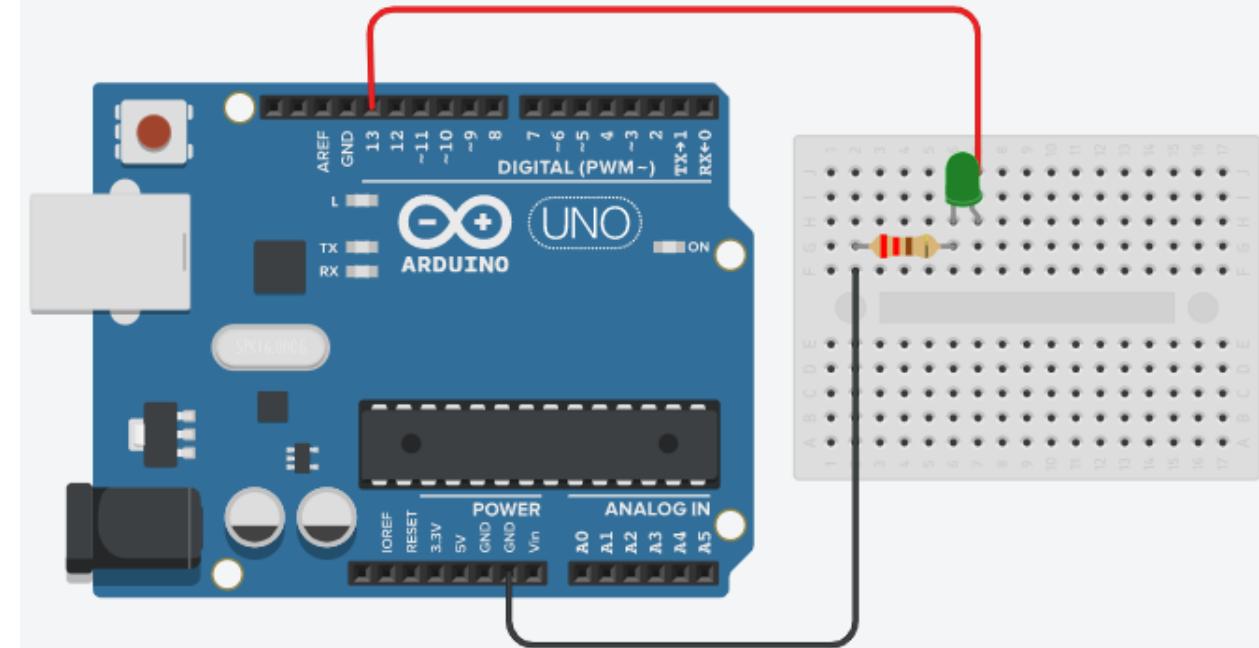
Compilando sketch...
Arduino/Genuino Uno
```

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.

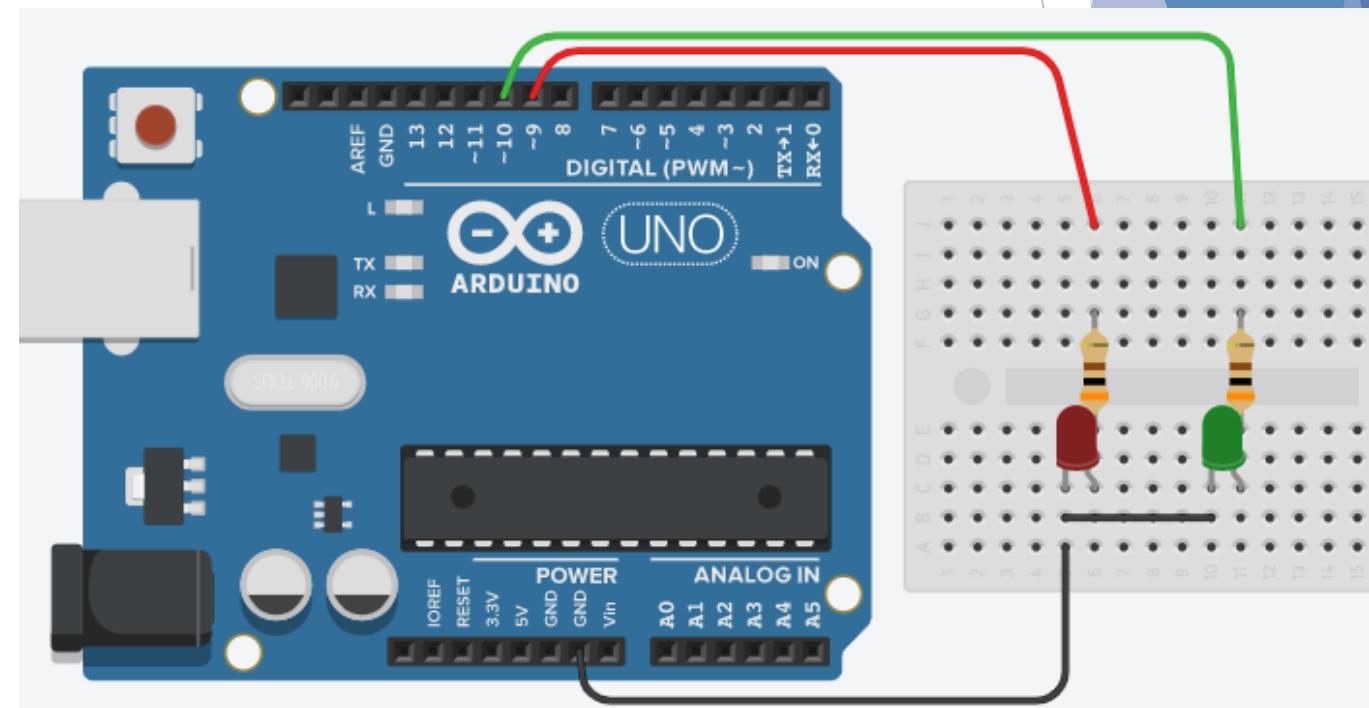


Mãos na massa! Piscar dois LEDs

O objetivo é piscar dois leds alternando e modificar o tempo.

► Material necessário:

- 1 Arduino;
- 2 Resistores 300 ohms (laranja, preto, marrom);
- 2 Leds (qualquer cor);
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Piscar dois LEDs

► Programa

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.

```
int Pisca1 = 0;
int Pisca2 = 0;
void setup()
{
    pinMode (10, OUTPUT);
    pinMode (9, OUTPUT);
}
void loop()
{
    Pisca1 = 2000;
    Pisca2 = 1000;
    digitalWrite (10, HIGH);
    delay (Pisca1);
    digitalWrite (10, LOW);
    delay (Pisca2);
    digitalWrite (9, HIGH);
    delay (Pisca1);
    digitalWrite (9, LOW);
    delay (Pisca2);
}
```

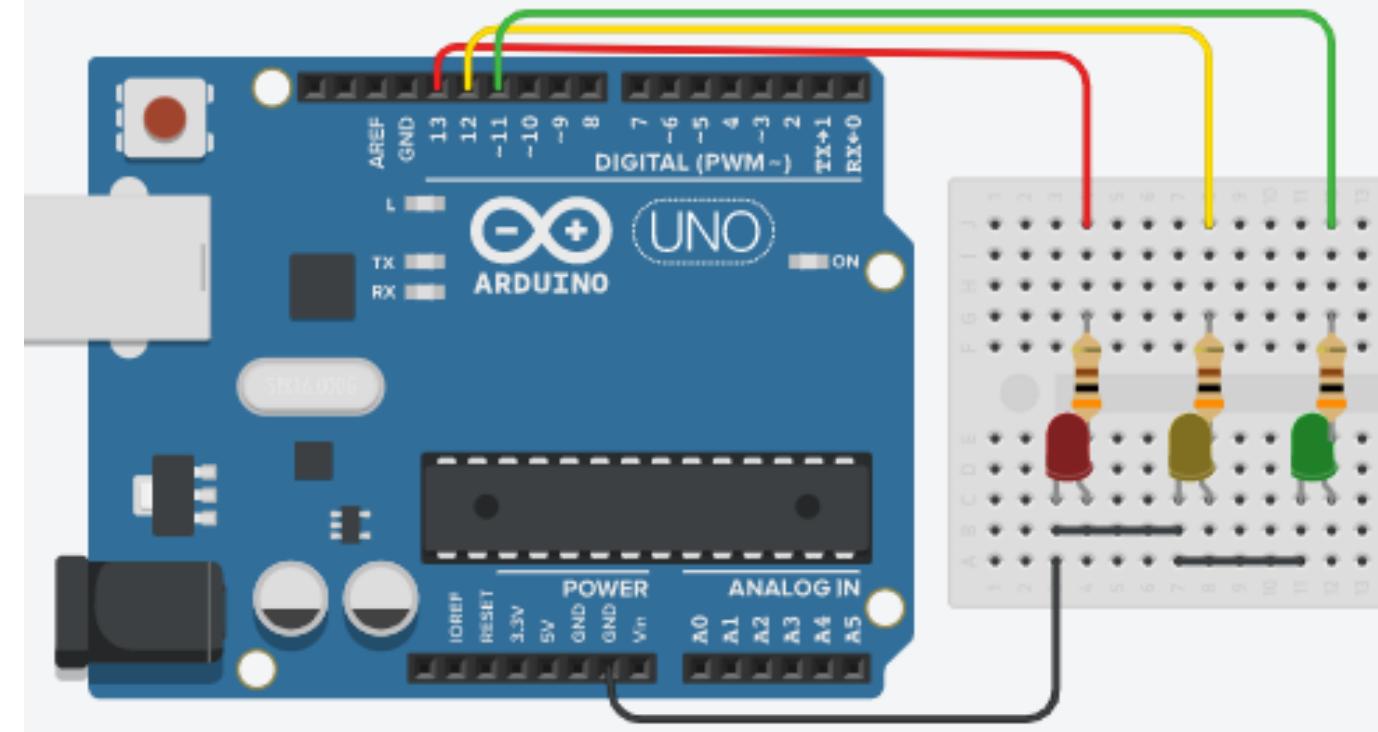


Mãos na massa! Semáforo

Reproduzir o funcionamento de um sinal de trânsito.

► Material necessário:

- 1 Arduino;
- 3 Resistores de 300 ohms (laranja, preto, marrom);
- 3 Leds ;
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Semáforo

► Programa

```
void setup()
{
    pinMode(13, OUTPUT); //define pino 13 como saída
    pinMode(12, OUTPUT); //define pino 13 como saída
    pinMode(11, OUTPUT); //define pino 13 como saída
}

void loop()
{
    digitalWrite(13, HIGH); // envia sinal 1 para o pino 13
    delay(2000); // aguarda 2 segundos
    digitalWrite(13, LOW); // envia sinal 0 para o pino 13
    digitalWrite(12, HIGH); // envia sinal 1 para o pino 12
    delay(1000); // aguarda 1 segundo
    digitalWrite(12, LOW); // envia sinal 0 para o pino 12
    digitalWrite(11, HIGH); // envia sinal 1 para o pino 11
    delay(2000); // aguarda 2 segundos
    digitalWrite(11, LOW); // envia sinal 0 para o pino 11
}
```

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.

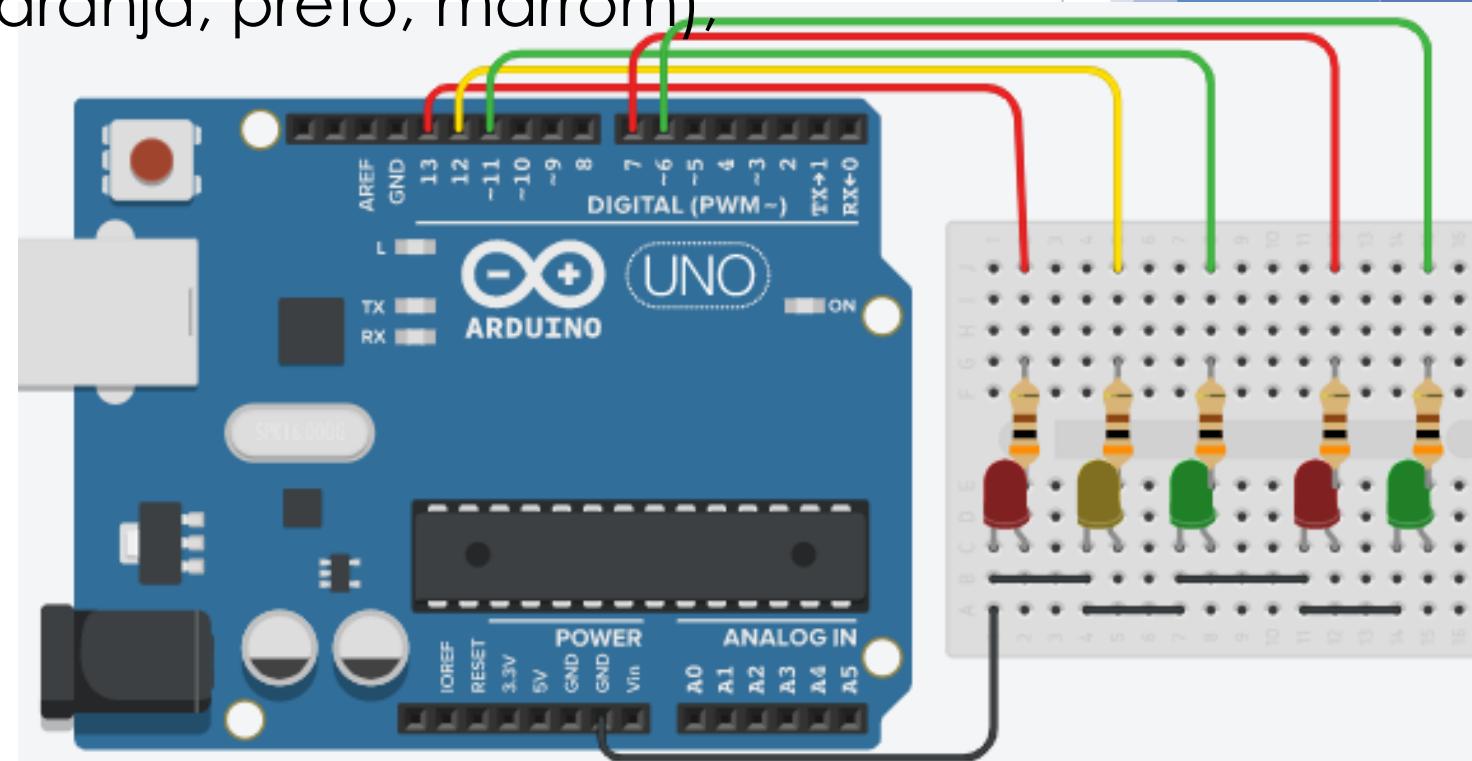


Mãos na massa! Semáforo completo

Reproduzir o funcionamento de um sinal de trânsito com o sinal de pedestres.

► Material necessário:

- 1 Arduino;
- 5 Resistores de 300 ohms (laranja, preto, marrom);
- 5 Leds;
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Semáforo completo

