OFICINA DE ROBÓTICA
INVENTAR E RECICLAR PARA EDUCAR
oficinaderobotica.ufsc.br

Oficina de Robótica

Programação Básica em Arduino – Aula 2

Execução:

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM
Laboratório de Automação
e Robótica Móvel

Variáveis

- Variáveis são lugares (posições) na memória principal que servem para armazenar dados.
- As variáveis são acessadas através de um identificador único.
- O conteúdo de uma variável pode variar ao longo do tempo durante a execução de um programa.
- Uma variável só pode armazenar um valor a cada instante.
- Um identificador para uma variável é formado por um ou mais caracteres, obedecendo a seguinte regra: o primeiro caractere deve, obrigatoriamente, ser uma letra.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Variáveis

- ATENÇÃO!!!
  - **Um identificador de uma variável ou constante não pode ser formado por caracteres especiais ou palavras reservadas da linguagem.**

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM

Tipos de Dados

- Tipos de dados definem:
  - A quantidade de memória que uma variável ou constante irá ocupar;
  - As operações que podem ser executadas sobre uma variável ou constante de determinado tipo;
  - A faixa de valores que uma variável ou constante pode armazenar;
  - O modo como o valor armazenado será interpretado.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Tipos de Dados

- Tipos de Variáveis no Arduino

Tipo	Definição
void	Indica tipo indefinido. Usado geralmente para informar que uma função não retorna nenhum valor.
boolean	Os valores possíveis são true (1) e false (0). Ocupa um byte de memória.
char	Ocupa um byte de memória. Pode ser uma letra ou um número. A faixa de valores válidos é de -128 a 127.
unsigned char	O mesmo que o **char**, porém a faixa de valores válidos é de 0 a 255.
byte	Ocupa 8 bits de memória. A faixa de valores é de 0 a 255.
int	Armazena números inteiros e ocupa 16 bits de memória (2bytes). A faixa de valores é de -32.768 a 32.767.
unsigned int	O mesmo que o **int**, porém a faixa de valores válidos é de 0 a 65.535.
word	O mesmo que um **unsigned int**.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Tipos de Dados

- Tipos de Variáveis no Arduino

Tipo	Definição
long	Armazena números de até 32 bits (4 bytes). A faixa de valores é de -2.147.483.648 até 2.147.483.647.
unsigned long	O mesmo que o **long**, porém a faixa de valores é de 0 até 4.294.967.295.
short	Armazena número de até 16 bits (2 bytes). A faixa de valores é de -32.768 até 32.767.
float	Armazena valores de ponto flutuante (com vírgula) e ocupa 32 bits (4 bytes) de memória. A faixa de valores é de -3.4028235E+38 até 3.4028235E+38
double	O mesmo que o **float**.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Operadores

- Em uma linguagem de programação existem vários operadores que permitem operações do tipo:
  - Aritmética
  - Relacional
  - Lógica
  - Composta

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM

Operadores

- Operadores aritméticos

Símbolo	Função
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão inteira)

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM

Operadores

- Operadores relacionais

Símbolo	Função
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
==	Igual
!=	Diferente

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM

Operadores

- Operadores lógicos

Símbolo	Função
&&	E (and)
||	OU (or)
!	Não (not)

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM

Operadores

- Operadores compostos

Símbolo	Função
++	Incremento
--	Decremento
+=	Adição com atribuição
-=	Subtração com atribuição
*=	Multiplicação com atribuição
/=	Divisão com atribuição

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Operadores

- Operador de Atribuição
  - A atribuição de valores a variáveis e constantes é feita com o uso do operador de atribuição (=).
  - O operador de atribuição coloca o valor situado à sua direita dentro do objeto localizado à sua esquerda.
  - Exemplos:
    - int valor = 100;
    - const float pi = 3.14;
  - **Atenção!!!**
    - O operador de atribuição não vale para o comando ***#define***.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Operadores

