

**INSTITUTO FEDERAL**

Santa Catarina

Câmpus Tubarão

# Introdução a Linguagens de Programação

Curso Superior de Tecnologia em Sistemas Embarcados

Professor: Fernando Silvano Gonçalves

[fernando.goncalves@ifsc.edu.br](mailto:fernando.goncalves@ifsc.edu.br)

Março de 2023

# Cronograma

Encontro	Data	Nº Aulas	Conteúdo
1	7-fev.	04	Recepção e Apresentação do Unidade / Apresentação do Plano de Ensino / Avaliação Diagnóstica / Introdução a sistemas embarcados / Conceitos, Características e Aplicações
2	14-fev.	04	Visita Tecnica Evolumenta Sistemas
3	28-fev.	04	Histórico de Sistemas Embarcados / Conceitos de Projeto de Sistemas Embarcados
4	9-mar.	04	Conceitos de Projeto de Sistemas Embarcados / Projeto de Sistemas Embarcados
5	14-mar.	04	Microcontroladores e Microprocessadores / Introdução ao Arduino
6	21-mar.	04	Introdução à Linguagens de Programação
7	28-mar.	04	Variáveis e Operadores
8	4-abr.	04	Estruturas Condicionais
9	11-abr.	04	Estruturas de Repetição
10	18-abr.	04	Avaliação 01



# Cronograma

Encontro	Data	Nº Aulas	Conteúdo
11	25-abr.	04	Microcontroladores
12	2-mai.	04	Entradas e Saídas Digitais
13	9-mai.	04	Conversor Analógico-Digital
14	16-mai.	04	Sensores
15	23-mai.	04	Comunicação Serial
16	30-mai.	04	PWM
17	6-jun.	04	Temporizadores
18	13-jun.	04	Interrupções
19	20-jun.	04	Avaliação 02
20	27-jun.	04	Conselho de Classe / Atividades de Encerramento da UC
		80	



# Pauta

- Microcontroladores e Microprocessadores;
- Periféricos;
- Arduino;
- Introdução ao C;
- Variáveis;
- Operadores;

# Microcontroladores e Microprocessadores

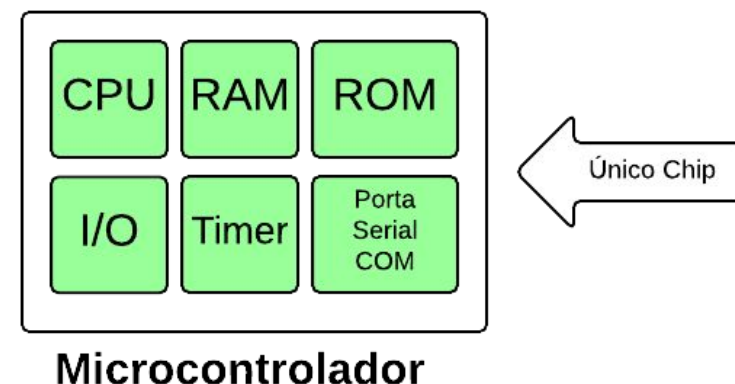
# Microcontroladores e Microprocessadores

O que são microcontroladores e microprocessadores?

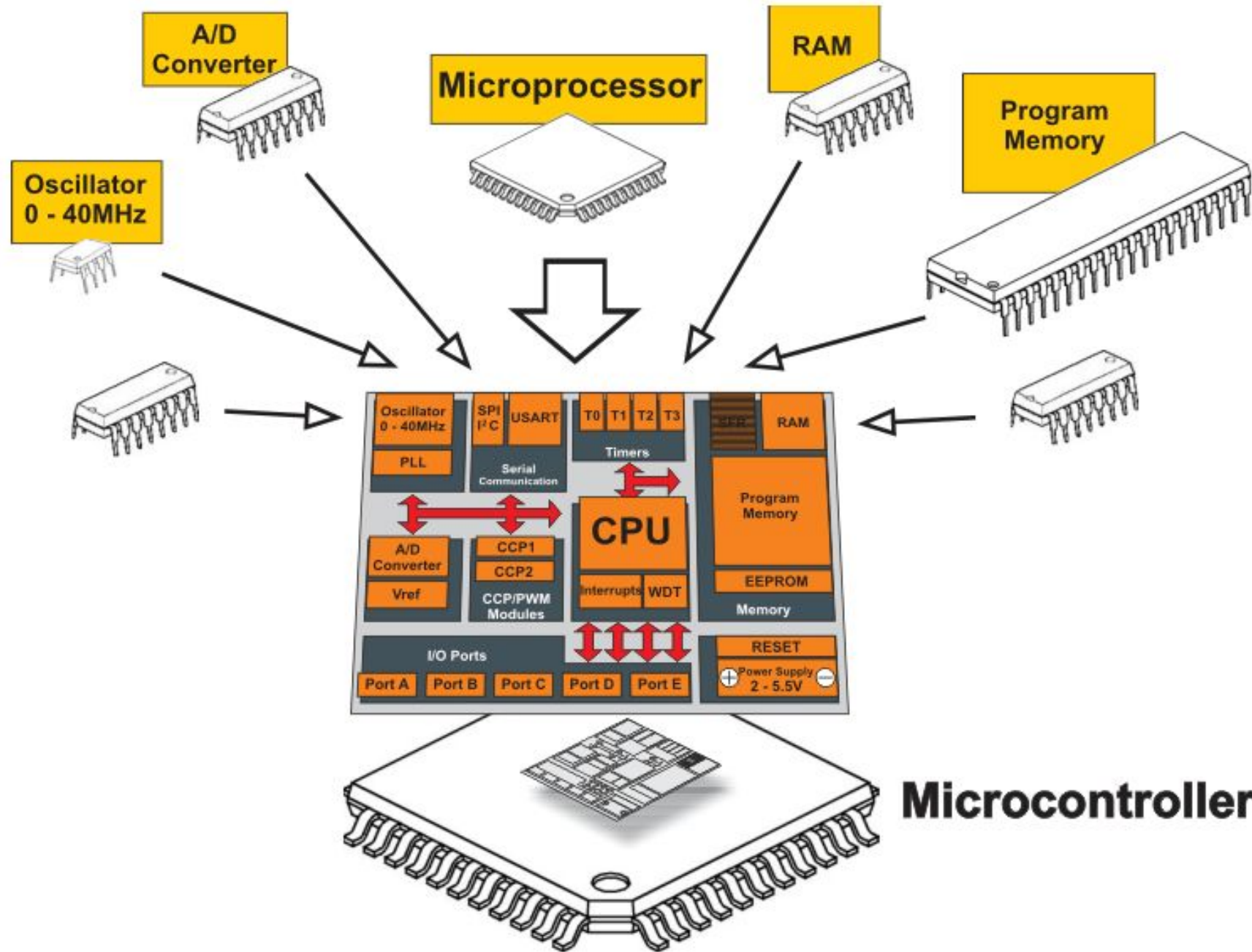


# Microcontroladores e Microprocessadores

- ❑ Computador em um único chip;
- ❑ Custo reduzido;
- ❑ Recursos integrados:
  - ❑ RAM;
  - ❑ ROM;
  - ❑ Conversor A/D e D/A;
  - ❑ Portas I/O;
- ❑ Timers e Interrupções;
- ❑ Podem ser programados com código Assembly ou em linguagens de alto nível;