```
void setup()
{
    pinMode(13, OUTPUT);
    pinMode(12, OUTPUT);
    pinMode(11, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(6, OUTPUT);
}

void loop()
{
    digitalWrite(13, HIGH);
    digitalWrite(6, HIGH);
    delay(2000);
    digitalWrite(13, LOW);
    digitalWrite(12, HIGH);
    digitalWrite(6, LOW);
    delay(250);
    digitalWrite(6, HIGH);
    delay(250);
    digitalWrite(6, LOW);
    delay(250);
    digitalWrite(6, HIGH);
    delay(250);
    digitalWrite(6, LOW);
    delay(250);
    digitalWrite(6, HIGH);
    delay(500);
}

digitalWrite(12, LOW);
digitalWrite(6, LOW);
digitalWrite(7, HIGH);
digitalWrite(11, HIGH);
delay(2000);
digitalWrite(11, LOW);
digitalWrite(7, LOW);
delay(1000);
}
```

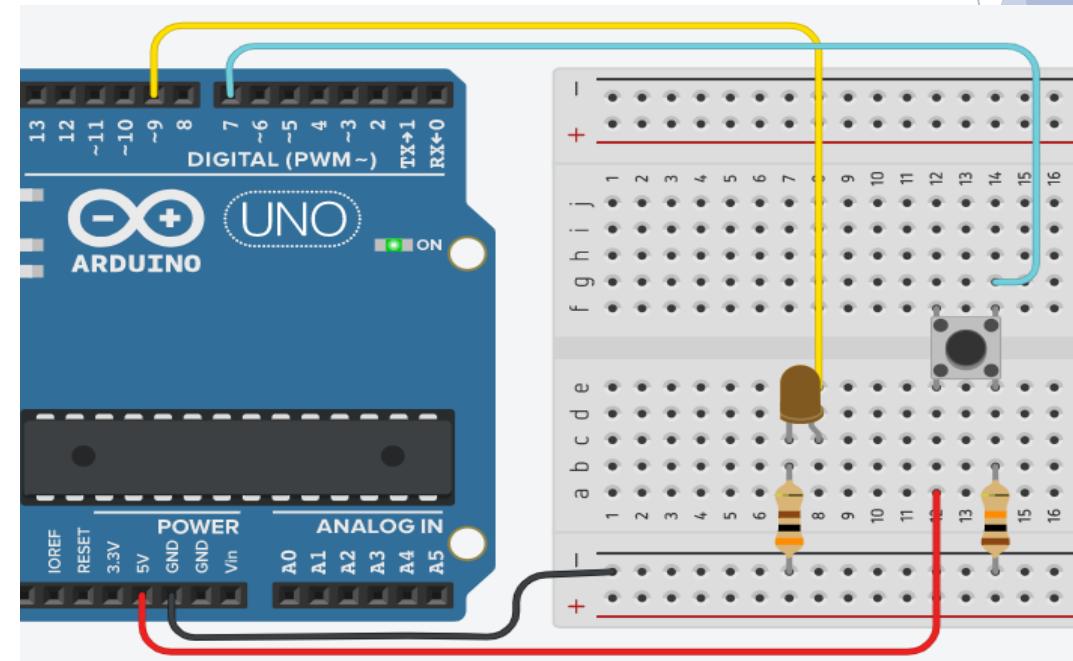


Mãos na massa! Ligar o LED

O objetivo deste projeto é utilizar um botão para acender, apagar o LED

► Material necessário:

- 1 Arduino;
- 1 Resistor de 300 ohms (laranja, preto, marrom);
- 1 Resistor de 10k ohms (marrom, preto laranja) para o botão;
- 1 Led (qualquer cor);
- 1 push button
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Ligar o LED

- ▶ Programa: neste caso o LED permanece aceso apenas quando o botão está pressionado

```
int valor;  
  
void setup() {  
    pinMode(9, OUTPUT); // Definir o pino como saída  
    pinMode(7, INPUT); // Definir o pino com entrada  
}  
  
void loop() {  
    valor = digitalRead(7);  
    digitalWrite(9, valor);  
    delay (500);  
}
```

digitalRead: Lê o valor de um pino digital especificado

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.



Mãos na massa! Ligar o LED

- Programa: neste caso o LED acende ao pressionar o botão e permanece aceso até quando o botão for pressionado novamente

```
int valor;
int anterior = 0;
int estado = LOW;
void setup() {
    pinMode(9, OUTPUT);
    pinMode(7, INPUT);
}
void loop() {
    valor = digitalRead(7);
    if (valor == HIGH && anterior == LOW) {
        estado = !estado;
    }
    digitalWrite(9, estado);
    anterior = valor;
    delay (50);
}
```

if: Condição para realizar a etapa

=! Diferente de

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.

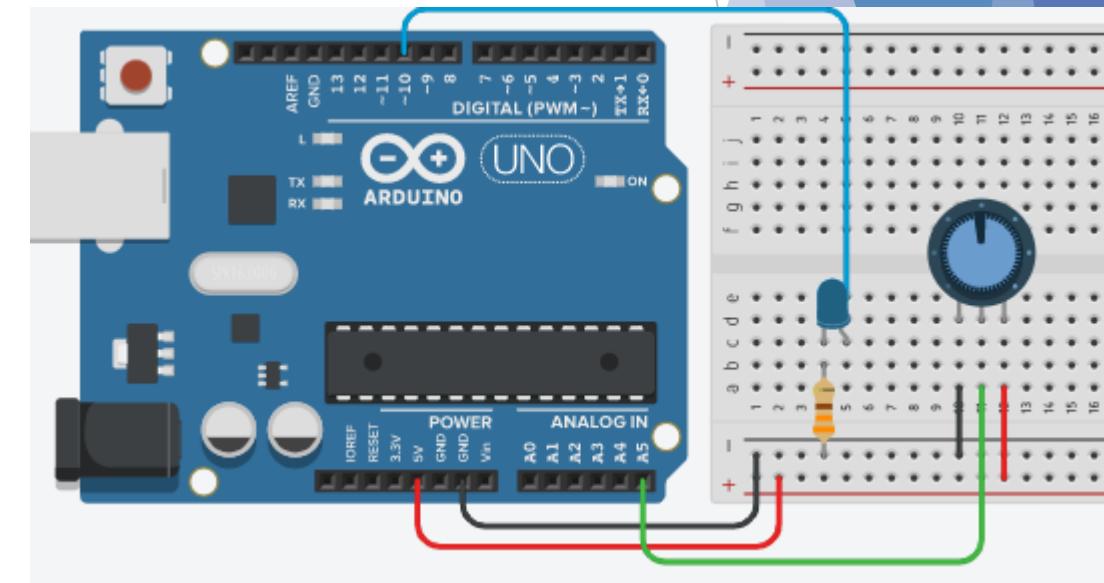


Mãos na massa! Controle de luminosidade

Neste projeto podemos controlar a luminosidade do Led usando um potenciômetro

► Material necessário:

- 1 Arduino;
- 1 resistor de 300 ohms (laranja, preto, marrom);
- 1 LED
- 1 Potenciômetro 10k
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Controle de luminosidade

► Programa:

```
int valorpot = 0;  
float luminosidade = 0;  
void setup()  
{  
    Serial.begin(9600); // Inicializa a serial  
    pinMode(10, OUTPUT); // Define o pino do led como saída  
    pinMode(5, INPUT); // Define o pino do potenciômetro como entrada  
}  
void loop()  
{  
    valorpot = analogRead(5); // Le o valor analógico  
    luminosidade = map(valorpot, 0, 1023, 0, 255); /* Converte e atribui  
    para a variável "luminosidade" o valor lido do potenciômetro*/  
  
    Serial.print("Valor lido do potenciômetro : ");  
    // Mostra o valor lido do potenciômetro no monitor serial  
    Serial.print(valorpot);  
    Serial.print(" = Luminosidade : ");  
    Serial.println(luminosidade); // Mostra o valor da luminosidade no monitor serial  
    // Envia sinal analógico para a saída do led, com luminosidade variável  
    analogWrite(10, luminosidade);  
}
```

PWM Pulse Width Modulation –
Modulação por Largura de Pulso

Reamapeia um número de um intervalo para
outro

Sintaxe da função map:

map (valor, menor_valor, maior_valor,
novo_menor_valor, novo_maior_valor)

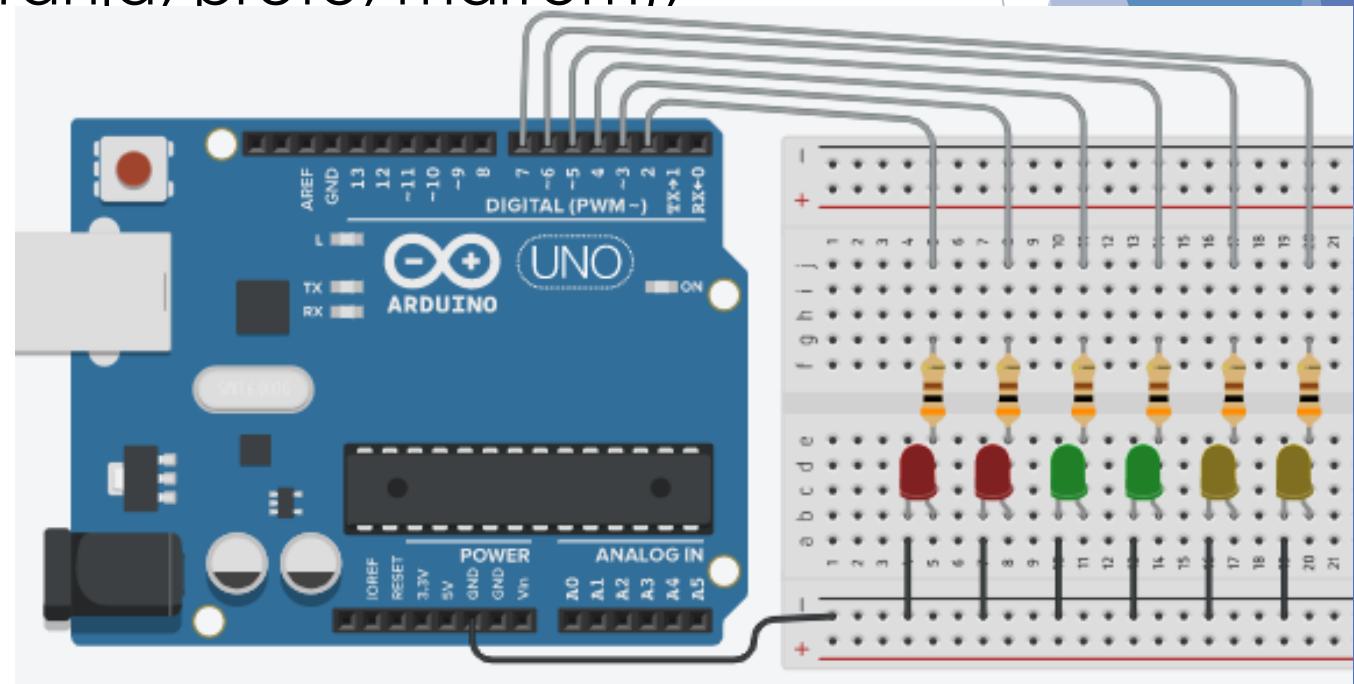


Mãos na massa! Faixa de luz

Neste projeto podemos vamos acionar os LEDs sequencialmente

► Material necessário:

- 1 Arduino;
- 6 resistores de 300 ohms (laranja, preto, marrom);
- 6 LEDs
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Faixa de luz

► Programa:

**for (inicialização; condição;
incremento):** laço de repetição,
com condição

