

**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	29-fev.	04	Recepção e Apresentação do Unidade / Apresentação do Plano de Ensino / Avaliação Diagnóstica / Introdução a sistemas embarcados
2	02-mar.	04	Conceitos e Características e Aplicações de Sistemas Embarcados / Histórico de Sistemas Embarcados / Práticas com Arduino
3	07-mar.	04	Microcontroladores, Microprocessadores / Periféricos / Introdução ao Arduino / Introdução ao C
4	14-mar.	04	Introdução à Linguagens de Programação
5	21-mar.	04	Linguagem de Programação C
6	28-mar.	04	Linguagens de Programação C para arduino
7	04-abr.	04	Variáveis e Operadores
8	11-abr.	04	Estruturas Condicionais
9	18-abr.	04	Estruturas de Repetição
10	25-abr.	04	Avaliação 01

# Cronograma

Encontro	Data	Nº Aulas	Conteúdo
11	02-mai.	04	Microcontroladores
12	09-mai.	04	Entradas e Saídas Digitais
13	16-mai.	04	Conversor Analógico-Digital
14	18-mai.	04	Sensores
15	23-mai.	04	Comunicação Serial
16	06-jun.	04	PWM
17	13-jun.	04	Temporizadores
18	20-jun.	04	Interrupções
19	27-jun.	04	Avaliação 02
20	04-jul.	04	Conselho de Classe / Atividades de Encerramento da UC
		80	

# Pauta

- Introdução ao C;
- Variáveis;
- Operadores;
- Comunicação Serial;
- Estruturas Condicionais;
- Entradas.

# 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)	!

# Exercícios

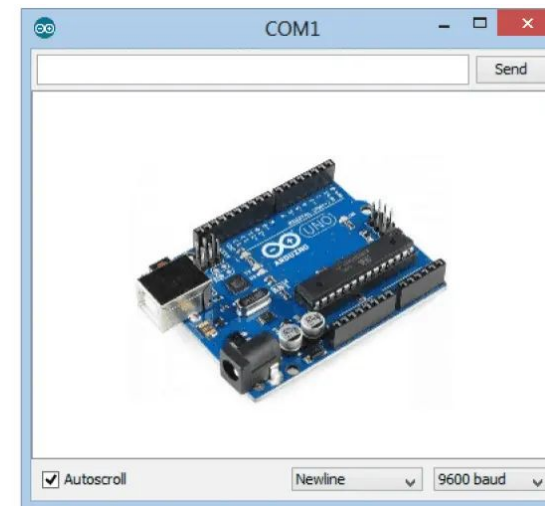
1. Crie um programa com as seguintes instruções:
  - a. Você deve criar pelo menos 4 variáveis numéricas;
  - b. Calcule a soma dos dois primeiros valores informados e mostre na tela;
  - c. Faça a multiplicação do terceiro e quarto valor e mostre o resultado na tela;
  - d. Faça a subtração do primeiro e do terceiro valor e mostre o resultado na tela;
  - e. Para cada uma das operações anteriores você deve criar uma variável para guardar o resultado;