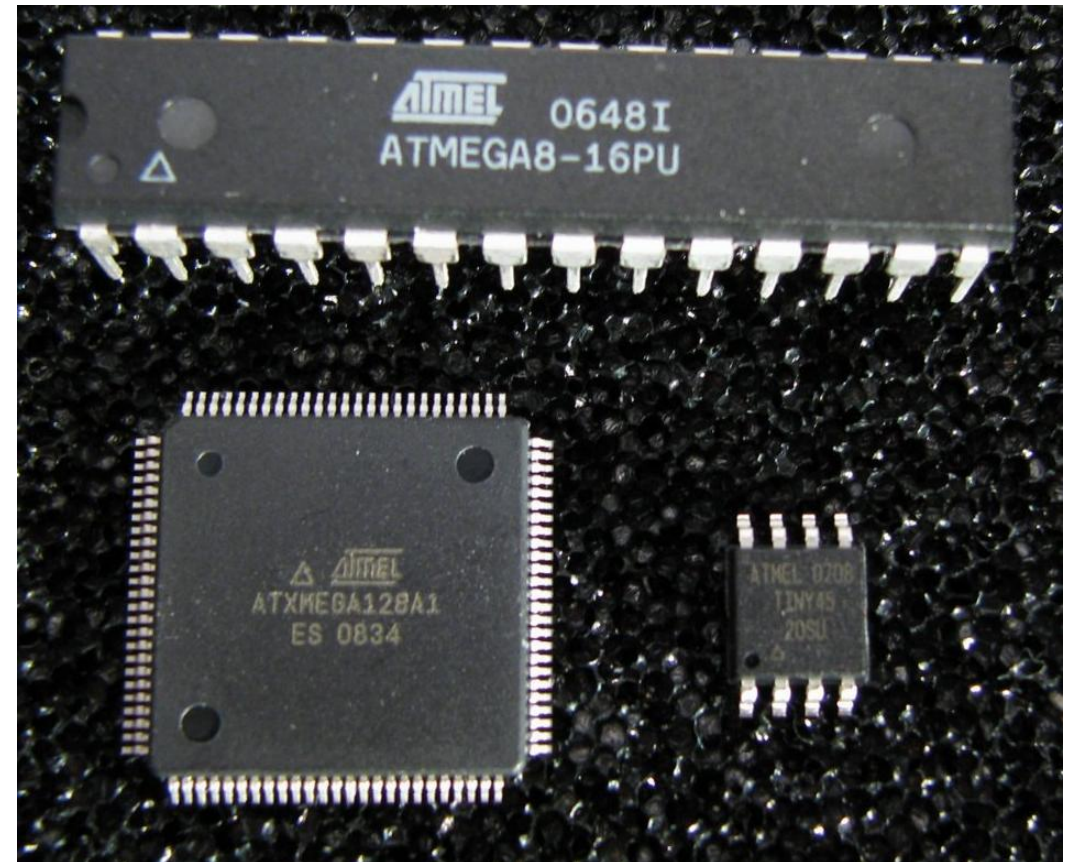
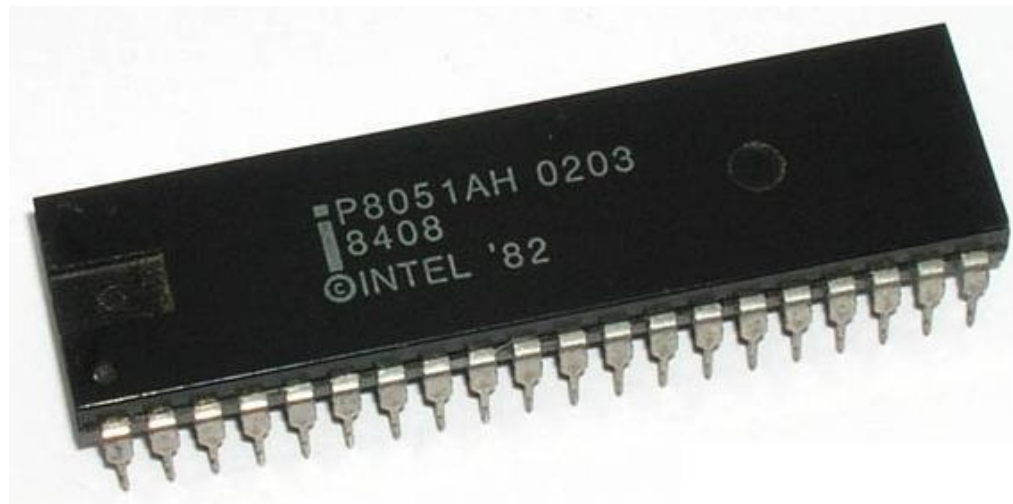
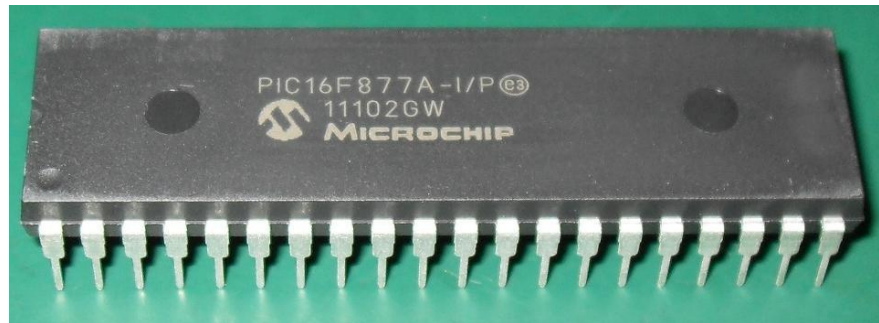
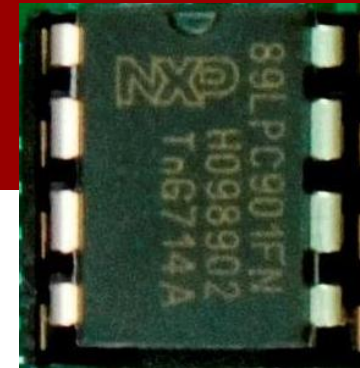


# Microc





# Microcontroladores



# Microprocessadores

- ❑ Circuito integrado que realiza as funções de cálculo e tomada de decisão de um computador;
- ❑ Controlado por um programa armazenado na memória, executando diferentes operações;
- ❑ Memórias RAM e ROM portas de I/O e Timers externos;
- ❑ Computador de propósito geral;
- ❑ Comunicação por barramentos.

# Microprocessadores - Características

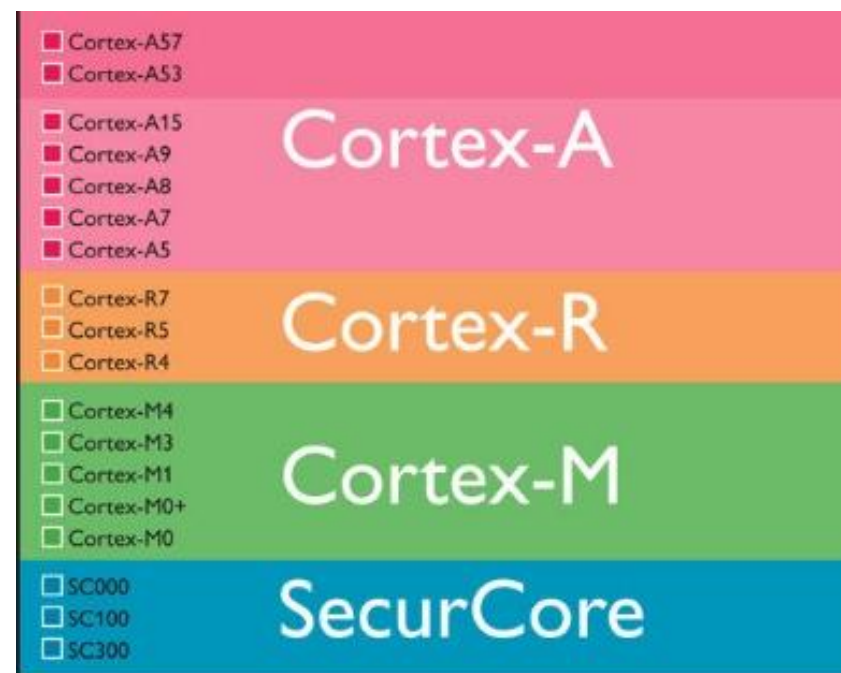
- ❑ **Memória de programa:** Armazena as instruções que um microprocessador deve executar (ROM);
- ❑ **Memória de dados:** Área de leitura e escrita de dados temporários sempre que necessário (RAM);
- ❑ **ULA:** Unidade Lógica Aritmética, responsável pelos cálculos e a lógica matemática para tomada de decisão;

# Microprocessadores - Características

- ❑ **CPU:** Unidade Central de Processamento, responsável pelo processamento de dados da unidade, ela quem interpreta os comandos e ativa os dispositivos de entrada e saída;
- ❑ **Periféricos:** Circuitos que realizam funções específicas auxiliando a CPU no controle e interface com dispositivos externos.








# Microprocessadores








# Microcontroladores Vs Microprocessadores

## **Microcontrolador:**

-  CPU, RAM, ROM, I/O e Timer encapsulados em um único chip;
-  Tamanho fixo no chip de RAM, ROM e portas I/O;
-  Aplicada a soluções onde custo, consumo de energia e espaço são críticas;
-  Propósito específico;
-  Ex: 8051, PIC, Atmel, Motorola, etc.

## **Microprocessador:**

-  CPU independente, RAM, ROM, I/O e Timer separados;
-  Desenvolvedor pode definir a quantidade de RAM, ROM e portas I/O;
-  Custo mais elevado;
-  Propósito geral;
-  Ex: 8085, 8086, Cortex-M4 Cortex-A5, etc.

# Periféricos

# Periféricos - Características

- ❑ Responsáveis pelo sensoriamento e atuação das aplicações;
- ❑ Diferentes características devem ser consideradas na sua especificação (consumo de energia, protocolo de comunicação, taxa de amostragem);
- ❑ Grande variedade existente hoje no mercado;
- ❑ Especificação diretamente dependente do escopo e orçamento do projeto;
- ❑ Necessário um profundo conhecimento do escopo da aplicação para sua definição.



