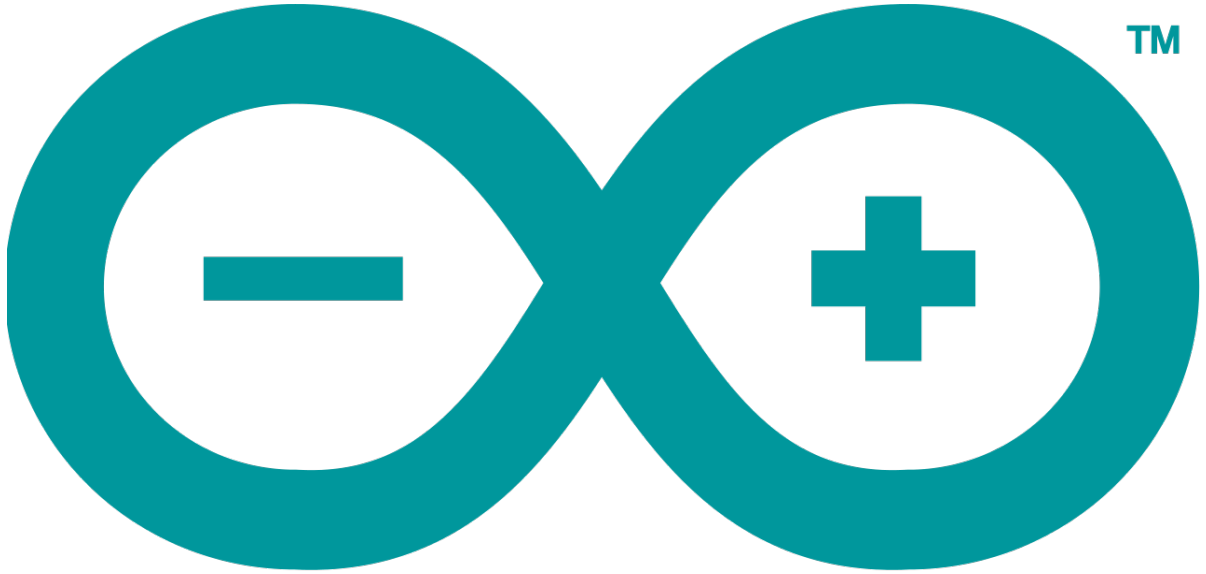



# INTRODUÇÃO A PLATAFORMA ARDUINO

TÁSSIO JOSÉ GONÇALVES GOMES
www.tassiogoncalves.com.br
tassiogoncalvesg@gmail.com

# APRESENTAÇÃO

## TÁSSIO JOSÉ GONÇALVES GOMES

Mestrando em Informática pela UFAL e Bacharel em Sistemas de Informação pela UFAL, com experiência na área de Tecnologia da Informação (TI), atuou como consultor de T.I. em Paulo Afonso - BA, na empresa COINPE, tem o conhecimento e o domínio de diversos softwares e variados seguimentos tais como: Redes de Computadores, Designer Gráfico, Web Designer, Compiladores de Linguagens de Programação, Banco de Dados. Atualmente trabalha com Suporte ao Sistema ERP da TOTVS o RM, Professor Tutor EaD da UFAL no curso de Sistemas de Informação e Técnico em Laboratório de Informática do IFBA - Câmpus de Paulo Afonso.

# CONTEÚDO

O Projeto Arduino

Breve história

Plataforma

IDE

Ciclo de Desenvolvimento

Funções Bases

Programar no Arduino

Shields

Exercícios Práticos

# O PROJETO ARDUÍNO

Arduíno é uma plataforma de prototipagem eletrônica de hardware livre, projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão, a qual tem origem em *Wiring*, e é essencialmente C/C++.