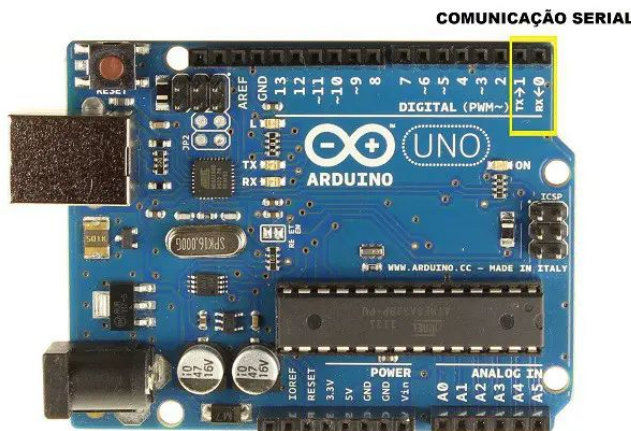
# Comunicação Serial

# Comunicação Serial



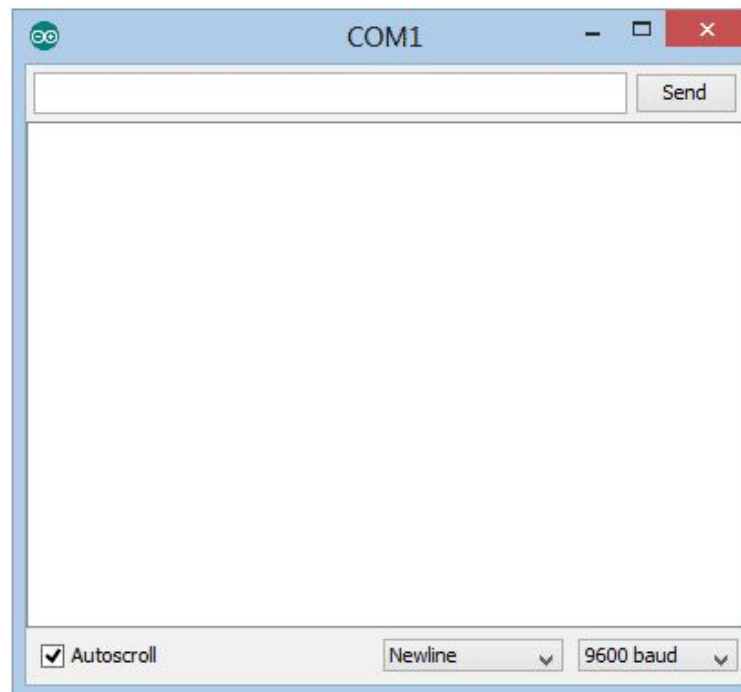
# Comunicação Serial

- ❑ A comunicação serial (UART) é um poderoso recurso que possibilita a comunicação entre a placa e um computador, ou entre a placa e outro dispositivo;
- ❑ É por meio desse canal que é realizado o upload do código para a placa.
- ❑ Esse canal está ligado aos pinos digitais 0 (RX) e 1 (TX).
- ❑ Esses mesmos pinos estão ligados ao microcontrolador ATMEGA16U2, responsável pela tradução do sinal para comunicação USB com o computador.



# Terminal Serial

- ❑ A IDE do Arduino possui um terminal serial que auxilia no recebimento e envio de dados para a placa sem a necessidade de recorrer a uma ferramenta externa;
- ❑ Para acessar essa ferramenta basta clicar no ícone Serial Monitor ou acessar o menu Tools> Serial Monitor.



# Comunicação Serial - Funções mais Utilizadas

- ☐ `Serial.begin()`
- ☐ `Serial.available()`
- ☐ `Serial.read()`
- ☐ `Serial.print()`
- ☐ `Serial.println()`
- ☐ `Serial.write()`

# Comunicação Serial

```
1  /*
2   echo
3   reenvia para o computador o dado recebido pela serial
4  */
5
6  byte byteRead;
7
8  void setup() {
9    //configura a comunicação serial com baud rate de 9600
10   Serial.begin(9600);
11 }
12
13 void loop() {
14
15   if (Serial.available()) //verifica se tem dados disponível para leitura
16   {
17     byteRead = Serial.read(); //le byte mais recente no buffer da serial
18     Serial.write(byteRead);   //reenvia para o computador o dado recebido
19   }
20 }
```

# Comunicação Serial

```
1  /*
2   * comandos via serial
3   * inverte o estado do led conctado a saída 13 do arduino quando recebe o caracter 'A' pela serial
4   */
5
6  const int LED = 13;
7
8  void setup() {
9      Serial.begin(9600);    //configura comunicação serial com 9600 bps
10     pinMode(LED,OUTPUT);   //configura pino do led como saída
11 }
12
13 void loop() {
14     if (Serial.available()) //se byte pronto para leitura
15     {
16
17         switch(Serial.read())    //verifica qual caracter recebido
18         {
19             case 'A':            //caso 'A'
20
21                 digitalWrite(LED,!digitalRead(LED)); //inverte estado do LED
22
23                 break;
24             }
25     }
26 }
```

# Prática com Comunicação Serial

- Crie uma aplicação com dois leds;
- Configure a comunicação serial;
- Escreva uma aplicação que ao receber os valores “X” e “Z”, acione respectivamente cada um dos leds adicionados ao circuito;
- Caso o Led já esteja acionado, você deve alterar o seu estado para desligado.