# Periféricos - Exemplos de Uso

- ❑ Avaliar se há pessoas em um ambiente;
- ❑ Monitorar nível de luminosidade;
- ❑ Monitorar nível de umidade do solo;
- ❑ Controlar quantidade de água dispersa em determinado local;
- ❑ Medir distância de um objeto;
- ❑ Seguir determinada trajetória;

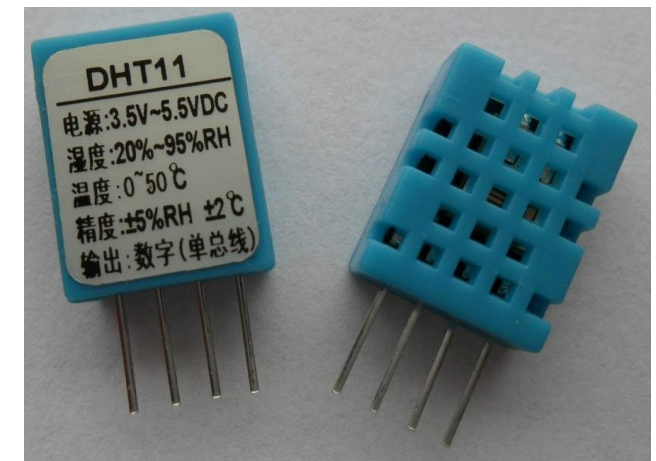
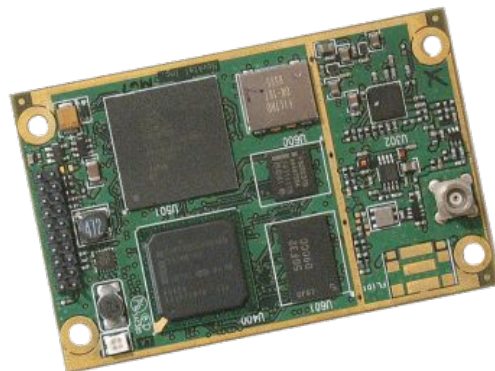
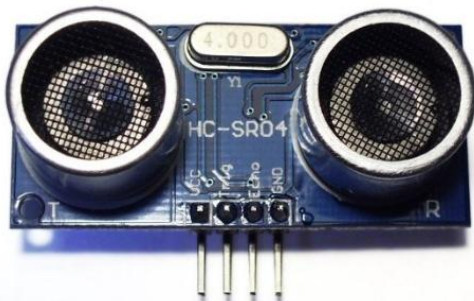
# Periféricos - Cuidados ao Especificar

- ☐ Qual a taxa de amostragem que eu preciso?
- ☐ Qual o protocolo de comunicação? Ele é compatível com a placa que estou utilizando?
- ☐ O custo deste sensor está dentro do orçamento do projeto?
- ☐ Qual o seu consumo de energia?

# Periféricos - Cuidados ao Manusear

- ❑ Verificar se as ligações estão corretas antes de ligar;
- ❑ Verificar se os polos de alimentação estão corretamente ligados;
- ❑ Verificar se a tensão de alimentação e de comunicação estão corretas;
- ❑ Verificar compatibilidade com a plataforma embarcada;
- ❑ Verificar necessidade de algum componente adicional no circuito de ligação (resistor, capacitor).

# Sensores

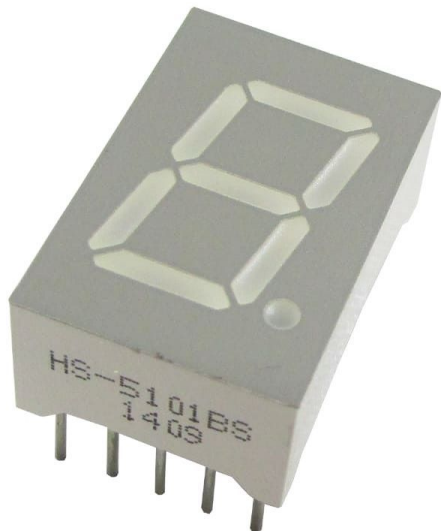


# Sensores





# Displays

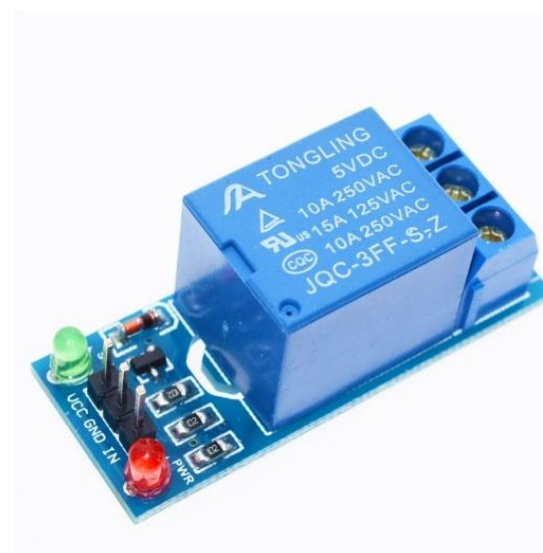
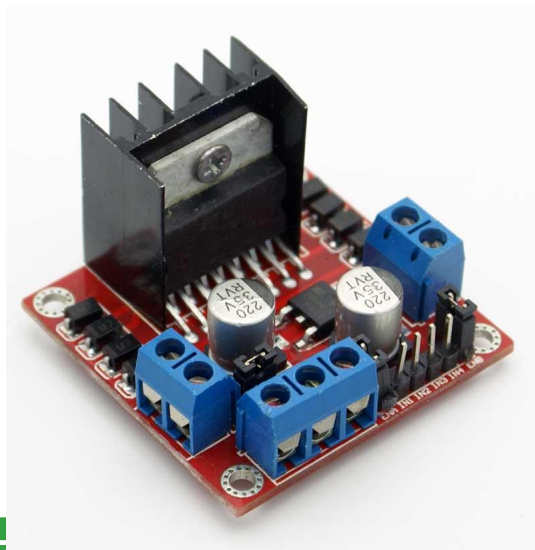


7 inch Display with SSD1963 Controller Board

Arduino Shield

Arduino MEGA2560

# Atuadores



# Onde comprar material?

## ☐ No Brasil

- ☐ Filipe Flop ([www.filipeflop.com](http://www.filipeflop.com));
- ☐ Proesi ([www.proesi.com.br](http://www.proesi.com.br));
- ☐ Robocore ([www.robocore.net](http://www.robocore.net));

## ☐ No exterior

- ☐ Ali Express ([pt.aliexpress.com/](http://pt.aliexpress.com/));
- ☐ Deal Extreme ([www.dx.com/pt/](http://www.dx.com/pt/));
- ☐ Ebay ([www.ebay.com](http://www.ebay.com));

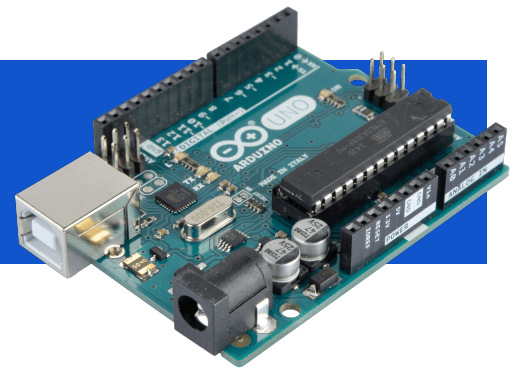


# Arduino

# Arduino

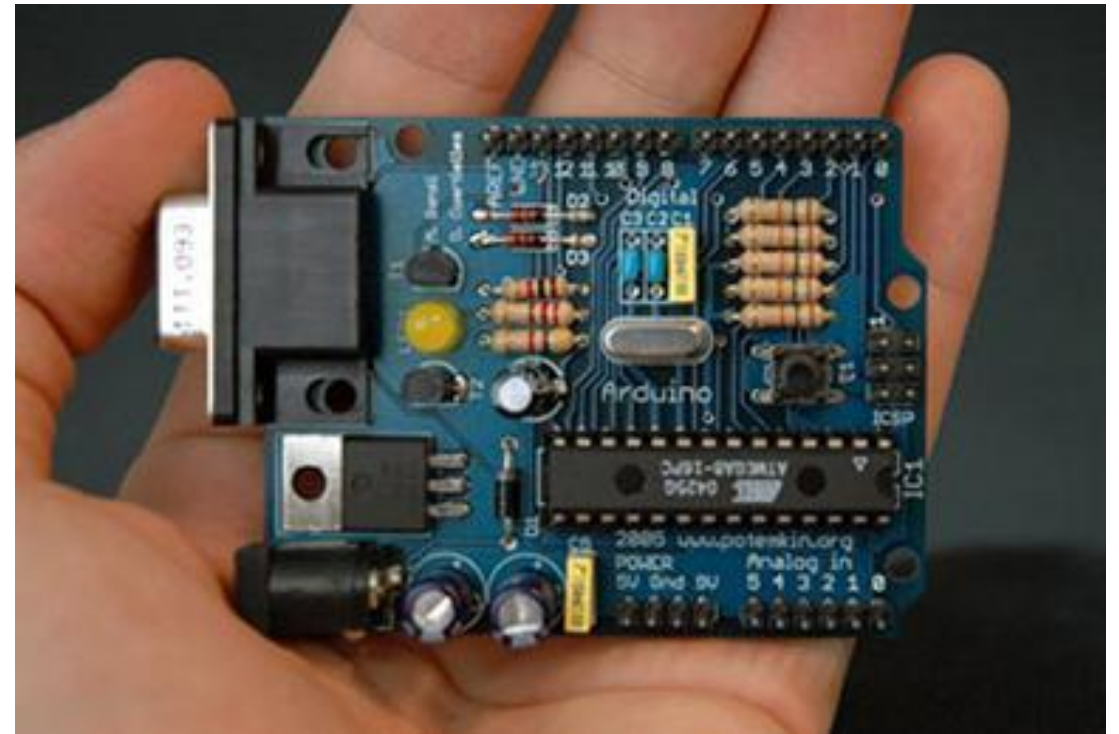
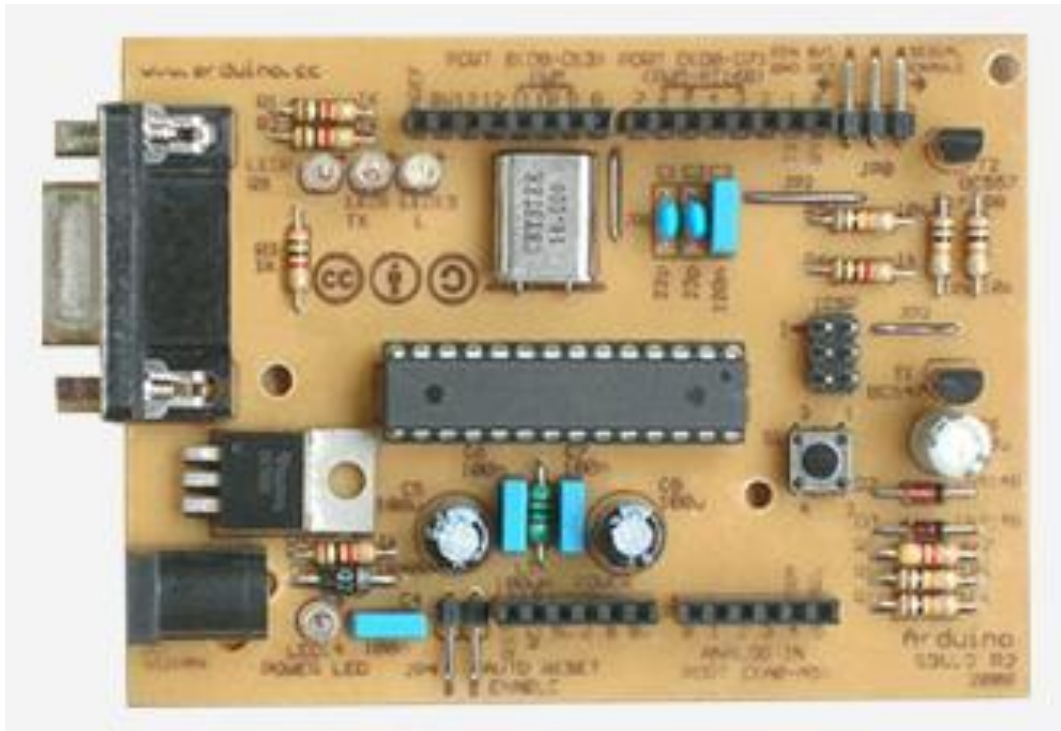


# Arduino



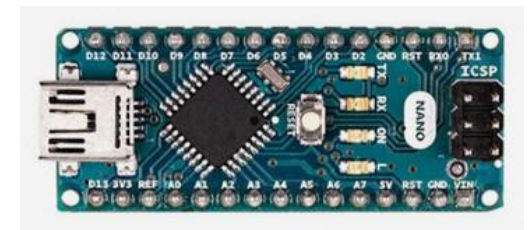
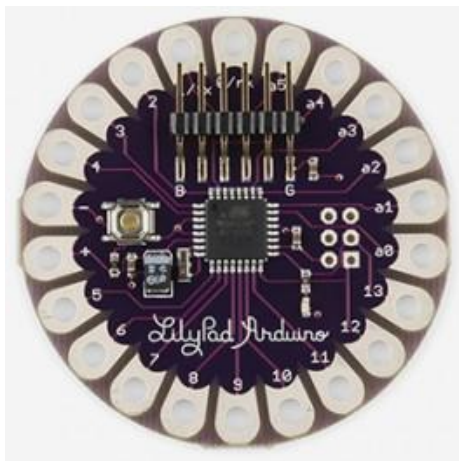
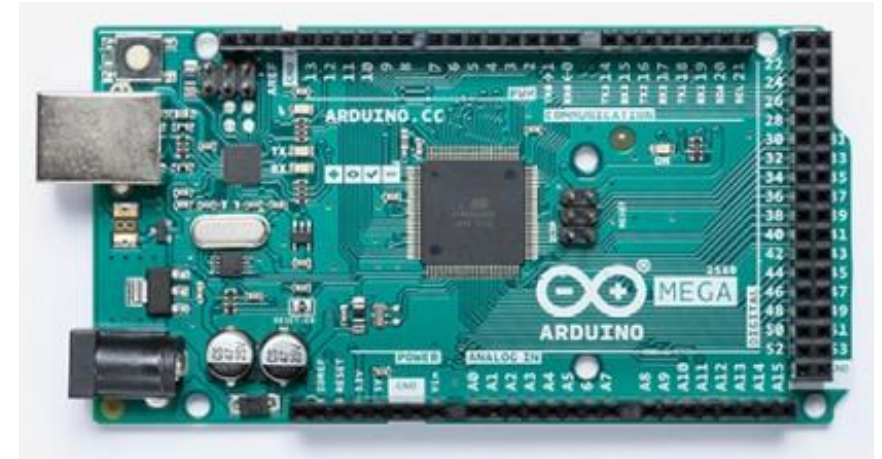
- ❑ Plataforma Open-Source de protótipos eletrônicos baseados em hardware e software flexível e fácil de usar;
- ❑ Destinado a qualquer pessoa interessada em criar objetos ou ambientes interativos;
- ❑ Projeto teve início em 2005 na cidade de Ivrea, Itália;
- ❑ Placa baseada em microcontrolador Atmel AVR e um ambiente de desenvolvimento baseado em C++;

# Arduino - Primeiras versões





# Arduino - Modelos existentes



# Características do Arduino UNO

<b>Microcontrolador:</b>	ATmega328
<b>Tensão de Operação:</b>	5V
<b>Tensão de Entrada:</b>	7 – 12v
<b>Pinos Digitais:</b>	14 (com 6 pinos com saída PWM)
<b>Pinos Analógicos:</b>	6
<b>Corrente por pino I/O:</b>	40mA
<b>Memória Flash:</b>	32KB
<b>SRAM:</b>	2KB
<b>Clock Speed:</b>	16MHz

# Portas

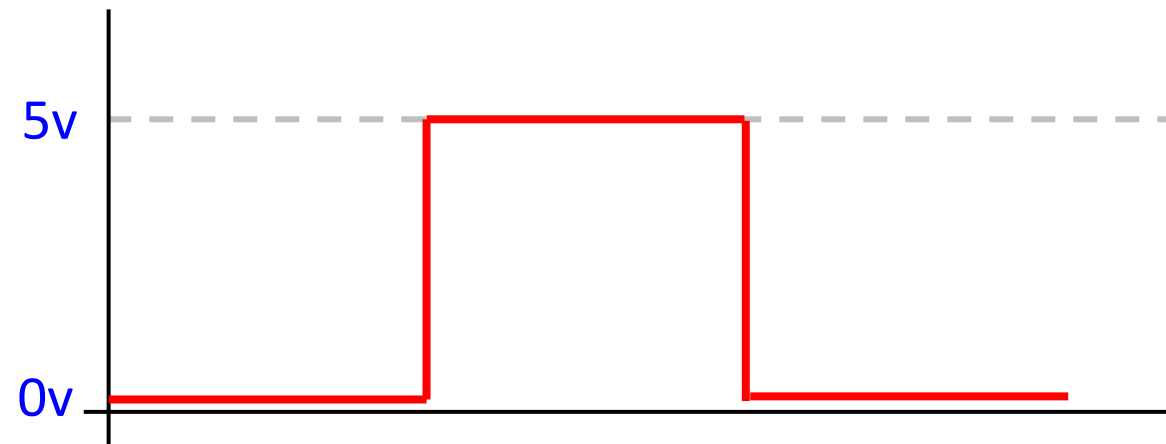
# Portas

# Tipos de Portas

- ❑ Portas Digitais;
- ❑ Portas Analógicas;
- ❑ Portas PWM;

# Portas Digitais

- ❑ 14 Portas Digitais;
- ❑ 0 / 5v
  - ❑ LOW e HIGH;