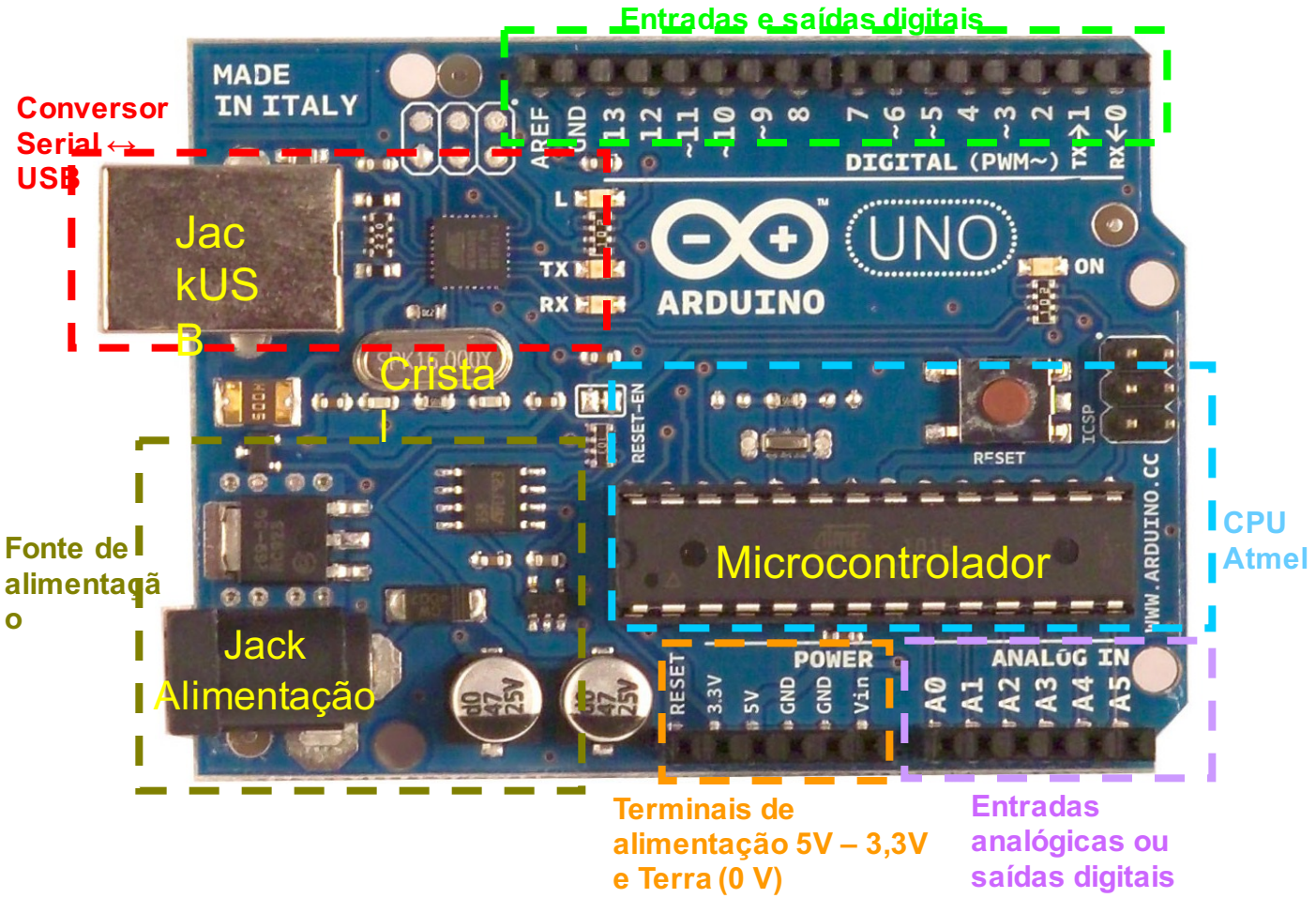
O objetivo do projeto é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de se usar por artistas e amadores. Principalmente para aqueles que não teriam alcance aos controladores mais sofisticados e de ferramentas mais complicadas.