**Entrada de Dados Via Serial**

# Entrada de Dados Via Serial



# Entrada de Dados Via Serial

- ❑ Para podermos receber dados via Serial, você deve previamente saber o tipo de dado que o usuário irá informar;
- ❑ Isto ocorre pois a leitura de cada tipo de dado é diferente e deve ser informada pelo programador;

`Serial.parseInt()`

`Serial.parseFloat()`

`Serial.readString()`

# Exemplo Entrada de Dados Via Serial

Média

# Prática com Entrada de Dados Via Serial

- ❑ Crie um programa que receba 3 notas de 4 alunos. Calcule a média de cada aluno e a média da turma. Ao final, apresente as notas dos alunos, as médias e a média da turma.
- ❑ Crie um programa que receba 3 valores. Calcule a soma dos dois primeiros. O resultado obtido deve ser multiplicado pelo terceiro. Ao final, apresente os números informados e o resultado da operação.


# Estruturas Condicionais

# Estruturas de Decisão

## **Decisão Simples;**

-  Inclui apenas uma condição;

## **Decisão Composta;**

-  Mais de uma condição pode ser avaliada, porém as verificações só ocorrem caso a condição anterior não seja satisfeita;

## **Decisão Múltipla;**

-  Diferentes análises para a mesma variável;

# Decisão Simples

```
if(expressão lógica){  
    sequência de comandos  
}
```

- ❑ **Expressão lógica:** deve representar uma tomada de decisão (**Verdadeiro** – **Falso**);
- ❑ **Sequência de comandos:** só é executada quando a condição é satisfeita, pode ser um único comando ou uma sequência destes;

# Decisão Simples

```
if(x%2 == 0){
```

```
    Serial.print("O número é par!");
```

```
}
```

# Praticando com Decisão Simples

- ❑ Crie um programa, composto por 3 variáveis para armazenar notas;
- ❑ Você deve informar via serial o valor para as notas;
- ❑ **Calcule a média das notas** e apresente resultado conforme abaixo:
  - ❑ Você deve imprimir as notas informadas e a média;
  - ❑ Caso o aluno tenha média maior que 7, deve ser apresentado a mensagem **aprovado**;



# Praticando com Decisão Simples

- ❑ Crie um programa que receba a altura de 3 pessoas;
- ❑ Ao final mostre na tela os valores recebidos e diga qual das três pessoas é a mais alta.

# Decisão Composta

```
if(expressão lógica){  
    sequência de comandos 1  
} else {  
    sequência de comandos 2  
}
```

- ❑ **Expressão lógica:** deve representar uma tomada de decisão (**Verdadeiro** – **Falso**);
- ❑ **Sequência de comandos 1:** só é executada quando a condição é satisfeita, pode ser um único comando ou uma sequência destes;
- ❑ **Sequência de comandos 2:** só é executada quando a condição **não é satisfeita**, pode ser um único comando ou uma sequência destes;

# Decisão Composta

```
if(x%2 == 0){  
    Serial.print("O número é Par!");  
} else {  
    Serial.print("O número é Impar!");  
}
```

# Decisão Composta

```
if(x == 'a'){  
    Serial.print("Encontrou a");  
} else if(x == 'b'){  
    Serial.printf("Encontrou b");  
} else {  
    if(x == 'c'){  
        Serial.print("Encontrou c");  
    } else{  
        Serial.print("Não encontrou!");  
    }  
}  
}
```

# Decisão Múltipla

```
switch(expressão de seleção){  
  case exp1:  
    sequência de comandos 1;  
    break;  
  case exp2:  
    sequência de comandos 2;  
    break;  
}
```

```
case expN:  
  sequência de comandos N;  
  break;  
default:  
  sequência de comandos default;  
  break;  
}
```

- ❑ **Expressão de seleção:** deve resultar em uma constante;
- ❑ **expN:** constante de verificação;
- ❑ **Sequência de comandos N:** pode ser um único comando ou uma sequência;
- ❑ **Default:** condição executada caso nenhuma das anteriores seja satisfeita;
- ❑ **break:** finaliza o bloco de verificação.