```
int pinLed [6] = {2, 3, 4, 5, 6, 7};  
  
int Led;  
  
void setup() {  
    int x;  
    for (x = 0; x <= 5; x++) {  
        pinMode (pinLed [x], OUTPUT);  
    }  
}  
  
void loop() {  
    for(Led = 0; Led<=5; Led++){  
        digitalWrite (pinLed [Led], HIGH);  
        delay(100);  
    }  
    for(Led = 5; Led>=0; Led--){  
        digitalWrite (pinLed [Led], LOW);  
        delay(200);  
    }  
}
```

Vetor (array): é uma coleção de variáveis que são acessadas com um número índice

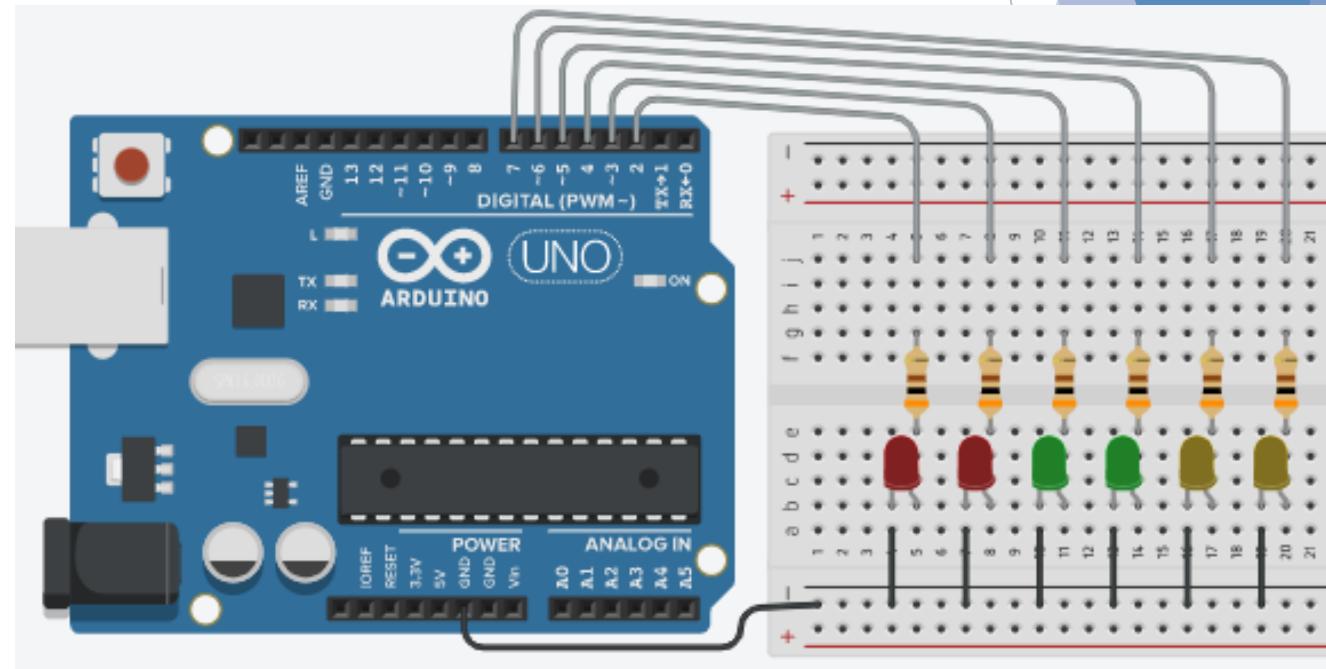


Mãos na massa! Sorteando a cor

Vamos testar sua sorte, tente sortear a cor verde!

► Material necessário:

- 1 Arduino;
- 6 resistores de 300 ohms (laranja, preto, marrom);
- 6 LEDs
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Sorteando a cor

► Programa:

```
int pinLed [6] = {2,3,4,5,6,7};
int Led;
char pausa;

void setup() {
  int x;
  for (x = 0; x <= 5; x++) {
    pinMode (pinLed [x],OUTPUT);
  }
  Serial.begin(9600);
}
```

```
void loop() {
  pausa = ' ';
  Led = 0;
  while (pausa != 'P') {
    digitalWrite (pinLed [Led], LOW);
    Led++;
    if(Led > 5) {
      Led = 0;
    }
    digitalWrite (pinLed [Led], HIGH);

    if (Serial.available()) {
      pausa = Serial.read();
    }
    delay(50);
    delay (5000);
}
```



Serial.available(): enviar resposta apenas quando receber dados

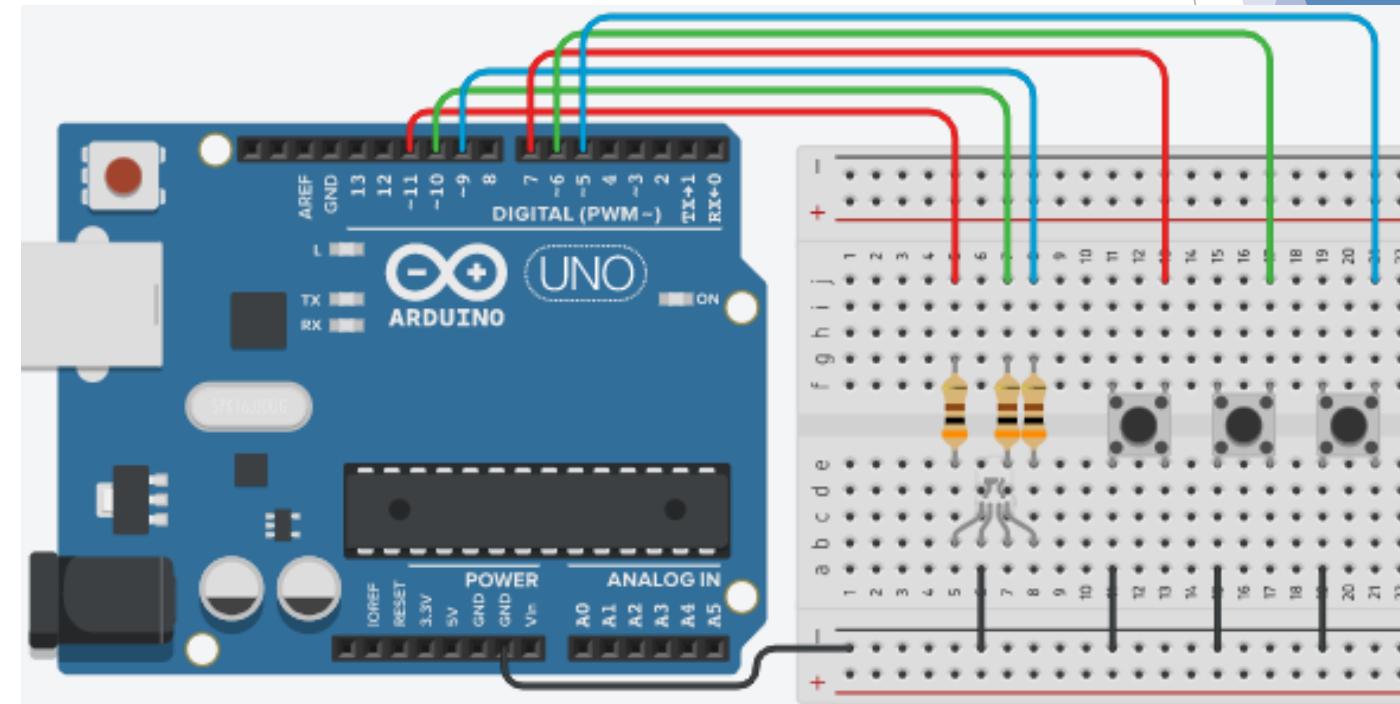
Serial.read(): Lê dados recebidos na porta serial

Mãos na massa! Então é natal..!

Essas luzes coloridas nos lembra do clima natalino, vamos aprender a produzi-las.

► Material necessário:

- 1 Arduino;
- 1 LED RGB
- 3 Resistor 300Ω
- 3 PushButton
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Então é natal..!

```
int btR;  
int btG;  
int btB;  
  
void setup() {  
    pinMode(11,OUTPUT);  
    pinMode(10,OUTPUT);  
    pinMode(9,OUTPUT);  
    pinMode(7,INPUT_PULLUP);  
    pinMode(6,INPUT_PULLUP);  
    pinMode(5,INPUT_PULLUP);  
}  
  
void loop() {  
    btR = digitalRead(7); //le estado dos botões  
    btG = digitalRead(6);  
    btB = digitalRead(5);  
  
    if(btR == LOW) { // verifica se o botão foi pressionado  
        analogWrite(11, 255); // aciona a cor com brilho máximo  
    } else { // senão  
        analogWrite(11, 0); // apaga o LED  
    }  
  
    if(btG == LOW) { // verifica se o botão foi pressionado  
        analogWrite(10, 255); // aciona a cor com brilho máximo  
    } else { // senão  
        analogWrite(10, 0); // apaga o LED  
    }  
  
    if(btB == LOW) { // verifica se o botão foi pressionado  
        analogWrite(9, 255); // aciona a cor com brilho máximo  
    } else { // senão  
        analogWrite(9, 0); // apaga o LED  
    }  
  
    delay(100); // aguarda para nova leitura  
}
```

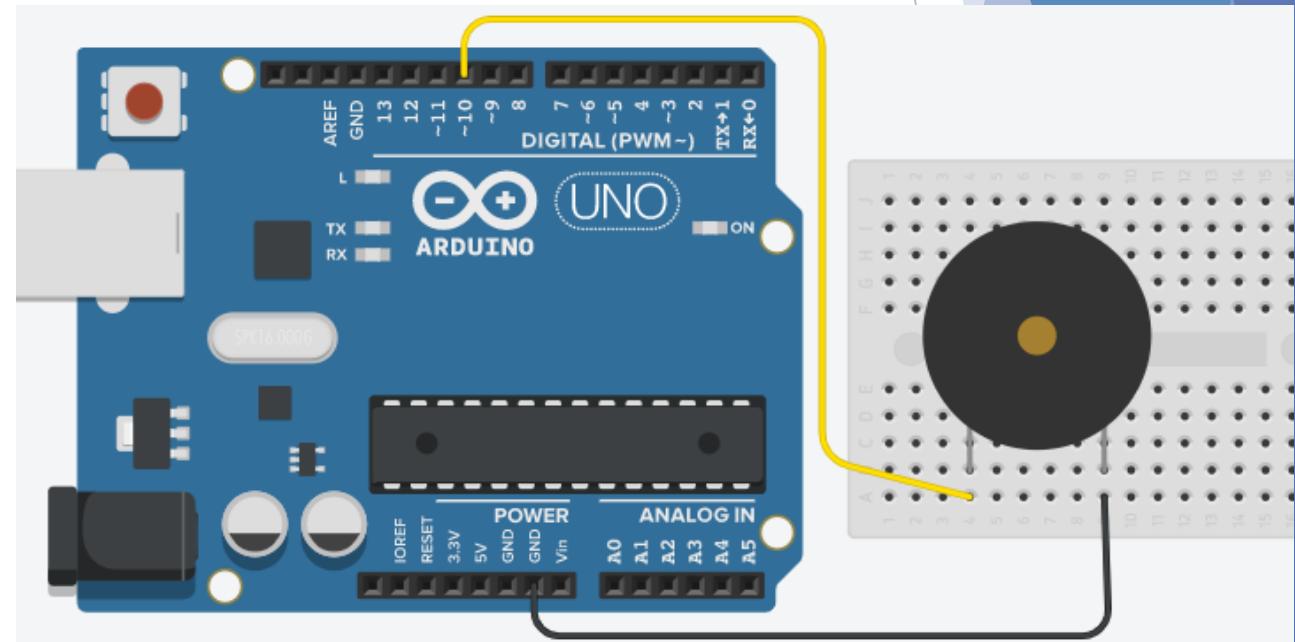


Mãos na massa! Lá vem a polícia

Nessa atividade vamos desenvolver uma sirene

► Material necessário:

- 1 Arduino;
- 1 buzzer 5V
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Lá vem a polícia

#define: permite dar um nome a um valor constante antes do programa ser compilado

tone (pino, frequência, duração): Gera uma onda quadrada na frequência especificada em um pino