- Usando o operador de atribuição

atribuicao

```
int numero = 1; // inicialização

void setup()
{
  Serial.begin(9600);
  Serial.print("A variavel 'numero' vale: ");
  Serial.println(numero);
  delay(2000);
  numero = 5; // atribuição
  Serial.print("Agora a variavel 'numero' vale: ");
  Serial.println(numero);
  delay(2000);
}

void loop()
{
  numero = numero + 1; // atribuição
  Serial.print("Agora a variavel 'numero' vale: ");
  Serial.println(numero);
  delay(2000);
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Monitor Serial

- O monitor serial é utilizado para comunicação entre o Arduino e o computador (PC).
- O monitor serial pode ser aberto no menu *tools* opção *serial monitor*, ou pressionando as teclas CTRL+SHIFT+M.
- As principais funções do monitor serial são: *begin()*, *read()*, *write()*, *print()*, *println()* e *available()*.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Monitor Serial

- Algumas funções bastante usadas:
  - *begin()*: inicializa a comunicação entre o Arduino e um computador;
  - *read()*: recebe caracteres inseridos no monitor serial;
  - *print()*: imprime caracteres no monitor serial;
  - *println()*: imprime caracteres no monitor serial, mas causa uma quebra de linha no final;
  - *available()*: retorna o número de bytes disponíveis no buffer de leitura do monitor serial.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Monitor Serial

- Imprimindo uma mensagem no monitor serial

monitor_serial §

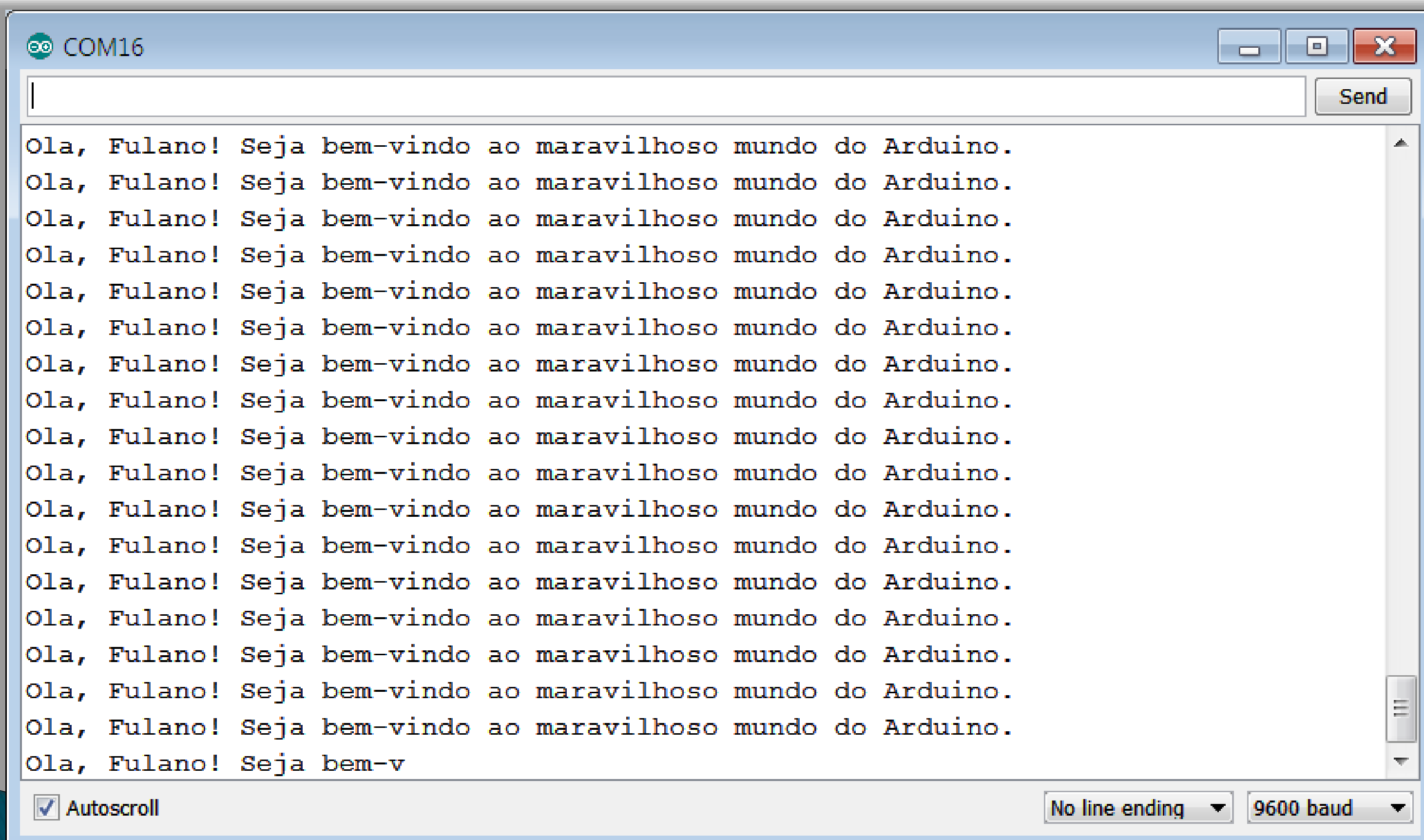
```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print("Ola, Fulano! Seja bem-vindo ao ");
  Serial.println("maravilhoso mundo do Arduino.");
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Monitor Serial