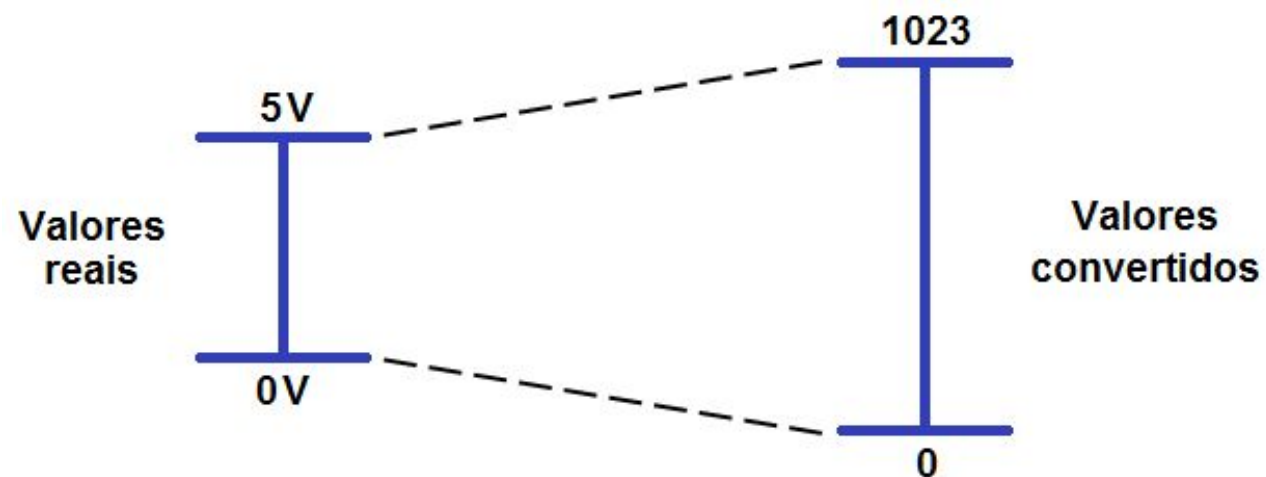
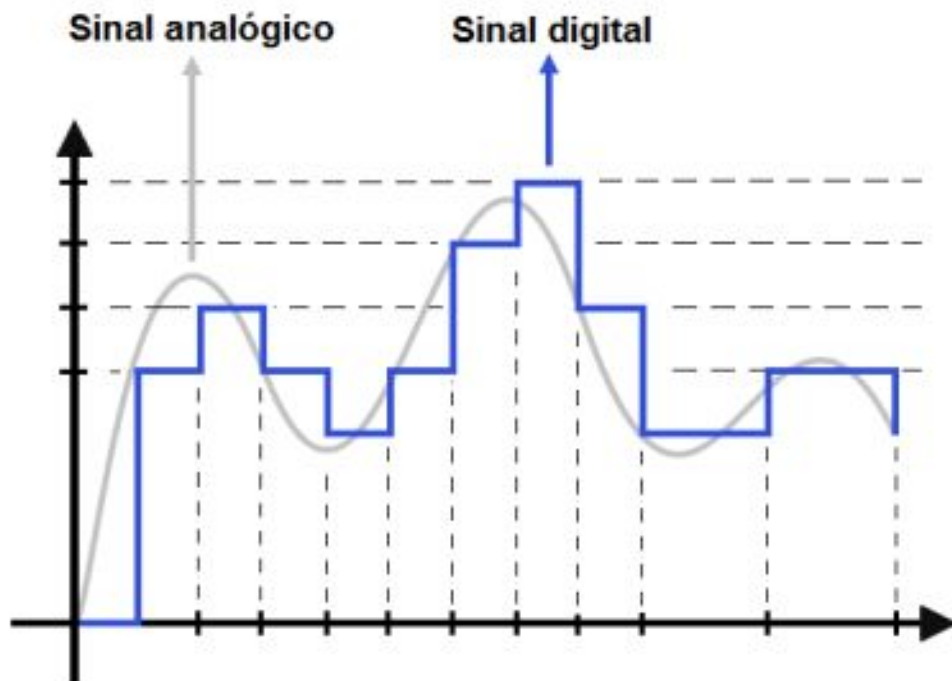




# Portas Analógicas

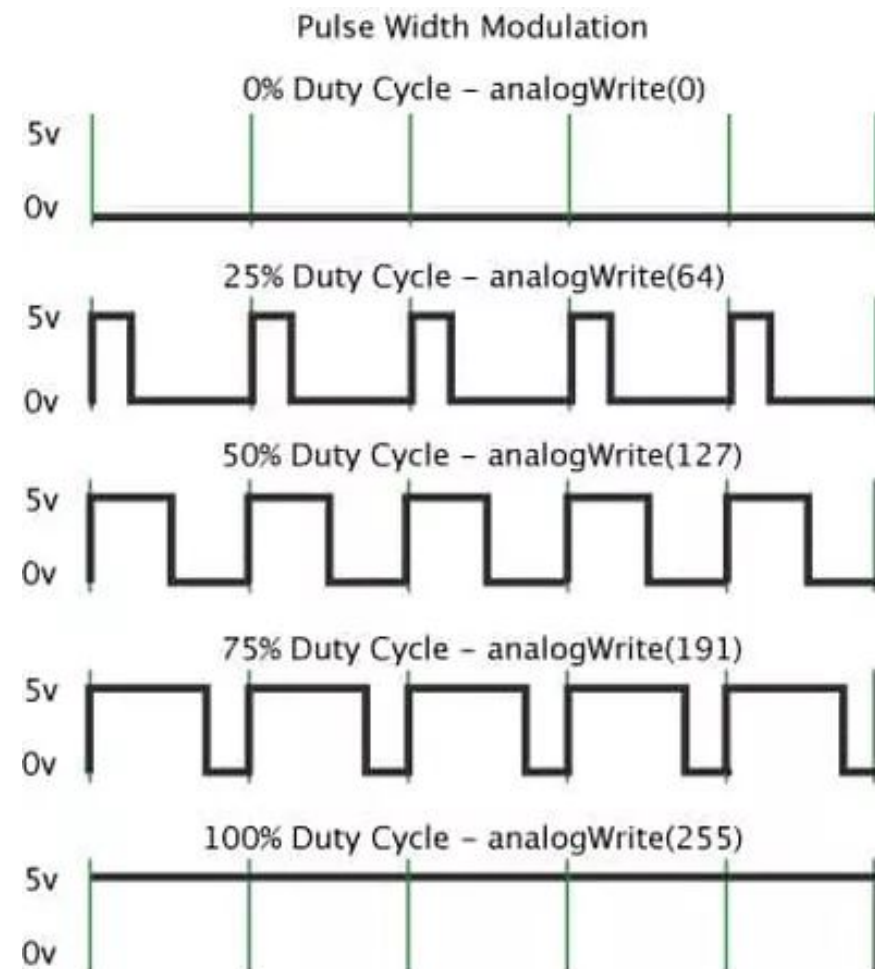
## 6 Portas Analógicas

0 ~ 5V.



# Portas PWM (~)

- ❑ Onda quadrada onde se controla a porcentagem do tempo em que ela permanece em nível lógico alto (Duty Cycle).
- ❑ Alteração no Duty Cycle provoca mudança no valor médio da onda, variando entre 0v e 5v.
- ❑ Valor do Duty Cycle armazenado em um registrador de 8 bits variando entre 0 e 255.



# Shields

# Shields

# Shields

- ❑ Shields são placas que podem ser conectados sobre o Arduino estendendo as suas funcionalidades;
- ❑ Shields seguem a mesma filosofia do Arduino:
  - ❑ Open;
  - ❑ Fácil de Montar;
  - ❑ “Barato”

# Sh



# Software de Desenvolvimento



# Software de Desenvolvimento

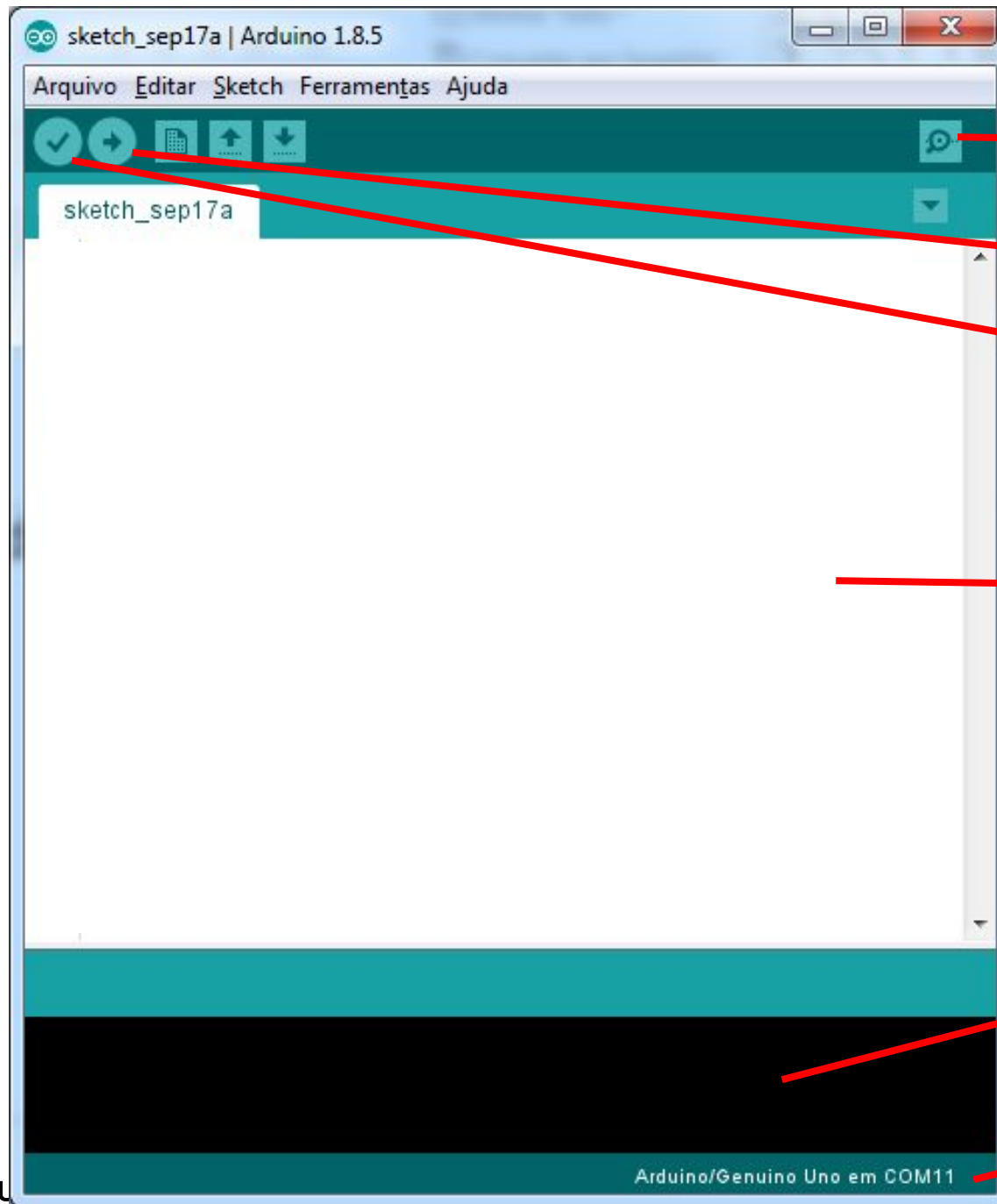
- ❑ Open-Source;
- ❑ Windows, Mac e Linux;
- ❑ Disponível em:  
<http://arduino.cc>



# Características da IDE

- ❑ Facilita o desenvolvimento de software;
- ❑ Muitos exemplos para as bibliotecas padrão;
- ❑ Permite a gravação do código no microcontrolador;
- ❑ Monitor serial para troca de mensagens;





Monitor Serial

Gravação do Código

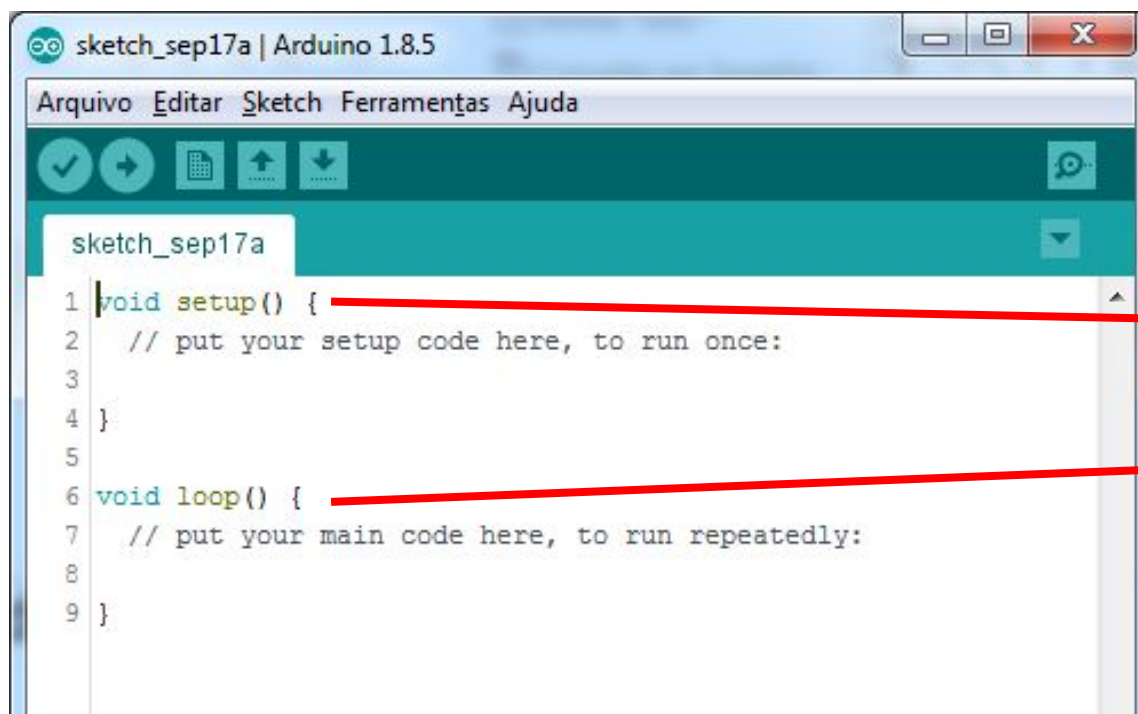
Compilar Código

Área de Programação

Informação

Comunicação

# Desenvolvimento para Arduino



```
sketch_sep17a | Arduino 1.8.5
Arquivo  Editar  Sketch  Ferramentas  Ajuda

sketch_sep17a
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

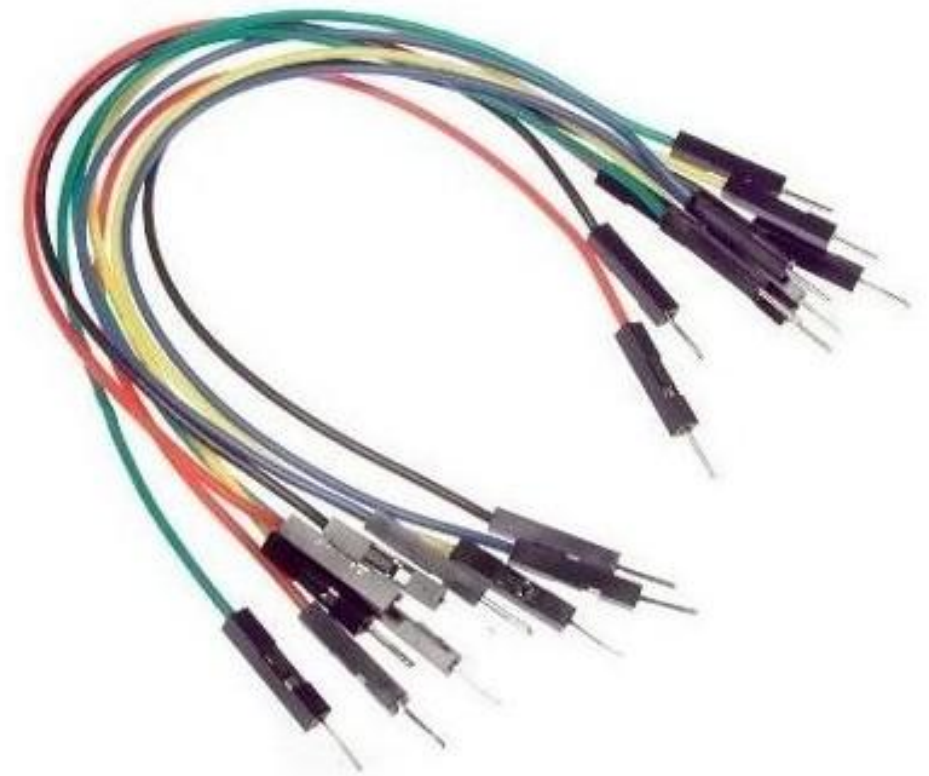
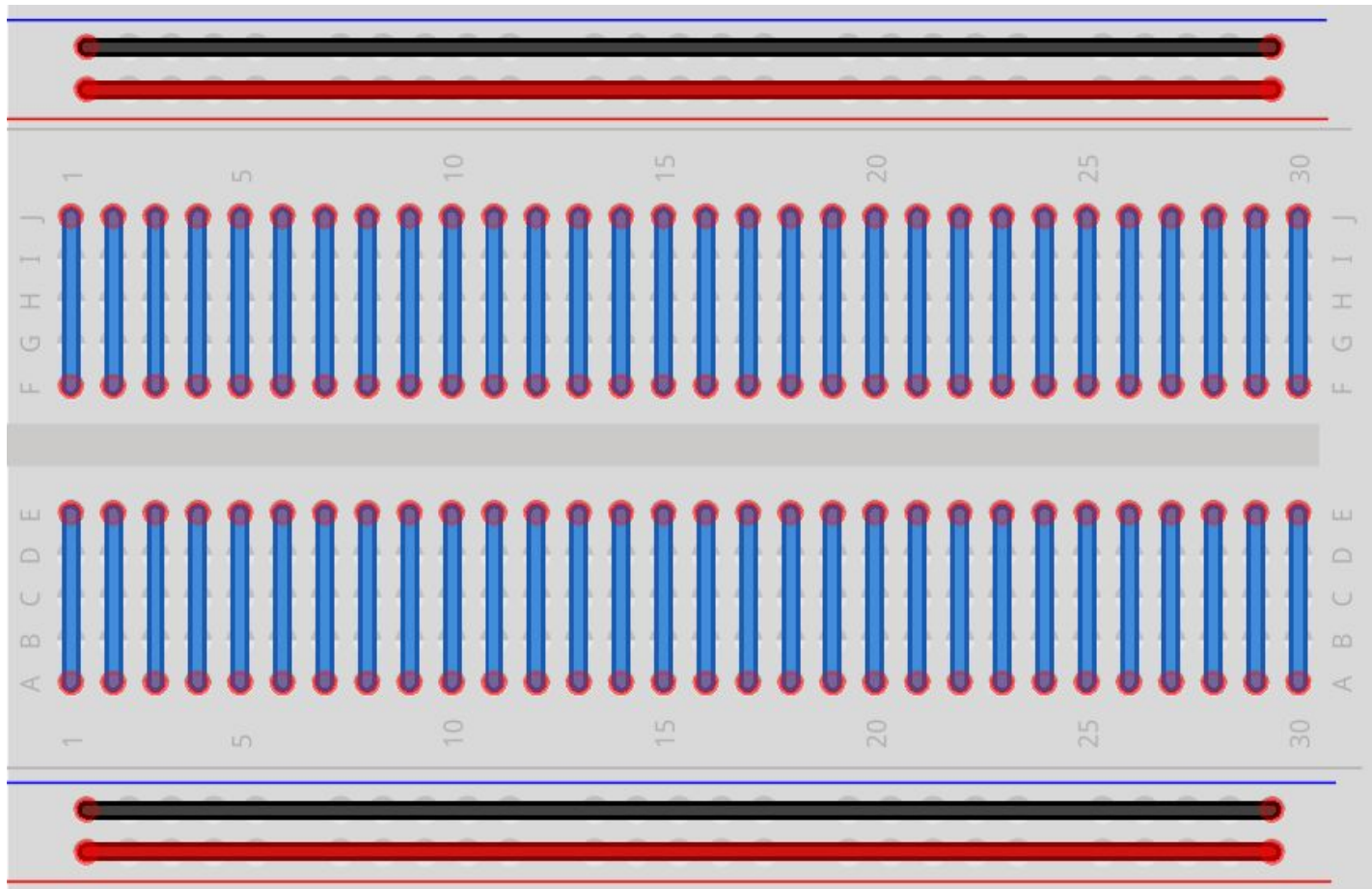
Configuração inicial

Loop de execução

# Primeiros Passos

# Primeiros Passos

# Protoboard e Jumpers



# Led

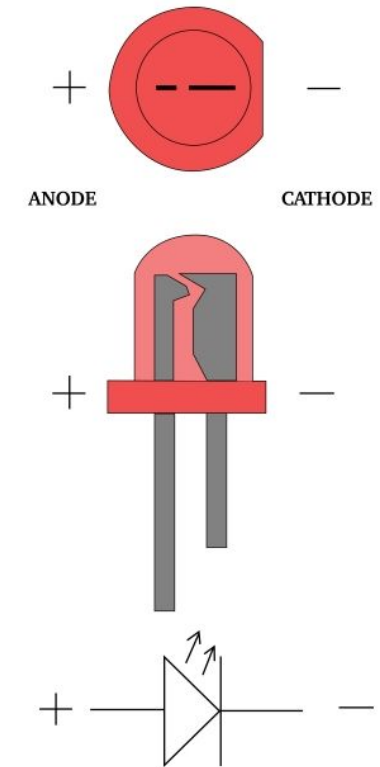
## ❑ Led Difuso

- ❑ Tensão 2V.
- ❑ Corrente 20 mA ~ 0,02 A

$$R = \frac{(V_{fonte} - V_{led})}{I_{led}}$$

- ❑ **R** = Resistência;
- ❑ **V<sub>fonte</sub>** = Tensão da fonte;
- ❑ **V<sub>led</sub>** = Tensão do Led;
- ❑ **I<sub>led</sub>** = Corrente do Led.

- ❑ <http://blog.novaeletronica.com.br/calculadora-online-resistor-limitador-led/>