```
#define tempo 10
int frequencia = 0;
int Pinofalante = 10;

void setup()
{
    pinMode(Pinofalante,OUTPUT); //Pino do buzzer
}
void loop()
{
    for (frequencia = 150; frequencia < 1800; frequencia += 1)
    {
        tone(Pinofalante, frequencia, tempo);
        delay(1);
    }
    for (frequencia = 1800; frequencia > 150; frequencia -= 1)
    {
        tone(Pinofalante, frequencia, tempo);
        delay(1);
    }
}
```

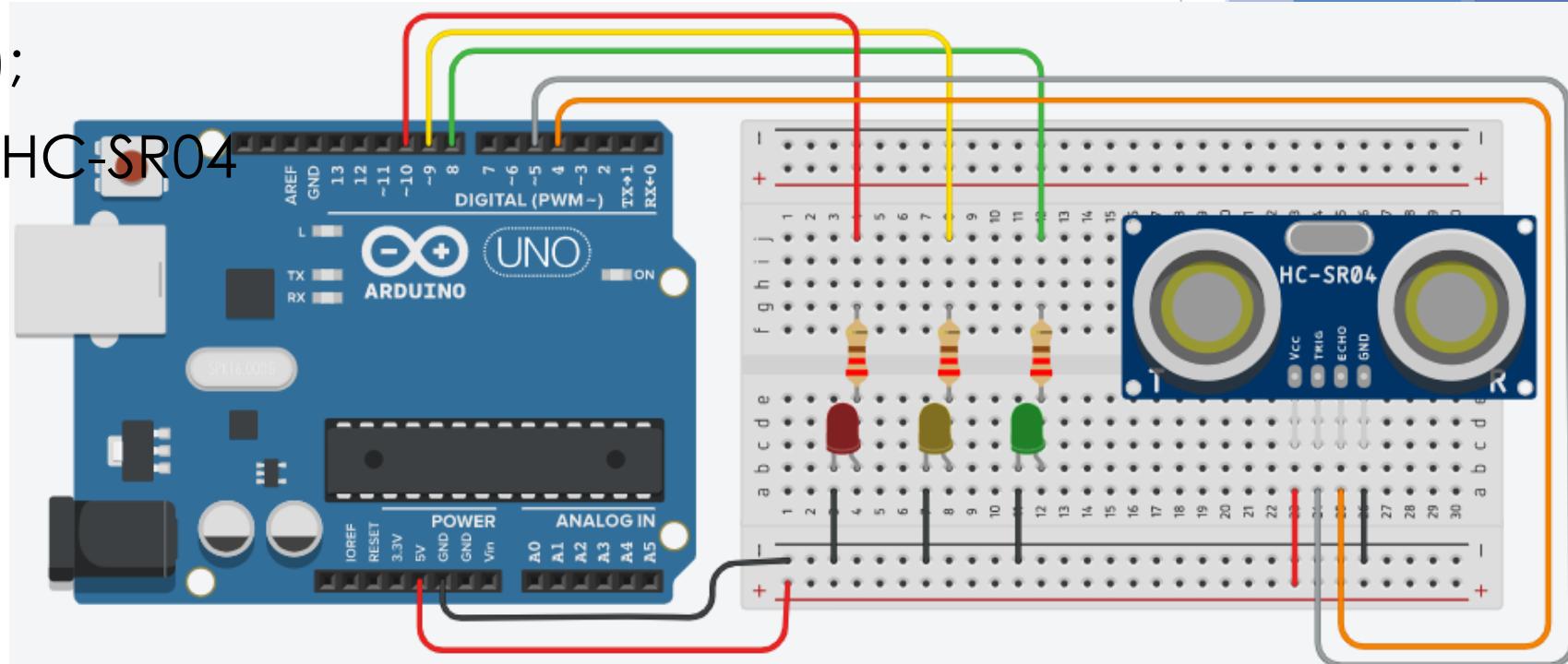


Mãos na massa! Sensor de proximidade

Desenvolver um sensor que indica quando o objeto está se aproximando.

► Material necessário:

- 1 Arduino;
- 3 Resistores 300 ohms (laranja, preto, marrom);
- 3 Leds (qualquer cor);
- 1 Sensor ultrassônico HC-SR04
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Sensor de proximidade

Programa

```
long readUltrasonicDistance(int triggerPin, int echoPin)
{
    pinMode(triggerPin, OUTPUT); // define pino TRIGGER saída
    digitalWrite(triggerPin, LOW); // limpa o trigger
    delayMicroseconds(2); // aguarda
    digitalWrite(triggerPin, HIGH); // envia sinal alto no trigger
    delayMicroseconds(10); // aguarda
    digitalWrite(triggerPin, LOW);
    pinMode(echoPin, INPUT); // define o pino ECO como entrada
    return pulseIn(echoPin, HIGH); // retorna o som do pino ECO
}
```

long: são variáveis de tamanho extendido para armazenamento de números



```
void setup()
{
    Serial.begin(9600);
    pinMode(10, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(8, OUTPUT);
}

void loop()
{
    Serial.println(0.01723 * readUltrasonicDistance(5, 4));
    if (0.01723 * readUltrasonicDistance(5, 4) < 10) {
        digitalWrite(10, HIGH);
        digitalWrite(9, LOW);
        digitalWrite(8, LOW);
    } else {
        if (0.01723 * readUltrasonicDistance(5, 4) < 20) {
            digitalWrite(10, LOW);
            digitalWrite(9, HIGH);
            digitalWrite(8, LOW);
        } else {
            digitalWrite(10, LOW);
            digitalWrite(9, LOW);
            digitalWrite(8, HIGH);
        }
    }
    delay(10); // atraso para aumentar a precisão
}
```

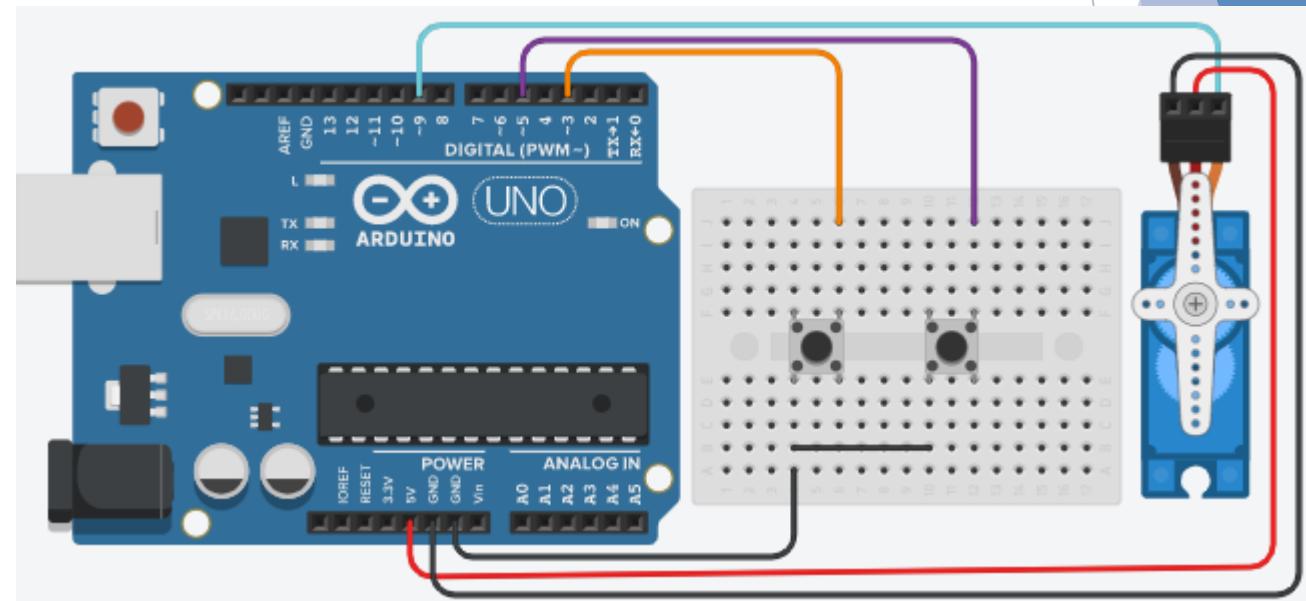


Mãos na massa! Direita ou Esquerda

O objetivo dessa atividade é controlar o servo usando posições pré-estabelecidas, o que é muito útil quando precisamos usar o motor para movimentos repetitivos

► Material necessário:

- 1 Arduino;
- 2 push buttons
- 1 servo motor sg90
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Direita ou Esquerda

A diretiva **#include** é usada para incluir bibliotecas externas ao seu sketch

```
#include <Servo.h>

Servo s;
int valor;
void setup()
{
    s.attach(9); // servo pino 9
    pinMode(3, INPUT); //Define o pino como entrada
    digitalWrite(3, HIGH);
    pinMode(5, INPUT);
    digitalWrite(5, HIGH);
}
void loop()
{
    valor=digitalRead(3); //Le o valor do botao
    if(valor!=1) //Caso o botao seja pressionado
    {
        s.write(10); //Move o servo para o angulo de 10 graus
        delay(15); } //Delay para o servo atingir a posicao
    valor=digitalRead(5);
    if(valor!=1)
    {
        s.write(120); //Move o servo para o angulo de 120 graus
        delay(15); }
}
```

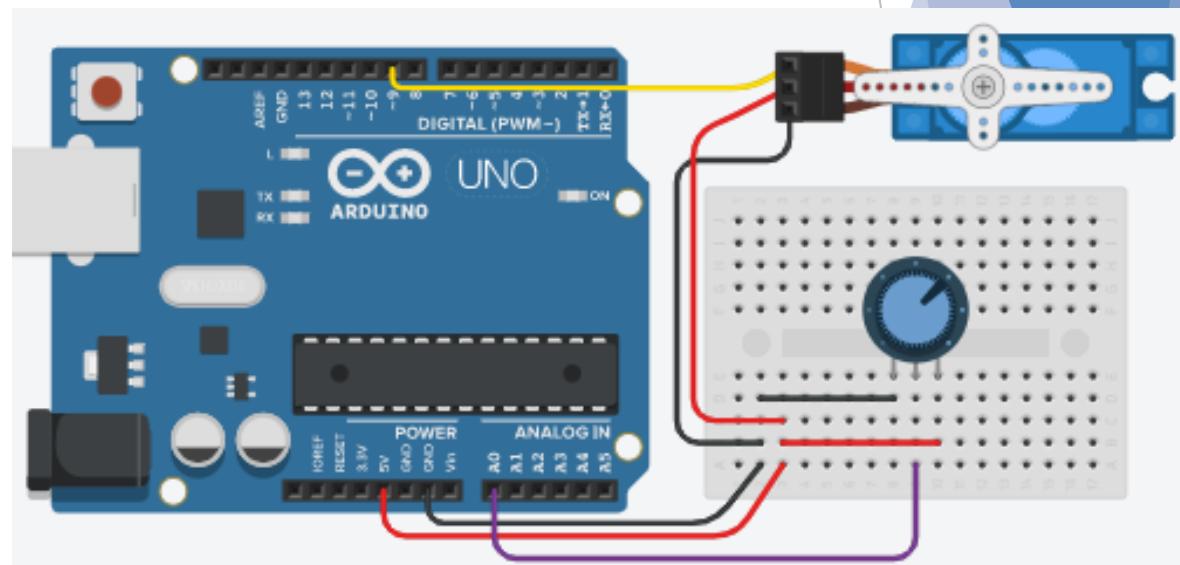


Mãos na massa! Rodando

O objetivo dessa atividade é controlar o servo usando apenas um potenciômetro

► Material necessário:

- 1 Arduino;
- 1 potenciômetro
- 1 servo motor sg90
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Rodando

```
#include <Servo.h>

Servo servo; //definir o nome do objeto
int potpin = 0; //pino analogico no potenciometro
int val; // variavel do pino analogico

void setup() {
    servo.attach(3); //conecta ("attaches") o servo no pino 3
    Serial.begin(9600);
}

void loop() {
    val = analogRead(potpin); // lê o valor analógico (0 a 1023)
    val = map(val, 0, 1023, 180, 0); // map define parâmetros para o servo (0 a 180)
    servo.write(val); // define a posição de acordo com a escala
    Serial.print("Ângulo atual:");
    Serial.println(val);
    delay(15); // aguarda o servo se mover
}
```

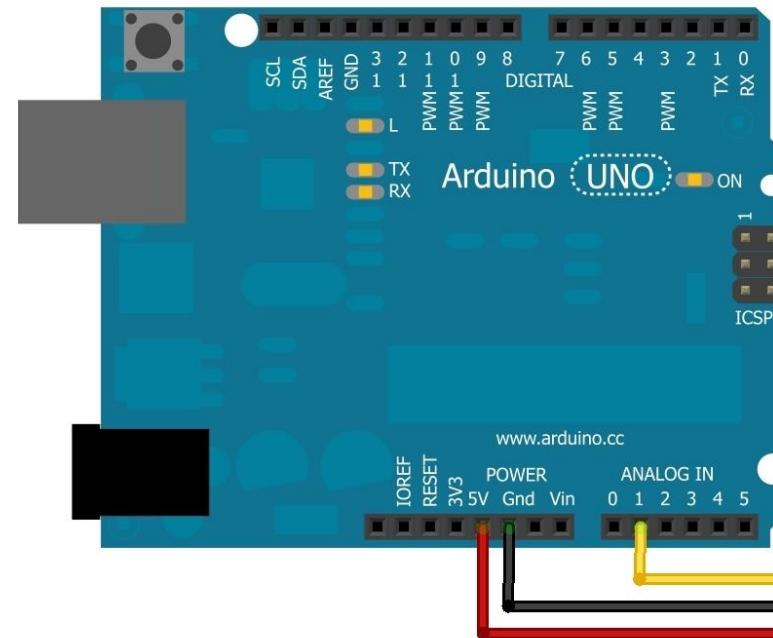


Mãos na massa! Será que chove?!

Este projeto irá te ajudar a monitorar com apenas 1 sensor a temperatura e umidade de seu clima local

► Material necessário:

- 1 Arduino;
- 1 resistor de 10k (marrom, preto, laranja);
- Sensor DHT11;
- 1 Protoboard;
- Jumpers cables.



Inserir um
resistor 10K
entre os dois
pinos

Mãos na massa! Será que chove?!

► Programa:

```
#include "DHT.h"
#define DHTPIN A1 // pino que estamos conectado
#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE); void loop()
{
    void setup() float h = dht.readHumidity();
    float t = dht.readTemperature();
    // testa se retorno é valido, caso contrário algo está errado.
    if (isnan(t) || isnan(h))
    {
        Serial.println("Failed to read from DHT");
    }
    else
    {
        Serial.print("Umidade: ");
        Serial.print(h);
        Serial.print("Temperatura: ");
        Serial.print(t);
    }
}
```

isnan (double __x): retorna 1 se
"não é um número"

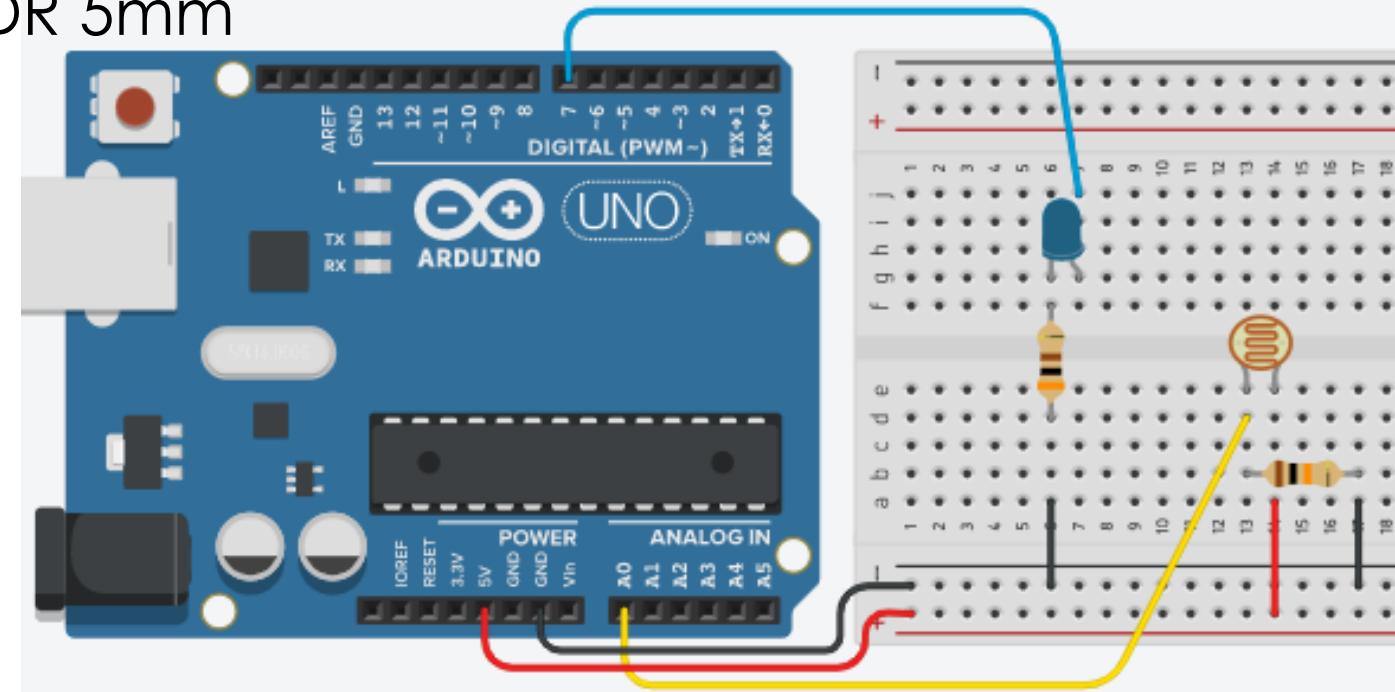


Mãos na massa! Sensor de luz

O LDR é capaz de identificar a luminosidade local

► Material necessário:

- 1 Arduino;
- 1 resistor de 10k ohms;
- 1 Sensor de Luminosidade LDR 5mm
- 1 resistor de 300 ohms;
- 1 LED;
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Sensor de luz

```
const int pinoLDR = A0; // pino LRD conectado
const int pinoLED = 7; // pino LED conectado
int leitura = 0; //variável p armazenar o valor lido

void setup() {
    pinMode(pinoLDR, INPUT); //config pino entrada
    pinMode(pinoLED, OUTPUT); //config pino saída
    Serial.begin (9600); // inicializa a serial
}

void loop() {

    leitura = analogRead(pinoLDR); // le o pino do LDR
    Serial.print ("Leitura:"); //mostrar no monitor
    Serial.println (leitura);

    if(leitura < 100){ // se a luminosidade for menor
        digitalWrite(pinoLED,HIGH); // acende o LED
    }
    else { // se não
        digitalWrite(pinoLED,LOW); // apaga o LED
    }
    delay(100); // aguarda 100 milissegundos p nova leitura
}
```

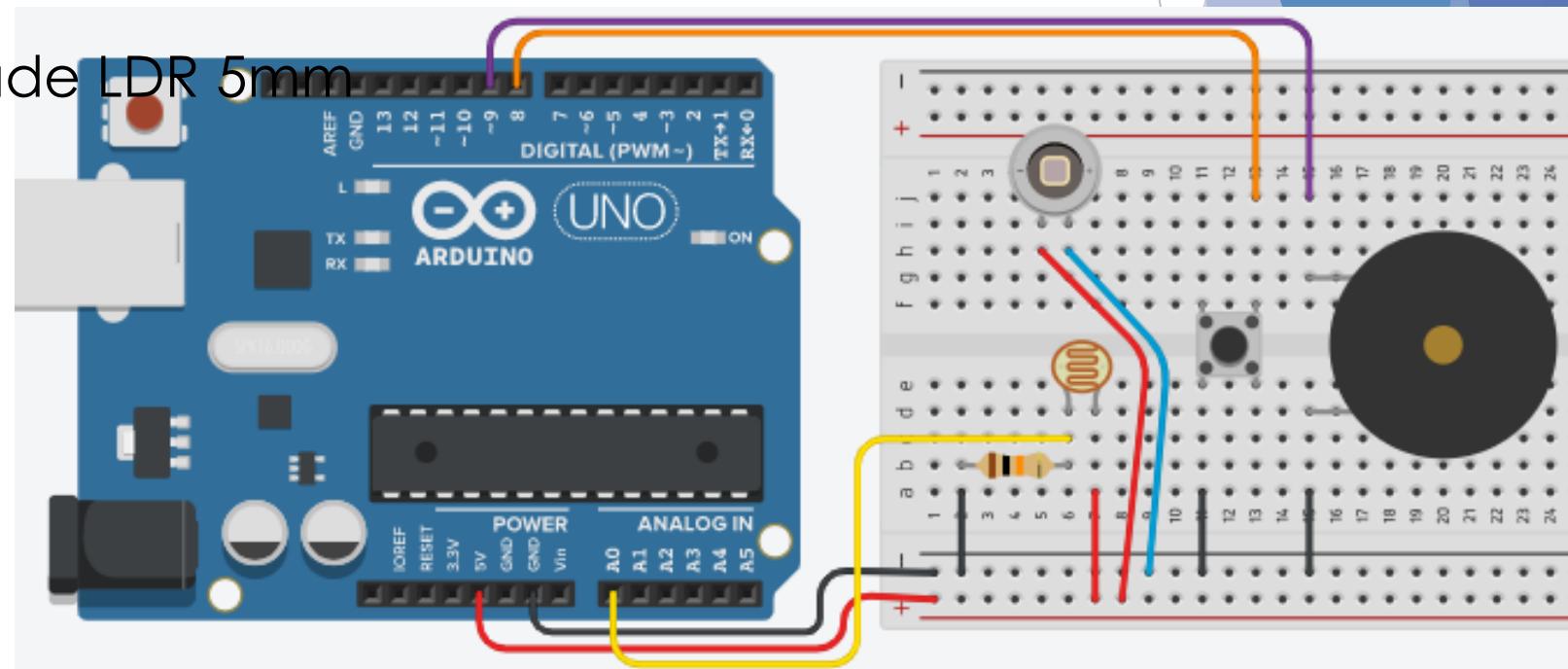


Mãos na massa! Xô, invasores!

Os sensores de presença identificam o movimento dentro de determinado raio de alcance, no nosso caso, quando algo ultrapassar o laser

► Material necessário:

- 1 Arduino;
- 1 resistor de 10k ohms;
- 1 Sensor de Luminosidade LDR 5mm;
- 1 Diodo Laser 5V;
- 1 Buzzer Passivo 5V;
- 1 Push button;
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Xô, invasores!

```
const int pinoLDR = A0;
const int pinoBuzzer = 9;
const int pinoBotao = 8;
int leituraLDR = 0;
int leituraBotao;

void setup() {
  pinMode(pinoLDR, INPUT); // configura como entrada
  pinMode(pinoBuzzer, OUTPUT);
  pinMode(pinoBotao, INPUT_PULLUP);
}

INPUT_PULLUP com pinMode():
monitora o estado de um switch
estabelecendo comunicação serial
entre seu Arduino e seu computador
via USB.

void loop() {
  // le o valor de tensão no pino do LDR
  leituraLDR = analogRead(pinoLDR);
  // le o estado do botão
  leituraBotao = digitalRead(pinoBotao);

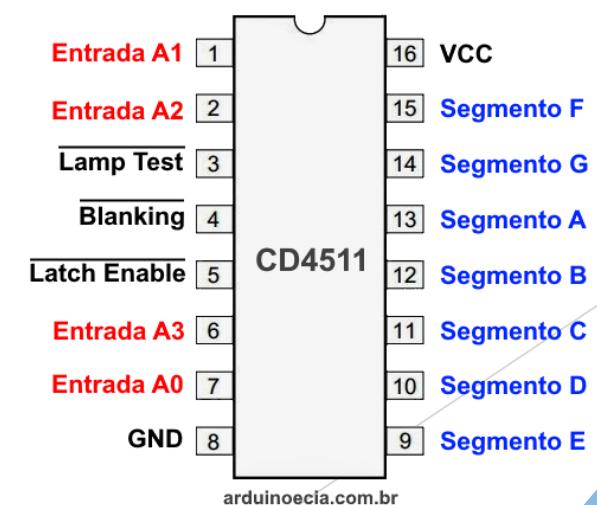
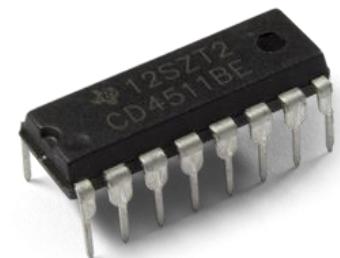
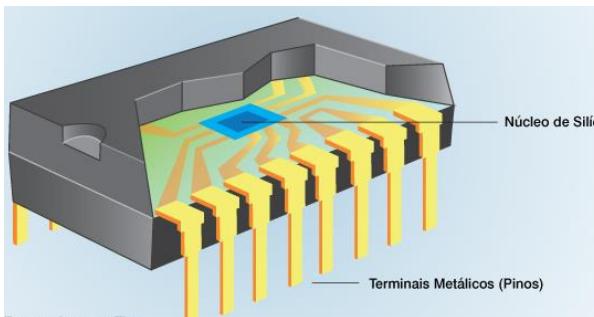
  if (leituraLDR <= 512) { // Se existir um obstáculo
    tone(pinoBuzzer, 1000); // aciona o buzzer
  }
  else if (leituraBotao == LOW) { // senão e botão estiver pressionado
    noTone(pinoBuzzer); // desliga o buzzer
  }
  delay(100); // aguarda 100 milissegundos para uma nova leitura
}
```



Círculo Integrado CD4511

Em eletrônica, um círculo integrado, microchip, chip, nanochip, abreviadamente **CI**, é um círculo eletrônico miniaturizado composto principalmente por dispositivos semicondutores sobre um substrato fino de material semicondutor.

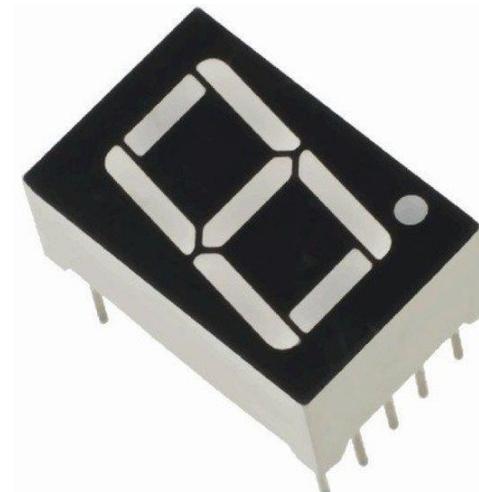
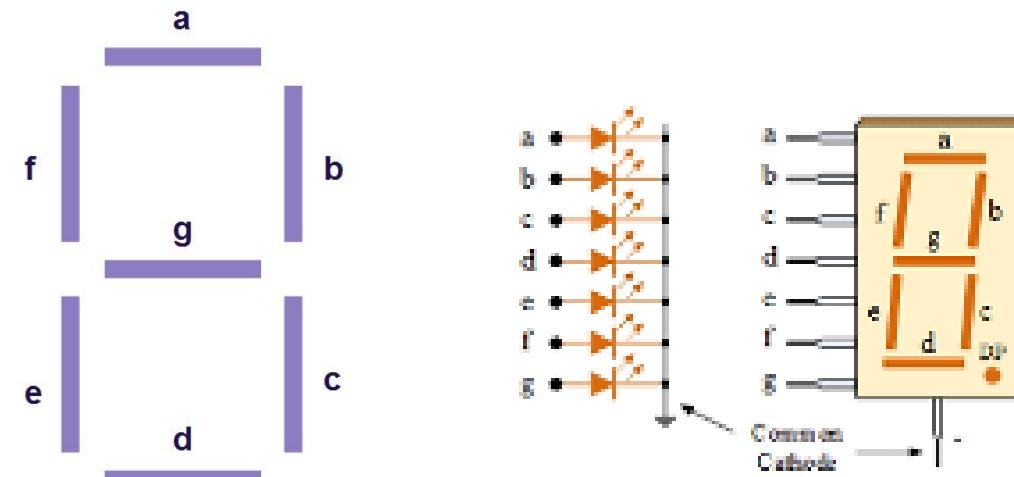
Este círculo integrado tem a função de converter BCD para display de 7 segmentos. Utilizando 4 pinos do CI, e através de uma simples contagem em binário, o CI consegue informar ao display de 7 segmentos quais segmentos deve acender para formar um determinado número



Display de 7 segmentos

Uma das formas mais simples para exibir informações para o usuário em um formato que seja inteligível – normalmente números ou letras é o uso de um componente denominado **Display de LEDs de 7 segmentos**

Com esse componente, é possível formar os caracteres de 0 a 9 e A a F

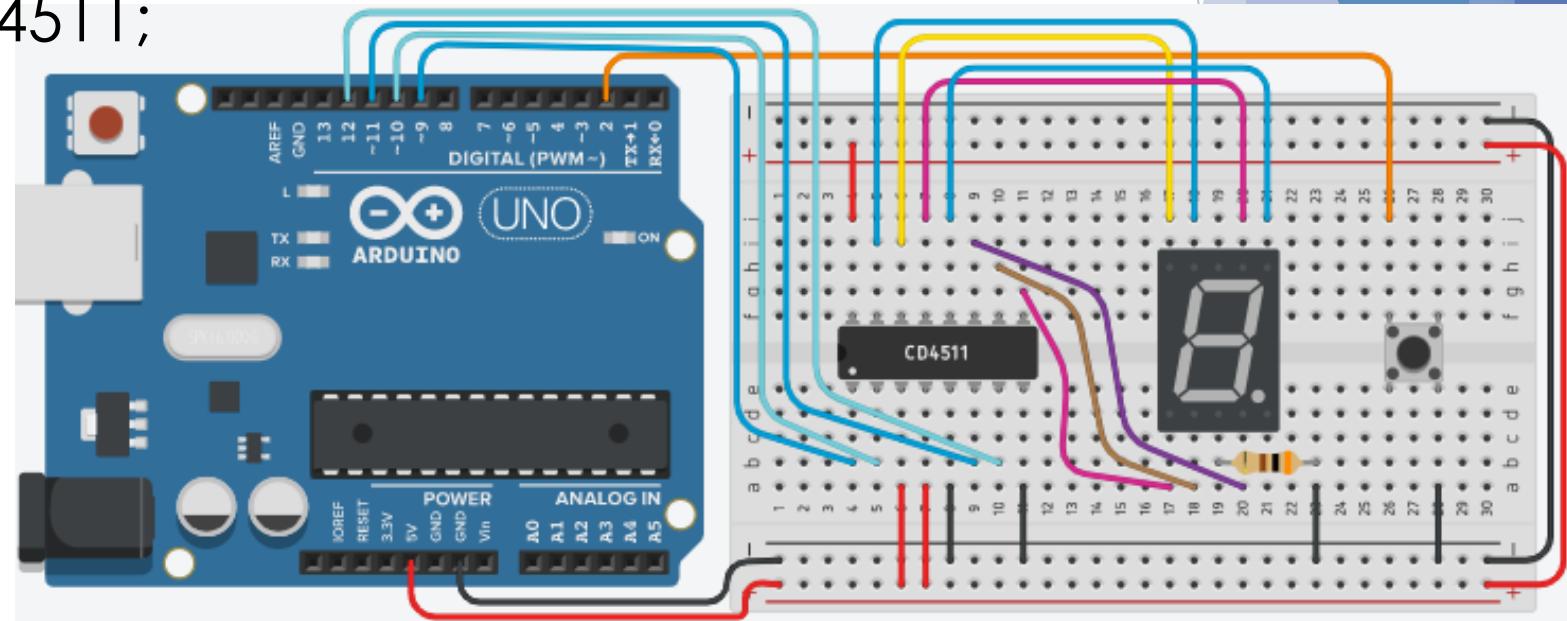


Mãos na massa! Dado eletrônico

Dados são utilizados quando precisamos sortear números de forma aleatória, então vamos tentar.

► Material necessário:

- 1 Arduino;
- 1 resistor de 300 ohms;
- 1 Display de 7 segmentos;
- 1 Circuito Integrado 4511;
- 1 Push button;
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Dado eletrônico