# BREVE HISTÓRIA

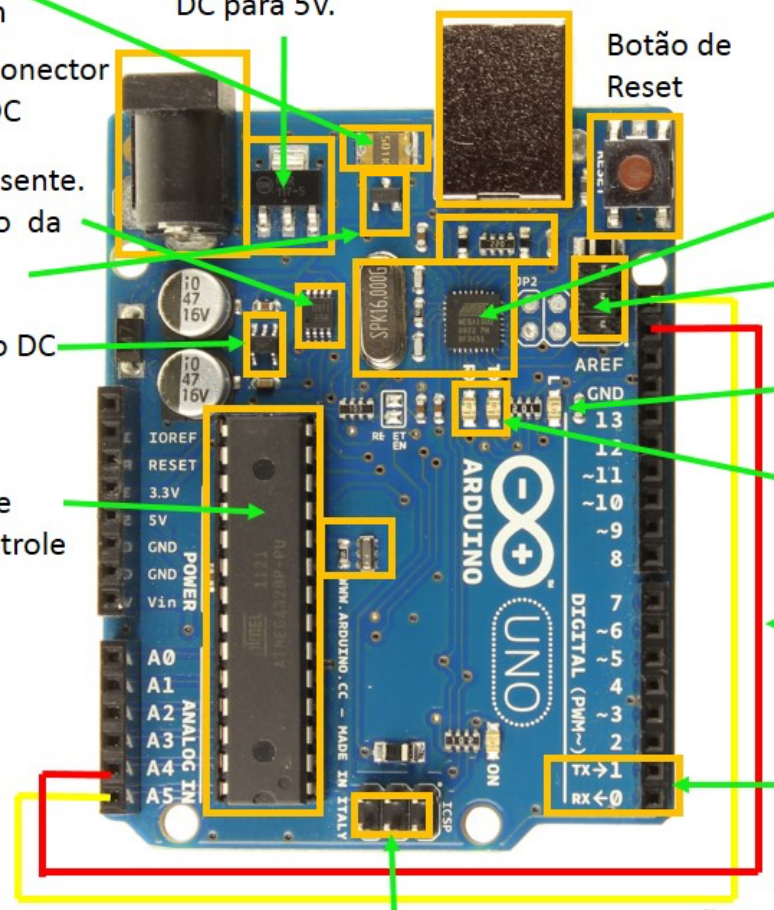
O projeto iniciou-se na cidade de Ivrea, Itália, em 2005, com o intuito de interagir em projetos escolares de forma a ter um orçamento menor que outros sistemas de prototipagem disponíveis naquela época.

Atualmente, seu hardware é feito através de um microcontrolador Atmel AVR, sendo que este não é um requisito formal e pode ser estendido se tanto ele quanto a ferramenta alternativa suportarem a linguagem Arduíno e forem aceitas por seu projeto.