# Decisão Múltipla

```
switch(x){  
    case 'a':  
        Serial.print("Encontrou a");  
        break;  
    case 'b':  
        Serial.print("Encontrou b");  
        break;  
    case 'c':  
        Serial.print("Encontrou c");  
        break;
```

```
    default:  
        Serial.print("Não encontrou!");  
        break;  
}
```

# Praticando com Decisão Composta

- ❑ Crie uma programa que **receba 3 notas, calcule a média e apresente resultado conforme abaixo:**
  - ❑ Caso o aluno tenha média maior que 7, deve ser apresentado a mensagem **aprovado**;
  - ❑ Caso a nota seja menor ou igual a 7, deve ser apresentado a mensagem **reprovado**;

# Praticando com Decisão Múltipla

- ❑ Com base no conteúdo apresentado, crie os seguintes programas:
  - ❑ **Crie um programa que receba um número:**
    - ❑ Se o número recebido for **par** você deve imprimir o seu valor ao quadrado;
  - ❑ **Crie um programa que receba um número de 1 a 7:**
    - ❑ Conforme o número recebido você deve imprimir o dia da semana correspondente;
  - ❑ **Crie um programa que receba um número de 1 a 12:**
    - ❑ Conforme o número recebido você deve imprimir o mês do ano correspondente;



# Praticando com Decisão Múltipla

- ❑ Crie um programa que contenha 4 leds;
- ❑ Você deve receber um valor numérico via serial e conforme o valor recebido faça as seguintes operações:
  - ❑ **Se o valor recebido for 2**, acione o primeiro led;
  - ❑ **Se o valor recebido for 4**, acione o segundo led;
  - ❑ **Se o valor recebido for 8**, acione o terceiro led;
  - ❑ **Se o valor recebido for 10**, acione o quarto led;
  - ❑ **Se o valor recebido for 24**, acione o primeiro e o segundo led;
  - ❑ Você deve criar as outras combinações para acionar os demais leds.
  - ❑ **OBS.: Caso o led esteja acionado o mesmo deve ser desligado.**

# Entradas

# Entradas

# Entradas

```
pinMode(A1, INPUT);  
pinMode(9, INPUT);
```

## ❑ Entrada Analógica:

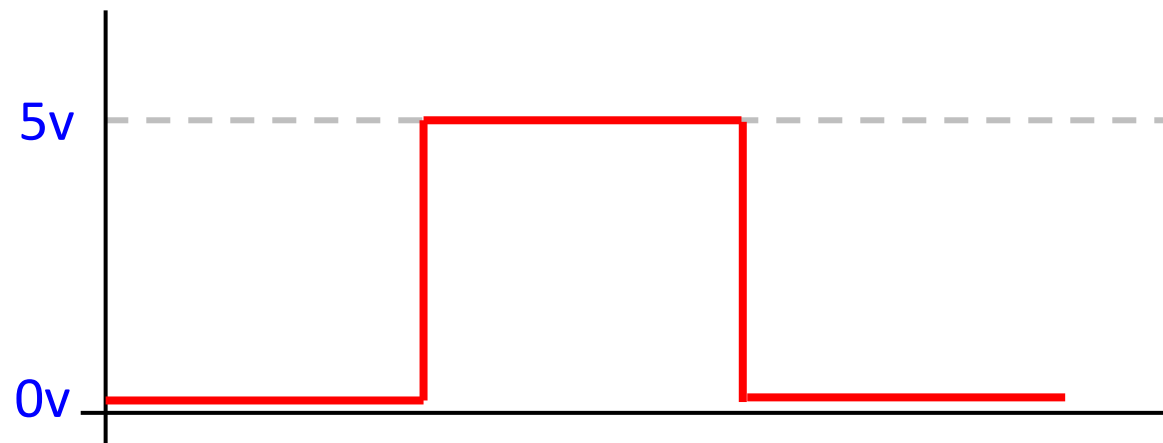
- ❑ Valores entre 0 – 1023;