```
const int pino_a = 12;
const int pino_b = 9;
const int pino_c = 10;
const int pino_d = 11;

const int pino_botao = 2 ; //botão de sorteio
int leitura_botao, resultado; //leitura do botão

void setup() {
  Serial.begin(9600);
  //Configuração dos pinos do código BCD
  pinMode(pino_a, OUTPUT); //digito menos significativo
  pinMode(pino_b, OUTPUT);
  pinMode(pino_c, OUTPUT);
  pinMode(pino_d, OUTPUT); //digito mais significativo
  pinMode(pino_botao, INPUT_PULLUP); //botão de sorteio
}
```

```
void loop() {
  leitura_botao = digitalRead(pino_botao); //ler botão
  if (leitura_botao == LOW) { // se o botão foi pressionado
    delay(150); // delay de 150 ms para que o código não fique sorteando varios números
    resultado = random(1, 9); //sortear valores de 1 a 8
    switch (resultado) { // verifica em qual case o valor da variável resultado é igual
      case 1: //aciona os pinos para o dígito 1, quando o resultado for igual a 1
        digitalWrite(pino_a, HIGH);
        digitalWrite(pino_b, LOW);
        digitalWrite(pino_c, LOW);
        digitalWrite(pino_d, LOW);
        break; // encerra o switch

      case 2: //aciona os pinos para o dígito 2, quando o resultado for igual a 2
        digitalWrite(pino_a, LOW);
        digitalWrite(pino_b, HIGH);
        digitalWrite(pino_c, LOW);
        digitalWrite(pino_d, LOW);
        break; // encerra o switch
    }
  }
}
```



Mãos na massa! Dado eletrônico

```
case 3: //aciona os pinos para o digito 3, quando o resultado for igual a 3
  digitalWrite(pino_a, HIGH);
  digitalWrite(pino_b, HIGH);
  digitalWrite(pino_c, LOW);
  digitalWrite(pino_d, LOW);
break; // encerra o switch

case 4: //aciona os pinos para o digito 4, quando o resultado for igual a 4
  digitalWrite(pino_a, LOW);
  digitalWrite(pino_b, LOW);
  digitalWrite(pino_c, HIGH);
  digitalWrite(pino_d, LOW);
break; // encerra o switch

case 5: //aciona os pinos para o digito 5, quando o resultado for igual a 5
  digitalWrite(pino_a, HIGH);
  digitalWrite(pino_b, LOW);
  digitalWrite(pino_c, HIGH);
  digitalWrite(pino_d, LOW);
break; // encerra o switch

case 6: //aciona os pinos para o digito 6, quando o resultado for igual a 6
  digitalWrite(pino_a, LOW);
  digitalWrite(pino_b, HIGH);
  digitalWrite(pino_c, HIGH);
  digitalWrite(pino_d, LOW);
break; // encerra o switch

case 7: //aciona os pinos para o digito 7, quando o resultado for igual a 7
  digitalWrite(pino_a, HIGH);
  digitalWrite(pino_b, HIGH);
  digitalWrite(pino_c, HIGH);
  digitalWrite(pino_d, LOW);
break; // encerra o switch

case 8: //aciona os pinos para o digito 8, quando o resultado for igual a 8
  digitalWrite(pino_a, LOW);
  digitalWrite(pino_b, LOW);
  digitalWrite(pino_c, LOW);
  digitalWrite(pino_d, HIGH);
break; // encerra o switch
}
}
```

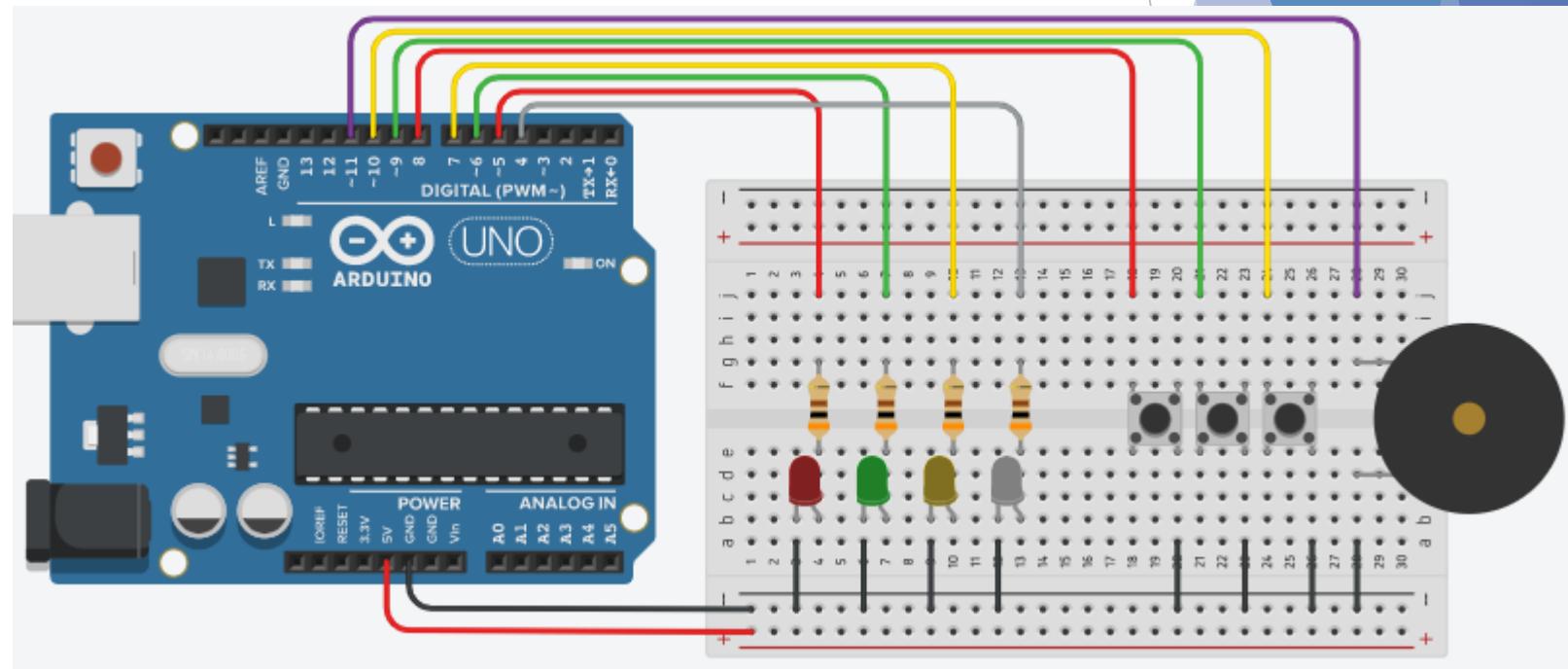
Mãos na massa! Lets play a game

Genius era um brinquedo muito popular na década de 1980 distribuído pela Brinquedos Estrela. O brinquedo buscava estimular a memorização de cores e sons.



► Material necessário:

- 1 Arduino;
- 4 resistores de 300 ohms;
- 4 LEDs;
- 1 buzzer passivo 5V
- 3 Push button;
- 1 Protoboard;
- Jumpers cables.



Mãos na massa! Lets play a game

```
const int c = 261; // Dó
const int d = 293; // Ré
const int e = 329; // Mi

const int ERRO = 4;//LED erro
const int ledR = 5;//LED1
const int ledG = 6;//LED2
const int ledB = 7;//LED3
const int b1 = 8;
const int b2 = 9;
const int b3 = 10;
const int pino_buzzer = 11;

int level = 0, i, compara = 0;
long notas[50] , resultado[50];

bool flag = LOW, flag_b1 = LOW, flag_b2 = LOW, flag_b3 = LOW, decrease = LOW;
```

```
void setup() {
    pinMode(b1, INPUT_PULLUP);
    pinMode(b2, INPUT_PULLUP);
    pinMode(b3, INPUT_PULLUP);
    pinMode(pino_buzzer, OUTPUT);
    pinMode(ledR, OUTPUT);
    pinMode(ledG, OUTPUT);
    pinMode(ledB, OUTPUT);
    pinMode(ERRO, OUTPUT);
    Serial.begin(9600);
    randomSeed(analogRead(0));
}

}
```



Mãos na massa! Lets play a game

```
void loop() {
    notas[level] = random(1, 4);
    delay(10);
    for ( i = 0; i < level; i++) {
        if (notas[i] == 1) {
            tone(pino_buzzer, c);
            digitalWrite(ledR, HIGH);
            delay(500);
            noTone(pino_buzzer);
            digitalWrite(ledR, LOW);
            delay(100);
        }
        else if (notas[i] == 2) {
            tone(pino_buzzer, d);
            digitalWrite(ledG, HIGH);
            delay(500);
            noTone(pino_buzzer);
            digitalWrite(ledG, LOW);
            delay(100);
        }
        else {
            tone(pino_buzzer, e);
            digitalWrite(ledB, HIGH);
            delay(500);
            noTone(pino_buzzer);
            digitalWrite(ledB, LOW);
        }
        flag = HIGH;
        i = 0;
        compara = 0;

        while (flag == HIGH) {
            bool leitura_b1 = digitalRead(b1);
            bool leitura_b2 = digitalRead(b2);
            bool leitura_b3 = digitalRead(b3);
            if ( i < level) {
                if (leitura_b1 == LOW) {
                    flag_b1 = HIGH;
                }
            }
        }
    }
}
```



Mãos na massa! Lets play a game

```
else if (leitura_b1 == HIGH && flag_b1 == HIGH)
{
    resultado[i] = 1;
    i++;
    flag_b1 = LOW;
}
if (leitura_b2 == LOW) {
    flag_b2 = HIGH;
} else if (leitura_b2 == HIGH && flag_b2 == HIGH) {

    resultado[i] = 2;
    i++;
    flag_b2 = LOW;
}
if (leitura_b3 == LOW) {
    flag_b3 = HIGH;
} else if (leitura_b3 == HIGH && flag_b3 == HIGH) {

    resultado[i] = 3;
    i++;
    flag_b3 = LOW;
}
delay(100);
}

else {
    for (int i = 0; i < level; i++) {
        resultado[i] = (resultado[i] - notas[i]);
        if (resultado[i] < 0) {
            resultado[i] = resultado[i] * -1;
        }
        compara = compara + resultado[i];
        Serial.println(compara);
        if (compara > 0) {
            decrease = HIGH;
        }
        else {
            decrease = LOW;
        }
    }
}
```



Mãos na massa! Lets play a game