# PLATAFORMA - HARDWARE

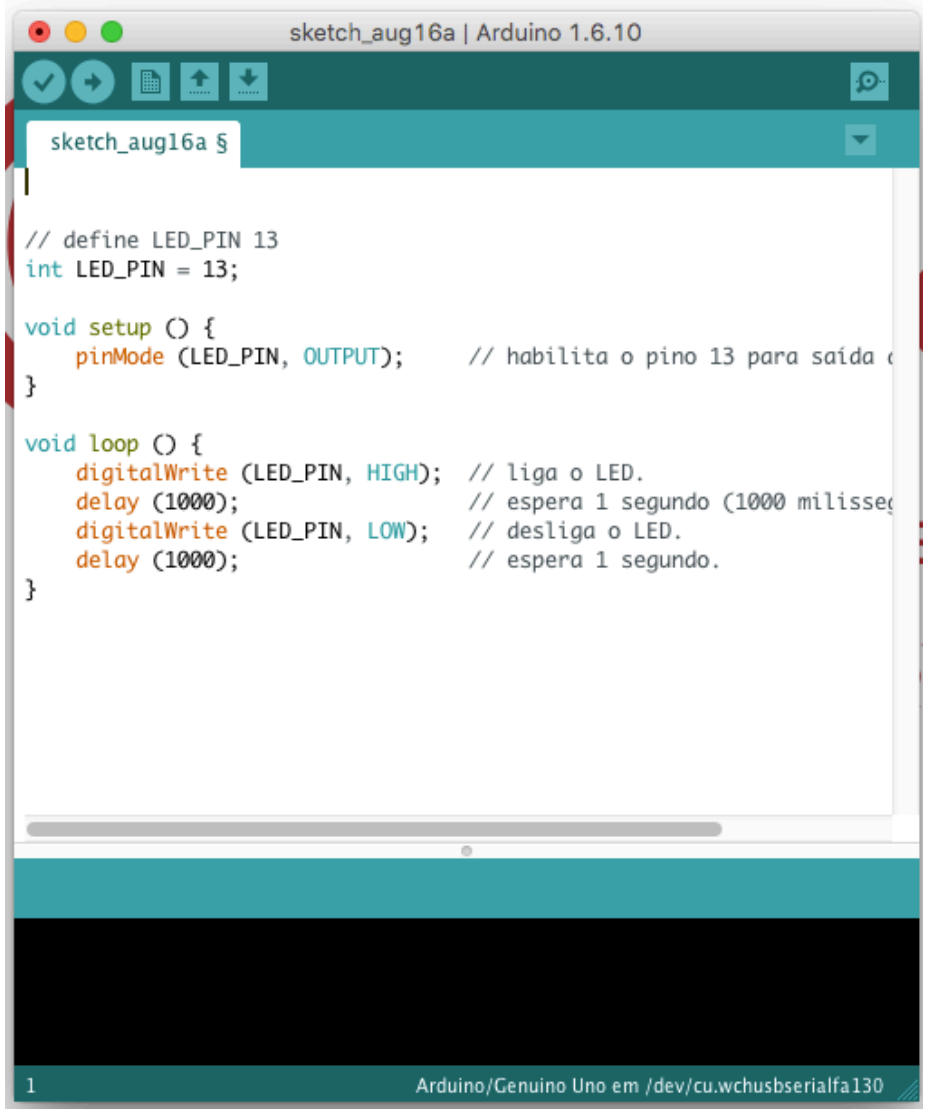


# PLATAFORMA - HARDWARE



# PLATAFORMA - SOFTWARE

O Arduino IDE é uma aplicação multiplataforma escrita em Java derivada dos projetos *Processing* e *Wiring*. É esquematizado para introduzir a programação a artistas e a pessoas não familiarizadas com o desenvolvimento de software. Inclui um editor de código com recursos de realce de sintaxe, parênteses correspondentes e identação automática, sendo capaz de compilar e carregar programas para a placa com um único clique.



# SOFTWARE

**1° passo:** consiste em efetuar o *download* do respectivo *software* de desenvolvimento, o Arduino IDE mais recente, através do site oficial *Arduino*.

**2° passo:** consiste em descompactar o ficheiro ".ZIP" para uma pasta à sua escolha.

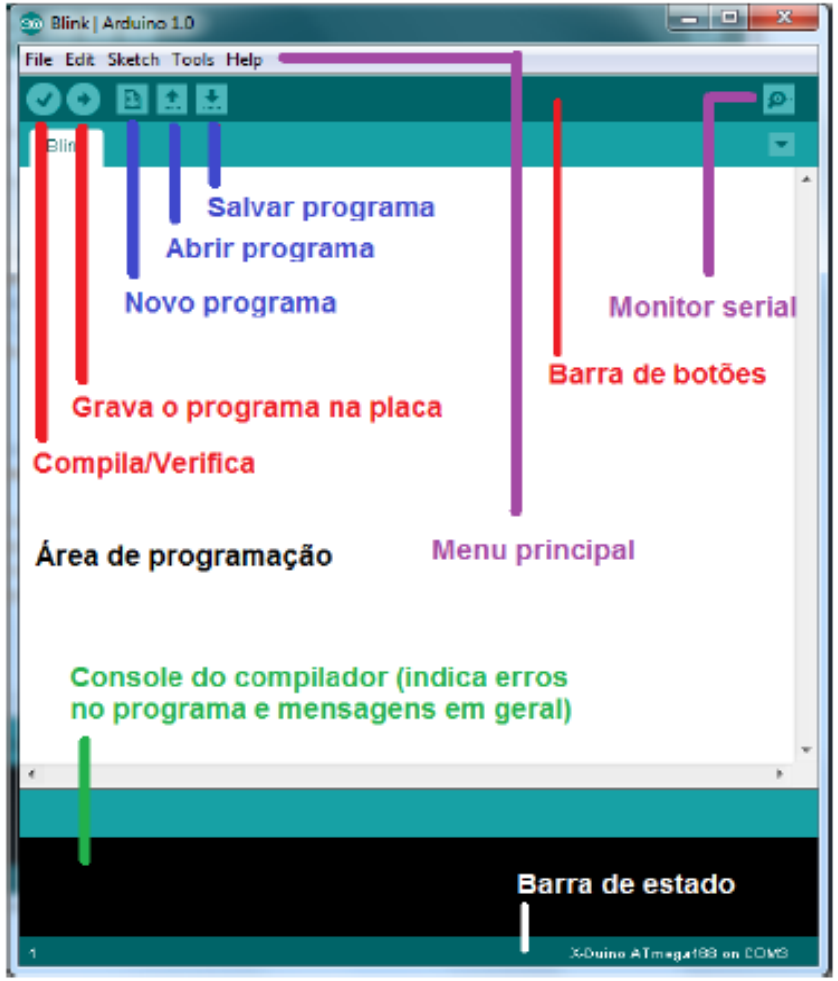
**3° passo:** consiste em ligar a placa Arduino ao computador através do cabo USB e instalar os *drivers*, para permitir uma conversão de USB para série.