## ❑ Entrada Digital

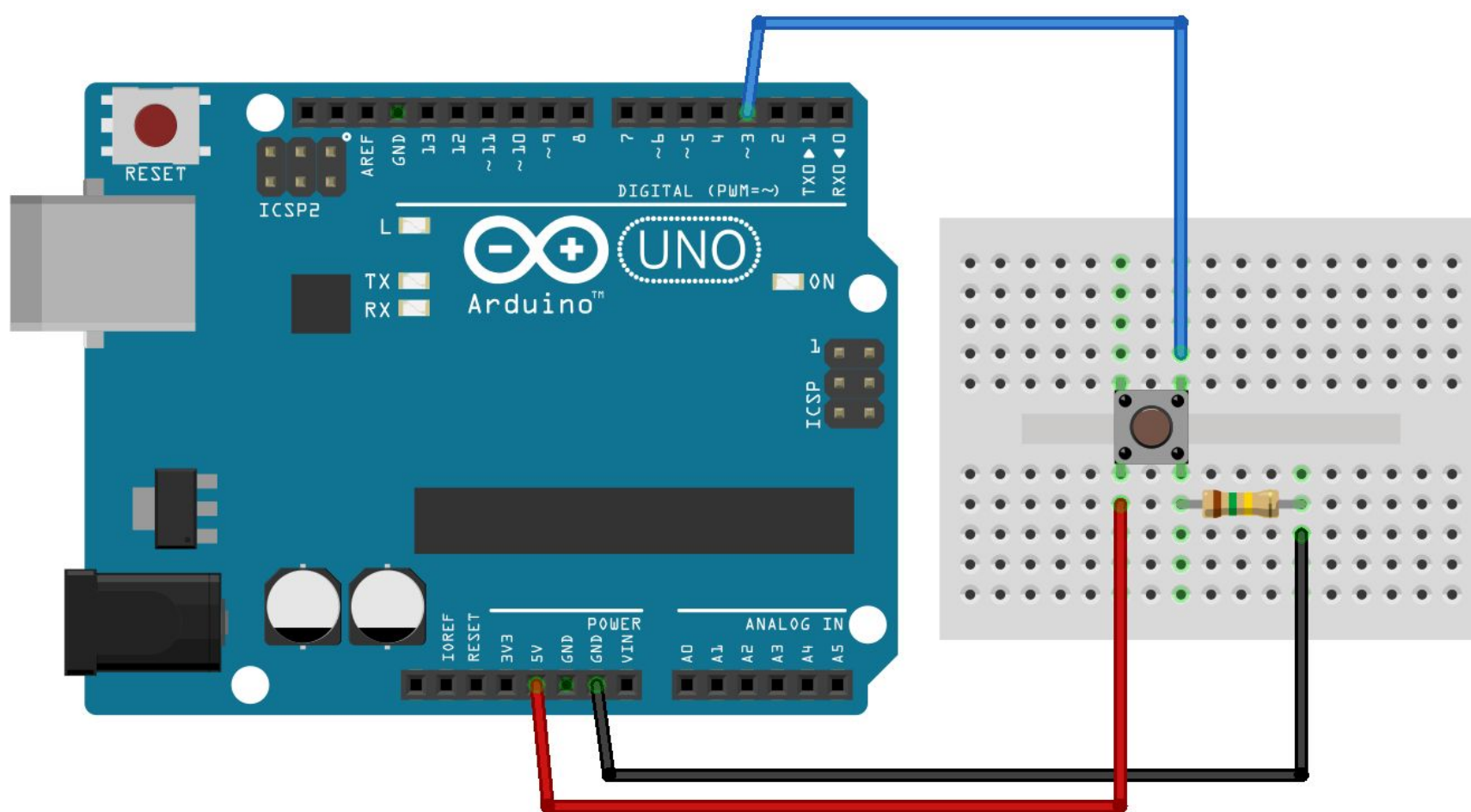
- ❑ Valor lógico 0 ou 1;

# Entradas Digitais

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



# Configurando Uma Entrada Digital



fritzing

# Acionando um Led Utilizando Botão

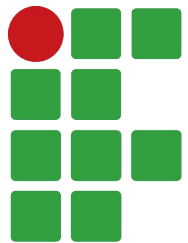
```
#define pb 3
#define led 9
int pbValue;
int ledState;

void setup() {
  Serial.begin(9600);
  pinMode(pb, INPUT);
  pinMode(led, OUTPUT);
  ledState = LOW;
}
```

```
void loop(){
  pbValue = digitalRead(pb);
  Serial.println(pbValue);
  if(pbValue == 1){
    if(ledState == HIGH){
      ledState = LOW;
    } else {
      ledState = HIGH;
    }
    digitalWrite(led, ledState);
  }
  delay(300);
}
```

# Atividade Com Botão e Leds

- ❑ Crie uma aplicação que faça piscar os três leds em sequência, alternando a cada 1 segundo.
  - Adicione um Botão ao seu projeto que quando pressionado, inverte o sentido que os leds estão piscando.



**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)