```
if (decrease == HIGH) {  
    level = 0;  
    flag = LOW;  
    decrease = LOW;  
    Serial.print("Errado!");  
    for (i = 0; i < 5; i++) {  
        digitalWrite(ERRO, HIGH);  
        delay(50);  
        digitalWrite(ERRO, LOW);  
        delay(50);  
    }  
}  
else {  
    level++;  
    flag = LOW;  
    Serial.print("Nivel: ");  
    Serial.println(level);  
}  
delay(500);  
}  
}  
}
```

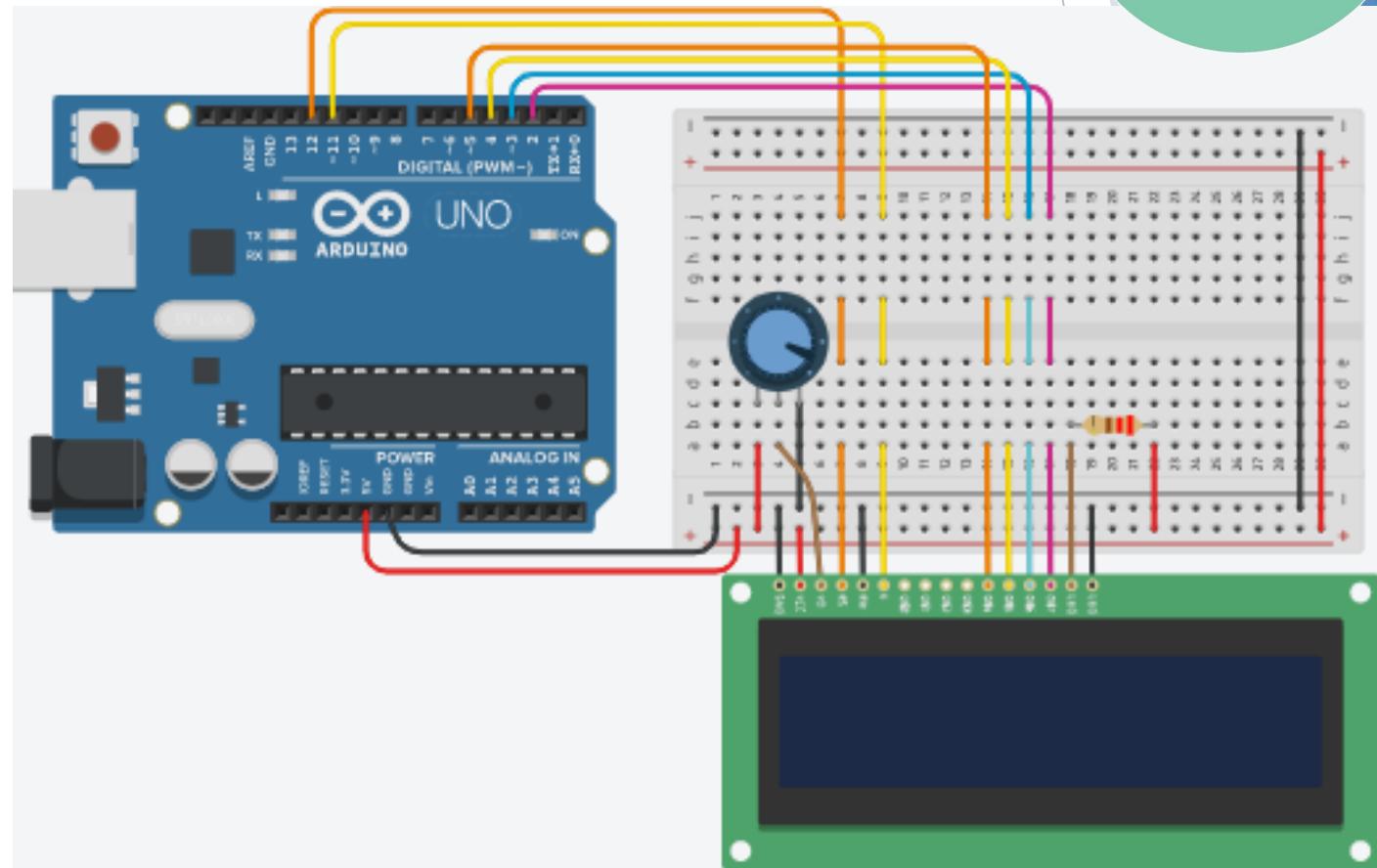


Mãos na massa! Olá, mundo!

Vamos iniciar a aprendizagem com o lcd

► Material necessário:

- 1 Arduino;
- 1 resistor de 220r;
- 1 potenciômetro 10K;
- 1 Protoboard;
- 1 LCD;
- Jumpers cables.

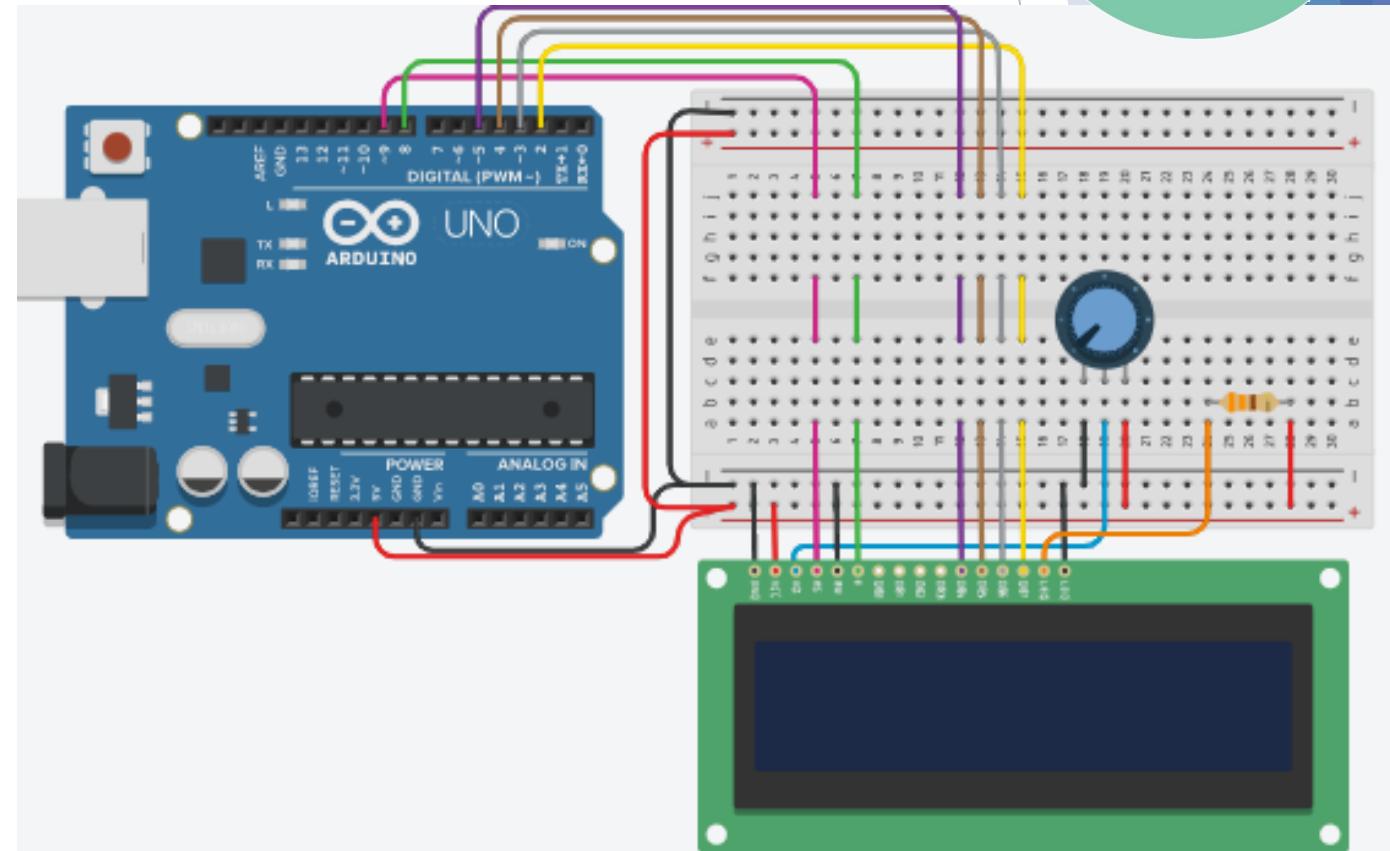


Mãos na massa! Tô programando!

Vamos iniciar a aprendizagem com o lcd

► Material necessário:

- 1 Arduino;
- 1 resistor de 220r;
- 1 potenciômetro 10K;
- 1 Protoboard;
- 1 LCD;
- Jumpers cables.



Mãos na massa! Olá, mundo!

► programa

```
#include <LiquidCrystal.h> // incluir a biblioteca

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // indicar os pinos

void setup() {
    lcd.begin(16, 2); // numero de colunas e linhas disponiveis
    lcd.print("Olá, mundo!"); // mostra a mensagem
    // lcd.write ('A'); Exibe apenas uma letra
}

void loop() {
    lcd.setCursor(0, 1); // indica que o cursor vai estar na coluna 0 linha 1
    lcd.print(millis() / 1000); // mostrar o tempo
}
```



Mãos na massa! Tô programando!

```
#include <LiquidCrystal.h> // incluir a biblioteca
LiquidCrystal lcd(9, 8, 5, 4, 3, 2); // indicar os pinos

void setup() {
  lcd.begin(16, 2); // informa o tamanho do LCD
}

void loop() {
  lcd.clear(); // limpa o display do LCD.
  lcd.print("Curso de"); // imprime a string no display do LCD.
  delay(1000);

  for (int i = 0; i < 3; i++) {
    lcd.setCursor(6,1); // posiciona cursor na coluna 0 linha 6 do LCD
    lcd.print("^ARDUINO^"); // imprime a string no display do LCD.
    delay(2000);
    lcd.noDisplay(); // pisca o display
    delay(500);
    lcd.display();
    delay(500);
  }

  lcd.clear();
  lcd.print("SENAI");
  lcd.setCursor(3,1);
  lcd.print("Jairo Candido");
  delay(4000);

  lcd.clear();
  lcd.print("Tô programando!!!");
  delay(1000);
  lcd.setCursor(15,1); // Teste da Tabela de Caracteres
  lcd.write(byte(244));