**4° passo:** consiste em configurar a porta série a ser utilizada e qual o tipo de modelo *Arduino*, que nos encontramos a utilizar. Para tal, necessitamos abrir o *Software* de desenvolvimento e escolher na barra de separadores a opção "Tools" (ferramentas).

**5° passo:** a utilização do *Software,* consiste em elaborar o seu *Sketch (programa),* compilar e, caso não tenha erros, fazer o *upload* para a placa *Arduino*.

**“Integrated Development Environment”**
(Ambiente de desenvolvimento integrado)

# ARDUINO IDE

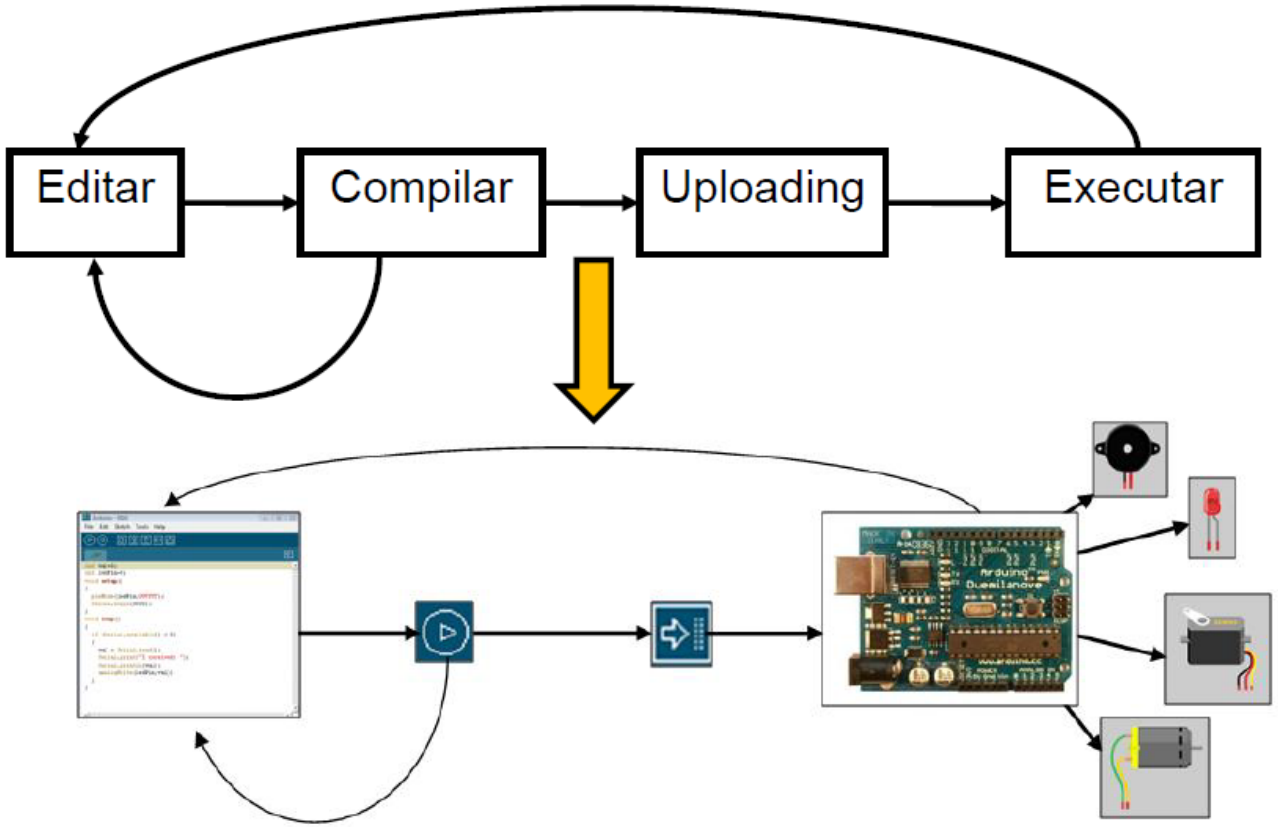



O monitor serial é utilizado para comunicação entre o Arduino e o computador (PC).

*As principais funcionalidades do IDE do Arduino são:*

- *Escrever o código do programa*
- *Salvar o código do programa*
- *Compilar um programa*
- *Transportar o código compilado para a placa do Arduino*

# CICLO DE DESENVOLVIMENTO



# FUNÇÕES BASE

**void setup()** - Esta função apenas é executada uma vez e é normalmente utilizada para executar a inicialização de variáveis, a inicialização da utilização bibliotecas, a definição dos pinos (como *input* ou *output*), o início do uso de comunicação série, entre outros. Esta função apenas volta a ser executada novamente ao ser efetuado o *reset* ou quando se desligar e volta a ligar a placa de desenvolvimento *Arduino*.

**void loop() -** Esta função faz um "**loop**" sucessivo (como o próprio nome indica), ou seja, todos os comandos existentes no interior desta função são sucessivamente repetidos, o que pode permitir a leitura sucessiva de portas, a leitura sucessiva de parâmetros provenientes de sensores externos e atuar de acordo com as condições estabelecidas.

```
Declaração de variáveis globais;
void setup(){
Instrução 1;
Instrução 2;
(....)
}
void loop(){
Instrução 6;
Instrução 9;
Função1();
(....)
}
```

Conjunto de instruções apenas executado uma vez, na inicialização do programa a executar

Conjunto de instruções que é executado em "loop"