- Saída no monitor serial



UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Comandos de Repetição

- Muitas vezes é necessário repetir uma determinada instrução ou conjunto de instruções.
- Os comandos de repetição mantêm em um "laço" uma instrução ou conjunto de instruções enquanto uma condição estiver sendo satisfeita.
- O Arduino possui 3 comandos de repetição:
  - while ( )
  - for (; ;)
  - do while ( )

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM
Laboratório de Automação
e Robótica Móvel

Comandos de Repetição

- while ( )
  - Este comando de repetição avalia uma expressão no início do laço e, caso esta seja verdadeira, a(s) instrução(ções) dentro do "laço" permanecem sendo executadas.
  - Sintaxe do comando while:

```
while (expr) {
  cmd;
}
```

  - onde:
    - *expr* – é uma expressão que pode ser lógica, relacional ou aritmética.
    - *cmd* – um ou mais comandos a serem repetidos.
  - A permanência de execução do "laço" é garantida enquanto a expressão for verdadeira.

Como a avaliação da expressão é realizada no início do laço, pode ser que o *cmd* não execute nenhuma vez.

Comandos de Repetição

- while ( )

while

```
char ch;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  while (Serial.available() == 0) {
    Serial.println("Informe um caractere.");
    delay(300);
  }

  ch = Serial.read();
  Serial.print("ch = ");
  Serial.println(ch);
  delay(2000);
}
```

Comandos de Repetição

- while ( )

semaforo

```
/**********************************
* Nome do Programa: Semáforo     *
*                                *
* Descrição:                     *
*    Mantém o sinal aberto até   *
*    que um pedestre solicite a  *
*    travessia. Então fecha o    *
*    sinal por alguns segundos.  *
**********************************/

const int BOTAO = 6;
const int VERDE = 8;
const int AMARELO = 9;
const int VERMELHO = 10;

void setup()
{
  pinMode(VERDE, OUTPUT);
  pinMode(AMARELO, OUTPUT);
  pinMode(VERMELHO, OUTPUT);
  pinMode(BOTAO, INPUT_PULLUP);
}

void loop()
{
  // abre o sinal
  digitalWrite(VERDE, HIGH);

  // aguarda pedestre
  while (digitalRead(BOTAO))
    ; // instrução vazia

  // muda para o amarelo
  digitalWrite(VERDE, LOW);
  digitalWrite(AMARELO, HIGH);
  delay(4000);

  // fecha o sinal
  digitalWrite(AMARELO, LOW);
  digitalWrite(VERMELHO, HIGH);
  delay(8000);

  // abre novamente
  digitalWrite(VERMELHO, LOW);
}
```

Comandos de Repetição

- while ( )

while_2

```
int contador;

void setup()
{
  Serial.begin(9600);

  contador = 0;
  while (contador < 10) {
    Serial.println("Oficina de Robotica");
    contador++;
  }
}

void loop() {}
```

Comandos de Repetição

- for (; ;)
  - Este comando de repetição deve ser utilizado quando se sabe a quantidade de vezes que uma determinada instrução deve ser executada.
  - O comando for permite que escrevamos de forma mais organizada os laços de repetição baseados em um contador.
  - Sintaxe do comando for:

```
for (inicialização; condição; incremento) {
  cmd;
}
```

  - onde:
    - *inicialização* – inicialização do contador;
    - *condição* – é uma expressão relacional ou lógica;
    - *incremento* – atualização do contador.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
LARM
Laboratório de Automação e Robótica Móvel

Comandos de Repetição

- for (; ;)

for

```
int vezes = 10; // qtde de vezes que a mensagem será impressa
int contador; // irá contar quantas vezes a mensagem já foi impressa

void setup()
{
  Serial.begin(9600);

  for (contador = 0; contador < vezes; contador++) {
    Serial.println("Testando o comando de repeticao for()");
  }
}

void loop()
{

}
```

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM
Laboratório de Automação
e Robótica Móvel

Comandos de Repetição

- **Nota**
  - É possível declarar o contador dentro do cabeçalho do laço for.

```
for (int i = 0; i < 50; i++) {
  Serial.println("Oficina de Robotica");
}
```

Comandos de Repetição

- do while ( )
  - Semelhante ao comando while. A única diferença é que a condição é testada no final do laço.