# Resistor

4- Faixas

1.0 K $\Omega$   $\pm$ 5%

1st 2nd 3rd 4th

Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador Decimal		Tolerância
Preto	0	0	0	1	1	
Marrom	1	1	1	10	10	$\pm$ 1%
Vermelho	2	2	2	100	100	$\pm$ 2%
Laranja	3	3	3	1K	1.000	
Amarelo	4	4	4	10K	10.000	
Verde	5	5	5	100K	100.000	
Azul	6	6	6	1M	1.000.000	
Violeta	7	7	7	10M	10.000.000	
Cinza	8	8	8		100.000.000	
Branco	9	9	9		1.000.000.000	
Ouro					0.1	$\pm$ 5%
Prata					0.01	$\pm$ 10%
Branco						$\pm$ 20%

1st 2nd 3rd 4th 5th

254  $\Omega$   $\pm$ 1 %

5- Faixas

## Resistor de 4 faixas

- Faixa 1: Valor
- Faixa 2: Valor
- Faixa 3: Fator Multiplicador
- Faixa 4: Tolerância

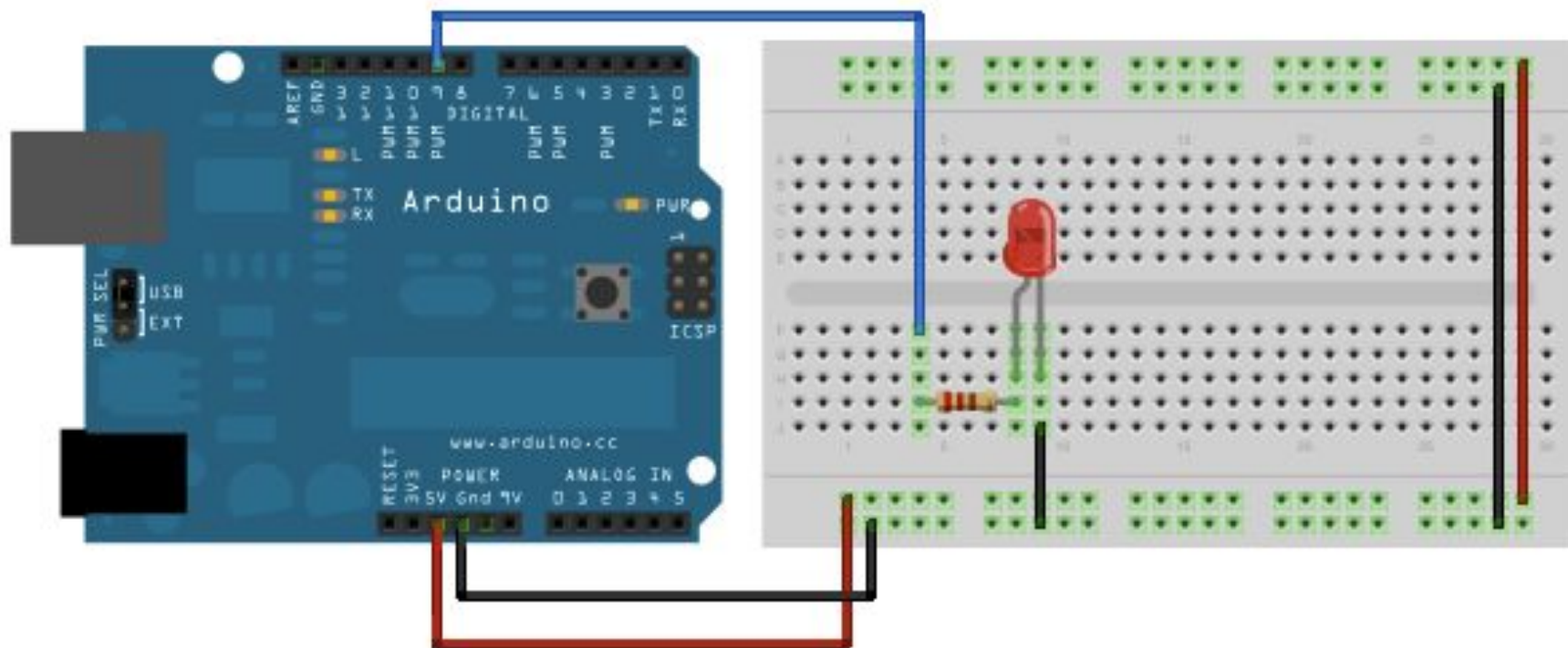
## Resistor de 5 faixas

- Faixa 1: Valor
- Faixa 2: Valor
- Faixa 3: Valor
- Faixa 4: Fator Multiplicador
- Faixa 5: Tolerância

[http://www.novaeletronica.com.br/ferramentas\\_online/cores-de-resistor-online.php](http://www.novaeletronica.com.br/ferramentas_online/cores-de-resistor-online.php)



# Pratica 1: Acionando um Led



# Prática 1: Acionando um Led

```
#define led1 9
Void setup(){
    pinMode(led1, OUTPUT);
}
Void loop(){
    digitalWrite(led1, HIGH);
    delay(1000);
    digitalWrite(led1, LOW);
    delay(1000);
}
```

## Prática 2: Semáforo

- ❑ Com base no exemplo anterior, monte um circuito que demonstre o funcionamento de um semáforo de trânsito. Esse deve obedecer os seguintes critérios:
  - ❑ A luz verde deve ficar acesa por 1,5 segundos;
  - ❑ A luz amarela deve ficar acesa por 1 segundo;
  - ❑ A luz vermelha deve ficar acesa por 3 segundos.
- ❑ Os leds devem ser acesos na sequência correta de funcionamento do semáforo de trânsito;
- ❑ Não deve haver mais de um led aceso ao mesmo tempo.



# Introdução ao C

# Introdução ao C

# História do C

- ❑ Criada por Dennis Ritchie em 1972 no centro de pesquisas da Bell Laboratories;
- ❑ 1ª utilização: reescrita do sistema operacional UNIX;
- ❑ 1980: vários compiladores C disponíveis;
- ❑ Linguagem imperativa de propósito geral;
- ❑ Algumas características:
  - ❑ Portabilidade;
  - ❑ Geração de código eficiente;
  - ❑ Simplicidade;
  - ❑ Facilidade de uso;
- ❑ O C é case sensitive.



# Estrutura Básica de um Programa em C

Diretivas para o pré-processamento

Declaração de variáveis globais

Main()

{

Declaração de variáveis

Comandos

}



# Estrutura Básica de um Programa em C

- ❑ Diretiva `#include` permite incluir uma biblioteca;
- ❑ Bibliotecas contêm funções pré-definidas, utilizadas nos programas;
- ❑ Exemplos:

<code>#include &lt;stdio.h&gt;</code>	Funções de entrada e saída
<code>#include &lt;stdlib.h&gt;</code>	Funções padrão
<code>#include &lt;math.h&gt;</code>	Funções matemáticas
<code>#include &lt;string.h&gt;</code>	Funções de texto

# Comentários em C

❑ Use comentários iniciados por `//` ou entre `/* */`

`/*` isso é um

Comentário `*/`

`//` isto também é um comentário

# Exemplo

```
/* meu primeiro programa C */  
#include <stdio.h>  
#include <stdlib.h>  
void main()  
{  
    printf("Hello world!"); //mostra o texto  
}
```

**Obs.: Todos os comandos devem terminar com ponto e virgula (;)**

# Variáveis

# Variáveis

# Declaração de Variáveis

- ❑ Os nomes das variáveis devem conter apenas letras, dígitos e o símbolo “\_”, podendo iniciar apenas com letra ou \_
- ❑ C diferencia letras maiúsculas de minúsculas!
- ❑ Para cada variável é necessário informar o tipo de dado armazenado.

# Declaração de Variáveis

- ❑ Os nomes das variáveis devem conter apenas letras, dígitos e o símbolo “\_”, podendo iniciar apenas com letra ou \_
- ❑ C diferencia letras maiúsculas de minúsculas!
- ❑ Para cada variável é necessário informar o tipo de dado armazenado.



# Tipos de Dados

# Tipos de Dados

- ❑ Valores inteiros (**int**);
- ❑ Valores reais:
  - ❑ Baixa precisão (**float**);
  - ❑ Alta precisão (**double**);
- ❑ Lógico (**bool**);
- ❑ Caratere (**char**).

# Modificadores de Tipos de Dados

- ❑ Modificadores são usados para alterar o significado de um tipo básico, adaptando às necessidades de diversas situações.
- ❑ Para caractere e inteiro:
  - ❑ Signed – com sinal;
  - ❑ Unsigned – sem sinal
  - ❑ Long – longo
  - ❑ Short curto
- ❑ Para double:
  - ❑ long

Tipo	Bits	Faixa
char	8	- 127 a 127
unsigned char	8	0 a 255
signed char	8	-127 a 127
int	16	-32.767 a 32.767
unsigned int	16	0 a 65.535
signed int	16	-32.767 a 32.767
short int	16	-32.767 a 32.767
unsigned short int	16	0 a 65.535
signed short int	16	-32.767 a 32.767
long int	32	-2.147.483.647 a 2.147.483.647
signed long int	32	-2.147.483.647 a 2.147.483.647
unsigned long int	32	0 a 4.294.967.295
float	32	Seis dígitos de precisão
double	64	Dez dígitos de precisão
long double	80	Dez dígitos de precisão

# Palavras Reservadas C

Auto	Double	Int	Struct
Break	Else	Long	Switch
Case	Enum	Register	Typedef
Char	Extern	Return	Union
Const	Float	Short	Unsigned
Continue	For	Signed	Void
Default	Goto	Sizeof	Volatile
Do	If	Static	while

# Declaração de Variáveis

- ❑ Sintaxe:

- ❑ Tipo identificador;

- ❑ `int x;`

- ❑ `int` é o tipo e `x` é o identificador;



# Variáveis Locais e Globais

- ❑ Uma variável local é declarada e está disponível somente dentro de um escopo específico. Ex. dentro de uma função.
- ❑ Uma variável global está disponível ao longo de toda a execução do programa, podendo ser consultado ou modificada quando se achar necessário.

# Constantes

- ❑ Como uma variável, uma constante também é uma posição de memória à qual devem ser associados um identificador e um tipo de dado;
- ❑ O que caracteriza uma constante é o fato de que o conteúdo de uma constante não pode ser modificado durante a execução do programa;
- ❑ Este conteúdo é fixado quando da declaração da constante o que deve ser feito de acordo com a seguinte sintaxe:

**Const Tipo de dado Identificador = valor;**

**Saída de Dados**

**Saída de Dados**

# Saída de Dados

- ❑ Necessário a biblioteca `stdio.h` para entrar e sair com dados;
  - ❑ `printf("expressão de controle", argumentos);`

# Saída de Dados

- ❑ Tamanho de campos na impressão:
  - ❑ `printf("\n%2d", 350);`
- ❑ Para arredondamento:
  - ❑ `printf("\n%4.2f", 3456.785);`
- ❑ Para alinhamento:
  - ❑ `printf("\n%10.2f %10.2f", 350.75, 8.0);`
- ❑ Complemento de zeros a esquerda:
  - ❑ `printf("\n%04d", 21);`
- ❑ Impressão de caracteres:
  - ❑ `printf("\n%d %c %x %o \n", 'A', 'A', 'A', 'A');`

# Códigos de Formatação de Saída de Dados

\n	Nova linha	%c	Caractere simples
\t	Tabulação	%d	Decimal
\b	Retrocesso	%e	Notação científica
\”	Aspas	%f	Ponto flutuante
\\	Barra	%o	Octal
\0	Nulo	%s	Cadeia de caracteres
		%u	Decimal sem sinal
		%x	Hexadecimal

# Operadores

# Operadores

# Tipos de Operadores

- ❑ Operadores Aritméticos;
- ❑ Operadores de Atribuição;
- ❑ Operadores de Comparação;
- ❑ Operadores Lógicos;



# Operadores Aritméticos

Nome	Operador
Soma	+
Subtração	-
Divisão	/
Multiplicação	*
Potência	**

# Operadores de Atribuição

Nome	Operador	Significado
Atribuição	$x = y$	$x = y$
Atribuição com adição	$x += y$	$x = x + y$
Atribuição com subtração	$x -= y$	$x = x - y$
Atribuição com multiplicação	$x *= y$	$x = x * y$
Atribuição com divisão	$x /= y$	$x = x / y$
Atribuição com resto	$x \% = y$	$x = x \% y$

# Operadores de Comparação

Nome	Operador
Igualdade	<code>==</code>
Desigualdade	<code>!=</code>
Maior	<code>x &gt; y</code>
Maior igual	<code>x &gt;= y</code>
Menor	<code>x &lt; y</code>
Menor igual	<code>x &lt;= y</code>

# Operadores Lógicos

Nome	Operador
E (and)	&&
Ou (or)	
Não (not)	!

# Entrada de Dados

# Entrada de Dados

- ❑ Utilizar a biblioteca `stdio.h` para entrada e saída de dados.
  - ❑ `scanf("expressão de controle", argumentos);`
  - ❑ `scanf("%_", &);`
- ❑ Argumentos: endereço das variáveis que receberão os dados. Deve ser utilizado `&` para referenciar o endereço.
  - ❑ `Int num;`
  - ❑ `Scanf("%d", &num);`

# Entrada de Dados Caracteres

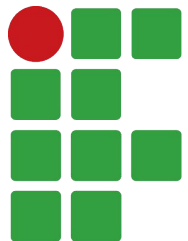
- ❑ É recomendado a utilização da função **fgets** para captura da entrada de textos, caso contrário, caso a string informada contenha espaços, essa só será considerada até o primeiro espaço.
  - ❑ `fflush(stdin);` //limpeza do buffer de entrada
  - ❑ `fgets(msg, sizeof(msg), stdin);` //recebe um texto como entrada do usuário
  - ❑ Contagem de caracteres `strlen(variável)`.

# Exercícios

1. Crie um programa com as seguintes instruções:
  - a. Você deve criar pelo menos 4 tipos diferentes de variáveis;
  - b. Das variáveis criadas 2 devem ser numéricas;
  - c. Deve ser solicitado ao usuário que informe valores para as 4 variáveis;
  - d. Calcule a soma dos valores informados e mostre na tela;
  - e. Faça a multiplicação dos valores e mostre o resultado na tela;
  - f. Para cada uma das operações anteriores você deve criar uma variável para guardar o resultado;







**INSTITUTO FEDERAL**

Santa Catarina  
Câmpus Tubarão

Obrigado!

Fernando Silvano Gonçalves

[fernando.goncalves@ifsc.edu.br](mailto:fernando.goncalves@ifsc.edu.br)

[se.cst.tub@ifsc.edu.br](mailto:se.cst.tub@ifsc.edu.br)