# PROGRAMAR NO ARDUINO

## Comentários

Muitas vezes é importante comentar alguma parte do código do programa.

Existem duas maneiras de adicionar comentários a um programa em Arduino.

A primeira é usando //, como no exemplo abaixo:

```
// Este é um comentário de linha
```

A segunda é usando /* */, como no exemplo abaixo:

```
/* Este é um comentário de bloco. Permite acrescentar comentários com mais de uma linha */
```

*Nota: Quando o programa é compilado os comentários são automaticamente suprimidos do arquivo executável, aquele que será gravado na placa do Arduino.*

# PROGRAMAR NO ARDUINO

## Constantes

No Arduino existem algumas constantes previamente definidas e são consideradas palavras reservadas.

As constantes definidas são:

**true ou 1** – indica valor lógico verdadeiro

**false ou 0** – indica valor lógico falso

**HIGH ou 1** – indica que uma porta está ativada, ou seja, está em 5V.

**LOW ou 0** – indica que uma porta está desativada, ou seja, está em 0V.

**INPUT** – indica que uma porta será de entrada de dados.

**OUTPUT** – indica que uma porta será de saída de dados.

# ENTRADAS ANALÓGICAS E DIGITAIS

**Portas digitais e analógicas**

As portas servem para comunicação entre o Arduino e dispositivos externos, por exemplo: ler um botão, acender um led ou uma lâmpada.

Conforme já mencionado, o Arduino UNO, possui 14 portas digitais e 6 portas analógicas (que também podem ser utilizadas como portas digitais).


**Portas digitais**

As portas digitais trabalham com valores bem definidos, ou seja, no caso do Arduino esses valores são 0V e 5V.

0V indica a ausência de um sinal e 5V indica a presença de um sinal.

Para escrever numa porta digital basta utilizar a função **digitalWrite(pin, estado).**

Para ler um valor numa porta digital basta utilizar a função **digitalRead(pin).**

# ENTRADAS ANALÓGICA

**Portas Analógicas**

São utilizadas para entrada de dados. O Arduino UNO possui 6 (seis) portas analógicas.

Estas, no Arduino UNO, são identificadas como A0, A1, A2, A3, A4 e A5. Também podem ser identificadas por 14 (A0), 15 (A1), 16 (A2), 17 (A3), 18 (A4) e 19 (A5).

Por padrão todas as portas analógicas são definidas como entradas de dados, desta forma não é necessário fazer esta definição na função setup().

Os valores lidos variam de 0V a 5V.

Para ler uma valor em uma porta analógica basta utilizar a função **analogRead(pin).**

Entradas Analógicas

# PROGRAMAR NO ARDUINO

**Para definir uma porta como entrada ou saída** é necessário explicitar essa situação no programa.

A função pinMode(pin, estado) é utilizada para definir se a porta será de entrada ou saída de dados.

Exemplo:

Define que a porta 13 será de saída

**pinMode(13, OUTPUT)**

Define que a porta 7 será de entrada

**pinMode(7, INPUT)**

# PROGRAMA EXPLICADO

```
/*
---------------------------------
1º Exercício
---------------------------------
Ligar e desligar um LED por um segundo
*/
//---------------------------------
//Função principal
//---------------------------------
void setup() // Executa-se quando o arduino é ligado
{
pinMode(13,OUTPUT); // Inicializa o pin 13 como uma saída
}
//---------------------------------
//Função repetitiva
//---------------------------------
void loop() // Esta função executa-se o instante todo
// quando está ligado o Arduino
{
digitalWrite(13,HIGH); // Liga o LED
delay(1000); // Temporiza um segundo (1s = 1000ms)
digitalWrite(13,LOW); // Desliga o LED
delay(1000); // Temporiza um segundo (1s = 1000ms)
}
```

# PROGRAMAR NO ARDUINO

Numa linguagem de programação existem vários operadores que permitem operações do tipo: Aritmética, Relacional, Lógica e Composta.

- Operadores aritméticos

| Símbolo | Função |
|---|---|
| + | Adição |
| - | Subtração |
| * | Multiplicação |
| / | Divisão |

# PROGRAMAR NO ARDUINO

- Operadores relacionais

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

# PROGRAMAR NO ARDUINO

- Operadores lógicos

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

# PROGRAMAR NO ARDUINO

- Operadores compostos

| Símbolo | Função |
|---|---|
| ++ | Incremento |
| -- | Decremento |

# UTILIZANDO SERIAL NO ARDUINO

A plataforma Arduino possui em sua biblioteca uma variedade de funções para manipulação de dados através de comunicação serial. Essas funções auxiliam o desenvolvedor em tarefas mais complexas de envio e recebimento de dados.

# FUNÇÕES SERIAL NO ARDUINO

Serial.begin(speed)

Serial.read()

Serial.print()

Serial.println()

# SHIELDS

É possível agregar novas funcionalidades a uma placa do Arduino.

As extensões das placas do Arduino são chamadas de shields.

Existem shields para as mais diversas funcionalidades, por exemplo:

- Comunicação ethernet
- Comunicação wifi
- Comunicação bluethooth
- Ponte H
- Banco de relês

Ethernet Shield

# EXERCÍCIOS PRÁTICOS

1. Criar um programa em Arduino para piscar o LED do pino 13 a cada 1 segundo.
2. Criar um programa em Arduino que pisque os LEDs dos pinos 13 e 12 intercalados.
3. Incremente o Exercício anterior e faça com que os LEDs pisquem duas vezes antes de inverter.
4. Utilizando a porta serial (Serial.read() e Serial.print()), realize um programa que acenda os LEDs quando digitado uma letra.
5. Modifique o exercício anterior e faça com que os LEDs pisquem quando for digitado uma letra e para cada LED e uma letra “l”, pisquem os dois LEDs 10 vezes.

# FONTES DE INFORMAÇÃO

Site oficial Arduino
http://www.arduino.cc

Para comandos, consultar o guia de referências em

http://arduino.cc/en/Reference/HomePage

Biblioteca MsTimer2
http://arduino.cc/playground/Main/MsTimer2

https://pt.wikipedia.org/wiki/Arduino

http://www.prof2000.pt/users/lpa

Simulador de Arduino: Virtual Breadboard
http://www.virtualbreadboard.com/Main.aspx?TAB=Downloads

http://www.embarcados.com.br/arduino-comunicacao-serial/