  - Sintaxe do comando do while:

```
do {
  cmd;
} while (expr) ;
```

Como a avaliação da expressão é realizada no final do laço, é garantido que o *cmd* será executado pelo menos uma vez.

Comandos de Seleção

- Em vários momentos em um programa precisamos verificar uma determinada condição afim de selecionar uma ação ou ações que serão executadas.
- Um comando de seleção também é conhecido por desvio condicional, ou seja, dada um condição, uma parte do programa é executada.
- Os comandos de seleção podem ser do tipo:
  - Seleção simples
  - Seleção composta
  - Seleção de múltipla escolha

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Comandos de Seleção

- Seleção simples
  - Um comando de seleção simples avalia uma condição, ou expressão, para executar uma ação ou conjunto de ações.
  - No Arduino o comando de seleção simples é:

```
if (expr) {
   cmd
}
```

  - onde:
    - ***expr*** – representa uma expressão a ser avaliada que poc ser do tipo lógica, relacional ou aritmética. O resultado avaliação de uma expressão é sempre um valor lógico.
    - ***cmd*** – comando(s) a ser executado.

LARM

Comandos de Seleção

- Seleção simples
  - Acendendo leds pelo monitor serial

selecao_simples

```
#define LED_VERDE    5
#define LED_AMARELO  6
#define LED_VERMELHO 7

char opcao;

void setup()
{
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_AMARELO, OUTPUT);
  pinMode(LED_VERMELHO, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  opcao = Serial.read();

  if (opcao == 'G')
    digitalWrite(LED_VERDE, HIGH);
  if (opcao == 'Y')
    digitalWrite(LED_AMARELO, HIGH);
  if (opcao == 'R')
    digitalWrite(LED_VERMELHO, HIGH);
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Comandos de Seleção

- Seleção simples
  - Verificando se há caracteres no buffer de leitura

serial_available

```
char caractere;

void setup()
{
   Serial.begin(9600);
}

void loop()
{
   if (Serial.available()) {
      caractere = Serial.read();
      Serial.print("Caractere lido: ");
      Serial.println(caractere);
   }
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Comandos de Seleção

- Seleção composta
  - Um comando de seleção composta é complementar ao comando de seleção simples.
  - O objetivo é executar um comando mesmo que a expressão avaliada pelo comando *if (expr)* retorne um valor falso.
  - No Arduino o comando de seleção composta é:

```
if (expr) {
    cmd;
}
else {
    cmd;
}
```

  - onde:
    - ***expr*** – representa uma expressão a ser avaliada que pode ser do tipo lógica, relacional ou aritmética. O resultado da avaliação de uma expressão é sempre um valor lógico.
    - ***cmd*** – comando(s) a ser executado.

LARM

Comandos de Seleção

- Seleção composta
  - Verificando se há caracteres no buffer de leitura

selecao_composta_01

```
char caractere;

void setup()
{
   Serial.begin(9600);
}

void loop()
{
   if (Serial.available()) {
      caractere = Serial.read();
      Serial.print("Caractere lido: ");
      Serial.println(caractere);
   }
   else {
      Serial.println("Insira um caractere!");
   }
   delay(400);
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Comandos de Seleção

- Seleção composta (Comandos if aninhados)
  - Acendendo e apagando leds pelo monitor serial

selecao_composta_02

```
#define LED_VERDE    2
#define LED_AMARELO  3
#define LED_VERMELHO 4

char opcao;

void setup()
{
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_AMARELO, OUTPUT);
  pinMode(LED_VERMELHO, OUTPUT);
  Serial.begin(9600);
}
```

```
void loop()
{
  if (Serial.available()) {
    opcao = Serial.read();

    if (opcao == 'G')
      digitalWrite(LED_VERDE, HIGH);
    else if (opcao == 'g')
      digitalWrite(LED_VERDE, LOW);
    else if (opcao == 'Y')
      digitalWrite(LED_AMARELO, HIGH);
    else if (opcao == 'y')
      digitalWrite(LED_AMARELO, LOW);
    else if (opcao == 'R')
      digitalWrite(LED_VERMELHO, HIGH);
    else if (opcao == 'r')
      digitalWrite(LED_VERMELHO, LOW);
    else
      Serial.println("Opcao Invalida.");
  }
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Comandos de Seleção

- Seleção de múltipla escolha
  - Na seleção de múltipla escolha é possível comparar vários valores.
  - No Arduino o comando de seleção de múltipla escolha é:

```
switch (valor) {
    case x: cmd1;
            break;
    case y: cmd2;
            break;
    default: cmd3;
}
```

  - onde:
    - *valor* – é um dado a ser avaliado. É representado por uma variável de memória.
    - cmd_x – comando a ser executado.
    - *case*– indica a opção a ser executada.
    - *default* – comando padrão que deverá ser executado se nenhuma outra escolha (*case*) tiver sido selecionada.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Comandos de Seleção

- Seleção de múltipla escolha
  - Acendendo e apagando leds pelo monitor serial

selecao_multipla_escolha

```
#define LED_VERDE    2
#define LED_AMARELO  3
#define LED_VERMELHO 4

char opcao;

void setup()
{
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_AMARELO, OUTPUT);
  pinMode(LED_VERMELHO, OUTPUT);
  Serial.begin(9600);
}
```

```
void loop()
{
  if (Serial.available()) {
    opcao = Serial.read();

    switch (opcao) {
      case 'G': digitalWrite(LED_VERDE, HIGH);
                break;
      case 'g': digitalWrite(LED_VERDE, LOW);
                break;
      case 'Y': digitalWrite(LED_AMARELO, HIGH);
                break;
      case 'y': digitalWrite(LED_AMARELO, LOW);
                break;
      case 'R': digitalWrite(LED_VERMELHO, HIGH);
                break;
      case 'r': digitalWrite(LED_VERMELHO, LOW);
                break;
      default:  Serial.println("Opcao Invalida.");
    }
  }
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

- Verificando o estado de um botão
  - Para verificar se um botão está pressionado ou não basta conectá-lo a uma porta digital.
  - Para que um circuito com botão funcione adequadamente é necessário o uso de resistores *pull-down* ou *pull-up*.
  - Os resistores *pull-down* e *pull-up* garantem que os níveis lógicos estarão próximos às tensões esperadas.

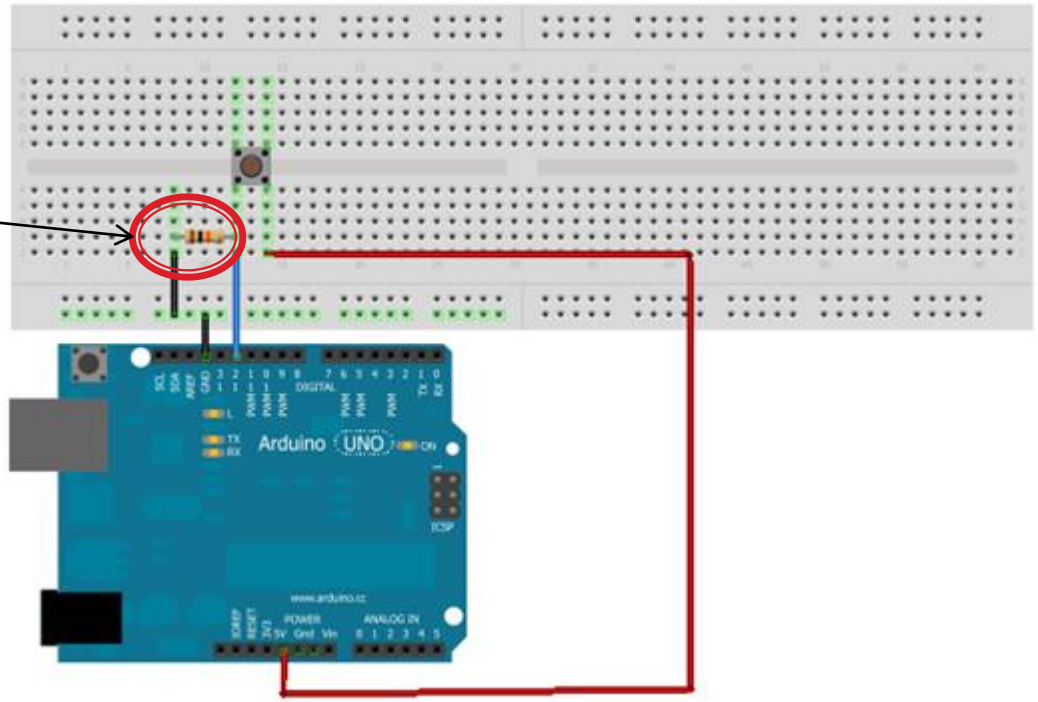
UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

- Leitura de um botão com resistor *pull-down*
  - Ligação na protoboard

Resistor pull-down 10kΩ



UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

- Leitura de um botão com resistor *pull-down*
  - Programa

pulldown

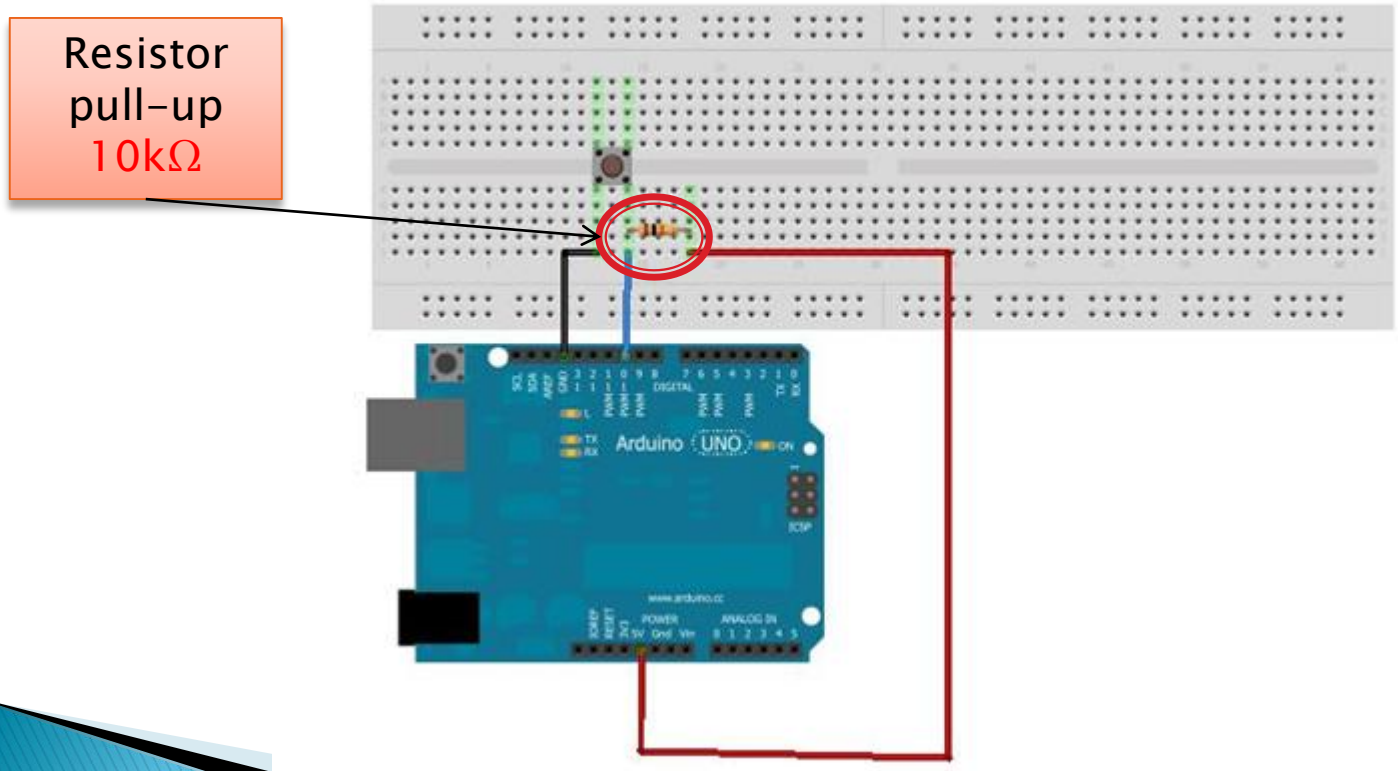
```
#define BOTAO 12

void setup()
{
  pinMode(BOTAO, INPUT);
  Serial.begin(9600);
}

void loop()
{
  if (digitalRead(BOTAO) == HIGH)
    Serial.println("Botao pressionado!");
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

- Leitura de um botão com resistor *pull-up*
  - Ligação na protoboard



UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LARM

Entrada Digital de Dados

- Leitura de um botão com resistor *pull-up*
  - Programa

pullup

```
#define BOTAO 10

void setup()
{
  pinMode(BOTAO, INPUT);
  Serial.begin(9600);
}

void loop()
{
  if (digitalRead(BOTAO) == LOW)
    Serial.println("Botao pressionado!");
}
```

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

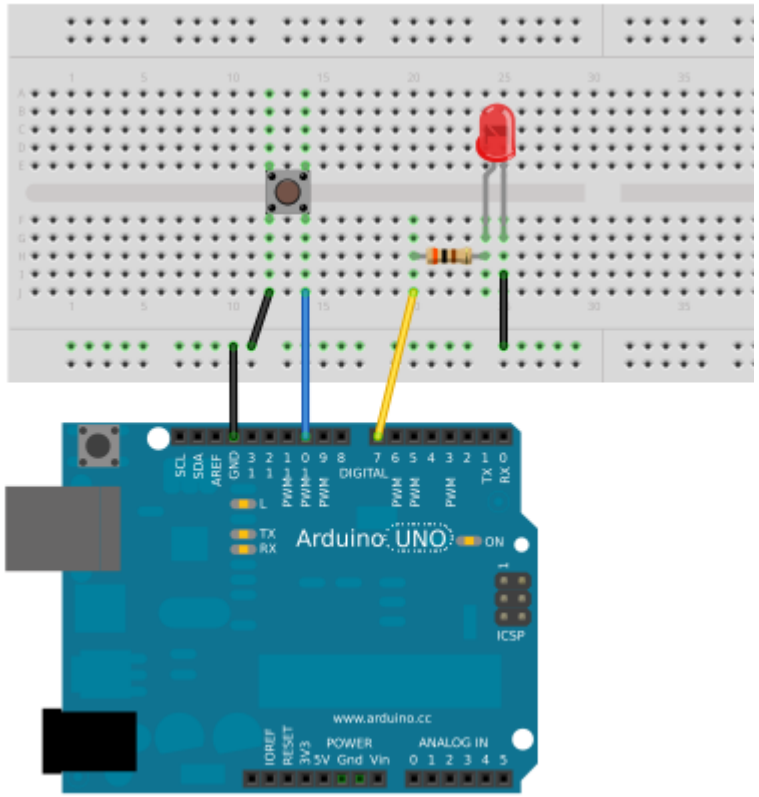
- **Nota**
  - O Arduino possui resistores *pull-up* nas portas digitais.
  - Para ativar os resistores *pull-up* de uma porta digital basta defini-la como entrada e colocá-la em nível alto (HIGH).
    - pinMode(*pin, INPUT*)
    - digitalWrite(*pin, HIGH*)
  - Para desativar os resistores *pull-up* de uma por digital basta colocá-la em nível baixo.
    - digitalWrite(*pin, LOW*)

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

- Ativando o *pull-up* de uma porta digital
  - Ligação na protoboard



UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

- Ativando o *pull-up* de uma porta digital
  - Programa

pullup_arduino

```
#define BOTAO 10
#define LED    7

void setup()
{
  pinMode(LED, OUTPUT);
  pinMode(BOTAO, INPUT);
  // ativa o resistor pull-up
  digitalWrite(BOTAO, HIGH);
}

void loop()
{
  if (digitalRead(BOTAO) == LOW)
    digitalWrite(LED, HIGH);
  else
    digitalWrite(LED, LOW);
}
```

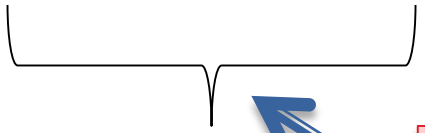
UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Entrada Digital de Dados

- **Nota**
  - O Arduino possui uma constante chamada *INPUT_PULLUP* que define que a porta será de entrada e o resistor *pull-up* da mesma será ativado.
  - Exemplo:

```
void setup()
{
  pinMode(10, INPUT_PULLUP);
}
```



Define a porta 10 como entrada de dados e ativa o resistor pull-up.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM

Exercícios

- Fazer um contador, com um botão pra incremento e um botão para decremento de uma variável. O valor da variável deve ser mostrado em tela.
- Criar um controle de LEDs (vermelho, amarelo e verde), onde o botão que for pressionado deve acender o LED correspondente a ele, e mostrar em tela qual LED está acesso.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

LARM