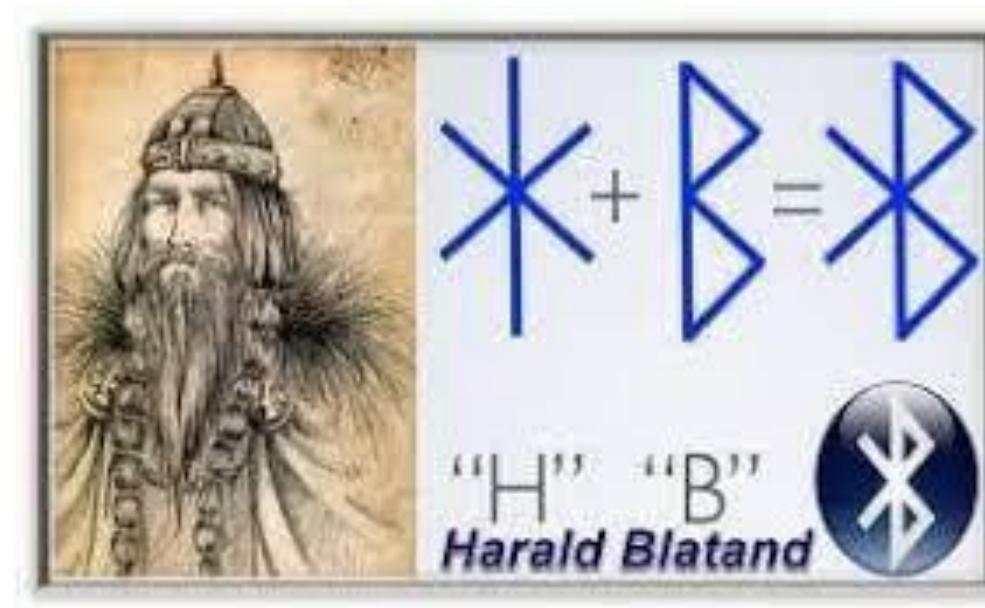
  for (int i = 0; i < 12; i++) { // Rolando para a esquerda 12 vezes
    lcd.scrollDisplayLeft();
    delay(600);
  }
  delay(1000);
}
```



Bluetooth

O nome é uma homenagem ao rei Haraldo Gormsson da Dinamarca, mais conhecido como Haraldo Dente-Azul.

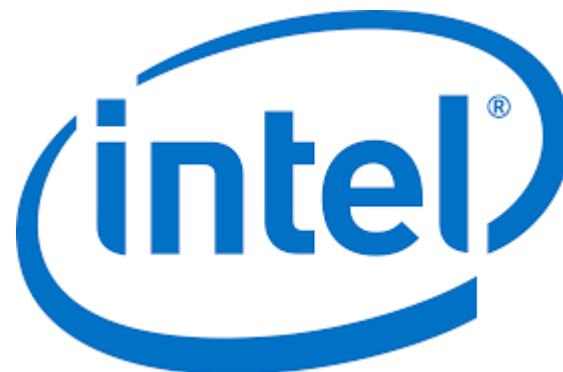
Esse monarca que viveu no fim do século 10 entrou para a história por ter unificado a Dinamarca e a Noruega



Bluetooth

Mais de mil anos depois, em 1996, representantes de três grandes nomes da indústria de tecnologia – Intel, Ericsson e Nokia – se reuniram para planejar a implementação de uma nova tecnologia, capaz de conectar aparelhos sem fio a curta distância.

O americano Jim Kardach, que havia ouvido a história do rei Haroldo de amigos suecos, sugeriu o nome Bluetooth como um apelido temporário para o projeto por unido os países nórdicos.



ERICSSON ≡ NOKIA

Módulo Bluetooth

► O **Módulo Bluetooth** possibilita transmitir e receber dados através de comunicação sem fio. Este módulo pode ser utilizado para criação de comunicação wireless para troca de informações entre dispositivos.

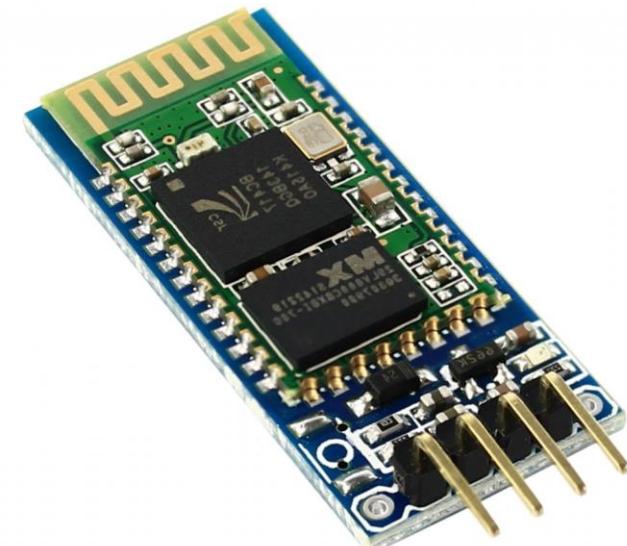
► **Especificações e características (HC-05):**

- Modelo: HC-05
- Tensão de operação: 3,6V – 6VDC
- Frequência de operação: 2,4GHz
- Nível de sinal lógico: 3,3V
- Protocolo bluetooth: v2.0+EDR
- Banda: ISM
- Modulação: GFSK
- Segurança: autenticação e criptografia
- Modo de funcionamento: master / slave
- Temperatura de operação: -40° ~ 105° celsius
- Alcance do sinal: ~10m
- Senha padrão (PIN): 1234

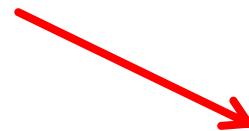


Aplicações módulos HC-05 / HC-06

- ▶ O módulo Bluetooth HC-05 pode trabalhar tanto em modo mestre (faz e aceita pareamento com outros dispositivos) como modo escravo (apenas aceita pareamento).
- ▶ Já o módulo Bluetooth HC-06 pode trabalhar apenas em modo escravo.



Baixar o aplicativo para Android.



The icon for the "Serial Bluetooth Terminal" app, which features a yellow and white design resembling a serial port or a small terminal screen with a blue Bluetooth symbol.

**Serial Bluetooth
Terminal**

Kai Morich

In-app purchases

Maquina de Estados

O Problema da Simultaneidade

- ▶ Em uma aplicação com microcontrolador é muito comum o firmware ter que tratar várias tarefas “ao mesmo tempo”



O Problema da Simultaneidade

- ▶ Em um sistema microcontrolado podem existir várias entradas que exijam tratamento.
- ▶ Estas entradas costumam ser assíncronas (ocorrem em momentos independentes).
- ▶ Alguns tratamentos podem ser demorados.
- ▶ Para resolver esses problemas temos algumas “**estratégias**”.

One Single Loop - Polling

- ▶ É a solução mais óbvia e comum.
- ▶ Consiste em um loop infinito dentro do qual todas as tarefas são executadas.
- ▶ Apenas as rotinas de inicialização acontecem antes dele
- ▶ Programação é mais simples
- ▶ Dificuldade de garantir requisições temporais, escalabilidade, reuso e manutenção prejudicados

```
while (1)
{
    if (TemTecla())
        TrataTecla();
    if (TemRx())
        TrataRx();
}
```

Cuidados a serem tomados

- ▶ Velocidade de execução do loop principal
 - ▶ Difícil cálculo da duração
 - ▶ Previsão de todas as opções
 - ▶ Leitura de valores/recepção de dados
- ▶ Travamento de uma função do sistema
- ▶ Atraso na resposta de eventos
- ▶ Pouco escalável

Quando utilizar One Single Loop?

- ▶ Sistemas simples
 - ▶ Pouca necessidade de temporizações
 - ▶ Prova de conceitos / Protótipo
- ▶ Sistemas de baixo custo
- ▶ Separar as funcionalidades em diferentes funções ou arquivos!

Interrupt Control System

- ▶ Visa resolver o problema das restrições temporais
- ▶ Apenas alguns subsistemas podem ser tratados via interrupções
- ▶ Os demais sistemas operam no loop principal
- ▶ Facilita a expansão do sistema, já que as interrupções não dependem do loop principal
- ▶ Ótima alternativa para sistemas que tem como requisito o baixo consumo de energia

Interrupt Control System

- ▶ O programa principal é interrompido e uma função pré definida é executada
 - ▶ O programa principal não percebe a pausa
 - ▶ Redução do tempo de resposta aos eventos
- ▶ O tempo para responder a um determinado evento é o menor possível utilizando um microcontrolador
 - ▶ As funções devem ser curtas.
- ▶ Deixar o processamento pesado para o loop principal

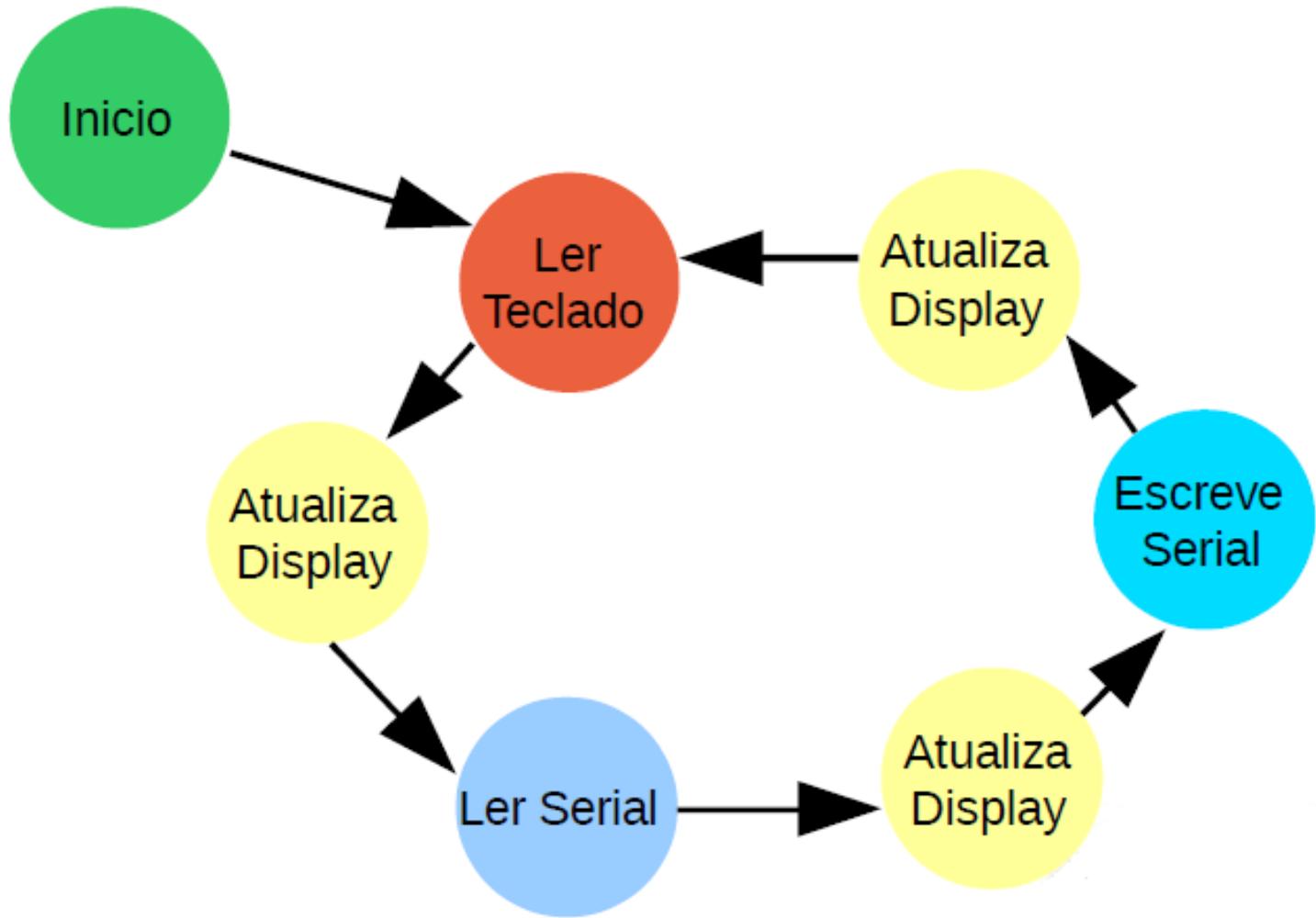
Máquina de Estados

- ▶ Uma máquina de estados se fundamenta, em direcionar o funcionamento de um software em um número finito de estados, sendo cada um desses estados uma **situação relevante** do sistema.
- ▶ É possível avançar, recuar ou permanecer em um estado, e o mecanismo deste fluxo é definido pelo desenvolvedor.

Máquina de Estados

- ▶ Uma máquina de estados se fundamenta, em direcionar o funcionamento de um software em um número finito de estados, sendo cada um desses estados uma **situação relevante** do sistema.
- ▶ É possível avançar, recuar ou permanecer em um estado, e o mecanismo deste fluxo é definido pelo desenvolvedor.

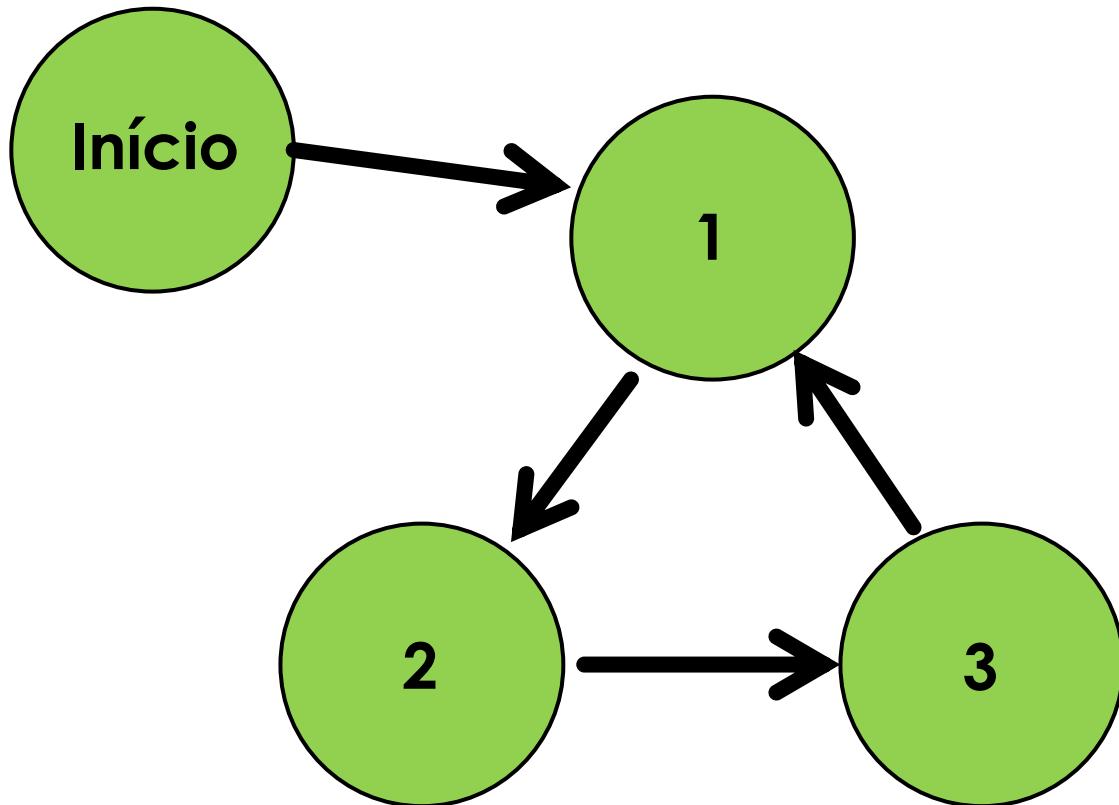
Máquina de Estados



Máquina de Estados

- ▶ Cada ciclo representa um estado do sistema, a tarefa atualmente em execução
- ▶ Após a fase de inicialização, o sistema entra num ciclo
- ▶ A transposição da máquina de estados para **linguagem C** pode ser feita de modo simples através da estrutura **Switch-Case**

Exemplo



Exercício

- ▶ Elaborar uma máquina de estados para o diagrama abaixo:

