

## Aula 2

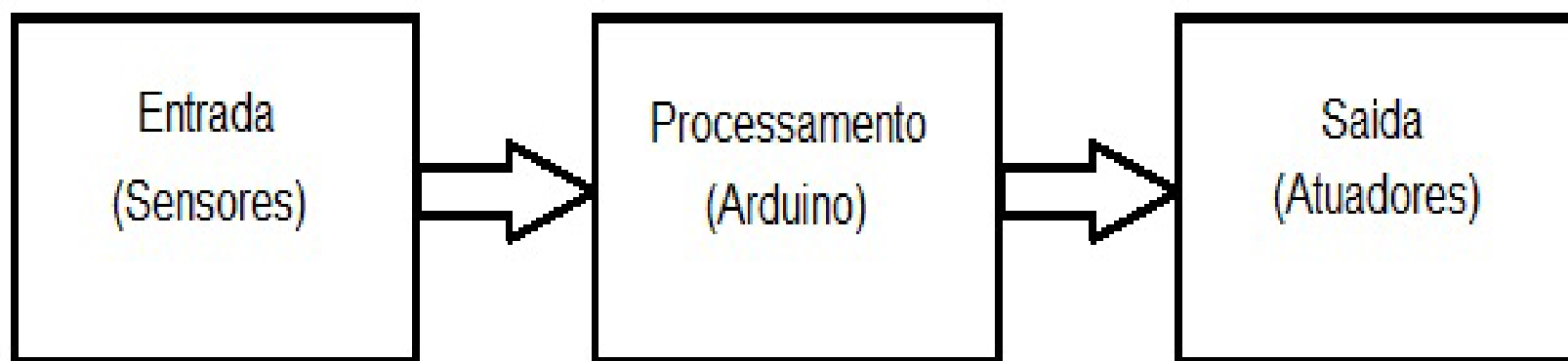
Palestrante: Adauto Silva

Email: [adauttosilva@gmail.com](mailto:adauttosilva@gmail.com)

Blog: [equipetechhunters.blogspot.com.br](http://equipetechhunters.blogspot.com.br)

Lattes:

# Sensores e Atuadores





**UNIFACS**

UNIVERSIDADE SALVADOR

LAUREATE INTERNATIONAL UNIVERSITIES\*

# COMPUTAÇÃO FÍSICA



**UNIFACS**

UNIVERSIDADE SALVADOR

LAUREATE INTERNATIONAL UNIVERSITIES\*

# HISTÓRICO

Local Ivrea, Itália

Ano 2005

Comunidades Digitais em 2006

Pris Arm Eletronicas

50.000 Placas

# HISTÓRICO

Gianluca Martino  
David Mellis.  
David Cuartielles  
Tom Igoe  
Massimo Banzi



<http://pt.wikipedia.org/wiki/Arduino>

# O QUE É ARDUINO?



<http://www.arduino.org.br/2011/01/uno/>,



**UNIFACS**

UNIVERSIDADE SALVADOR

LAUREATE INTERNATIONAL UNIVERSITIES\*

# O QUE É ARDUINO?

É uma placa Open-source

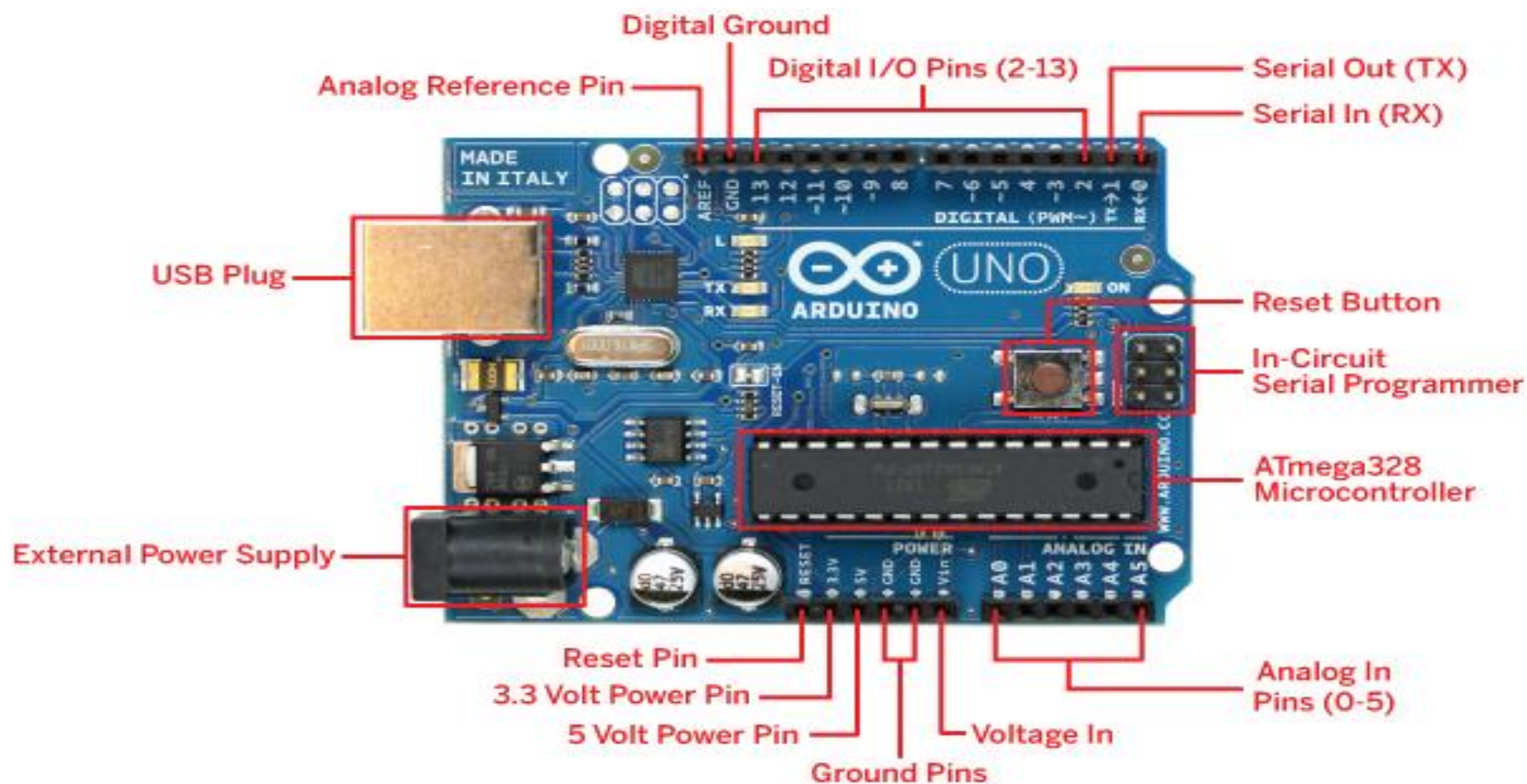
Microcontrolador Atmel

Suporte a entradas e saídas

Linguagem de programação Wiring C/C++



# ARDUINO UNO



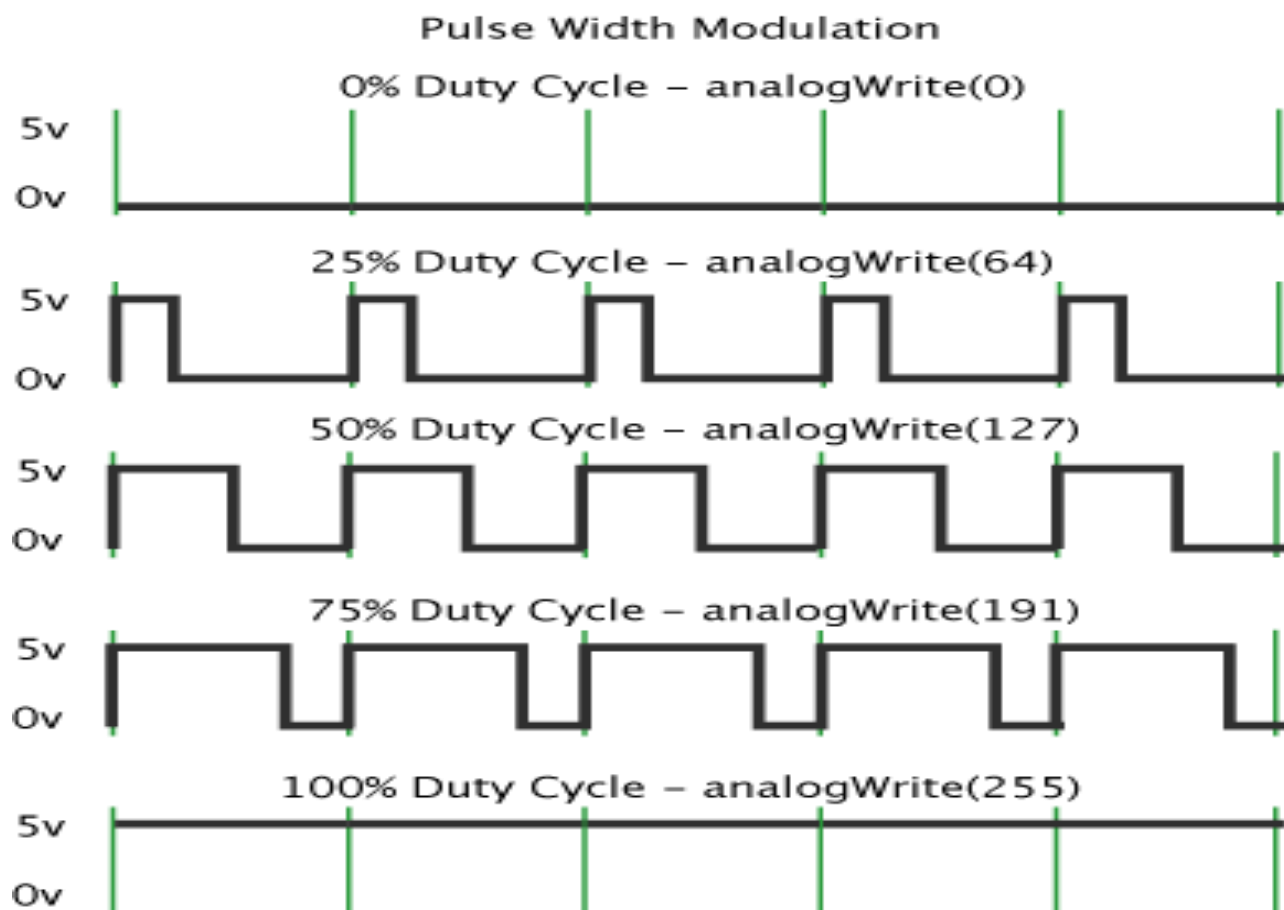
<http://www.arduino.com.br/2011/01/uno/>,



# CARACTERÍSTICAS

Tamanho:	<b>5,3cm x 6,8cm x 1,0cm</b>
Microcontrolador:	<b>ATmega328</b>
Tensão de operação:	<b>5V</b>
Tensão de entrada (recomendada):	<b>7-12V</b>
Tensão de entrada (limites):	<b>6-20V</b>
Pinos de entrada/saída (I/O) digitais:	<b>14 (dos quais 6 podem ser saídas PWM)</b>
Pinos de entrada analógicas:	<b>6</b>
Corrente DC por pino I/O:	<b>40mA</b>
Corrente DC para pino de 3,3V:	<b>50mA</b>
Memória Flash:	<b>32KB (dos quais, 0,5KB são usados pelo bootloader)</b>
SRAM:	<b>2KB</b>
EEPROM:	<b>1KB</b>
Velocidade de Clock:	<b>16MHz</b>

# PWM



<http://www.arduino.cc/en/Tutorial/PWM>

# MODELOS DE ARDUINOS



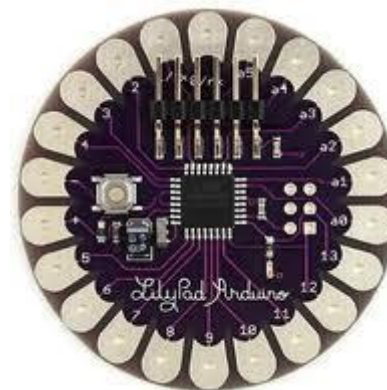
<http://www.arduino.cc/en/Tutorial/PWM>



<http://leomar.com.br>



<http://leomar.com.br>



<http://leomar.com.br>

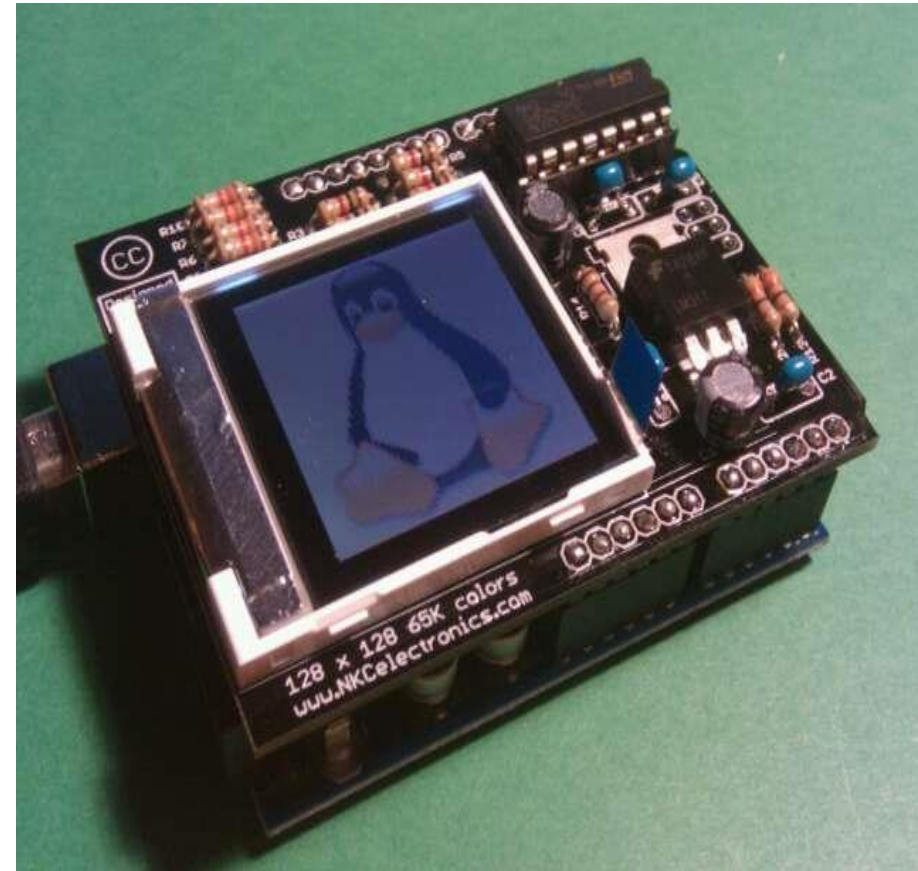
# MODELOS DE SHIELD



<http://www.robotshop.com/ProductInfo.aspx?pc=RB-Ard-05>



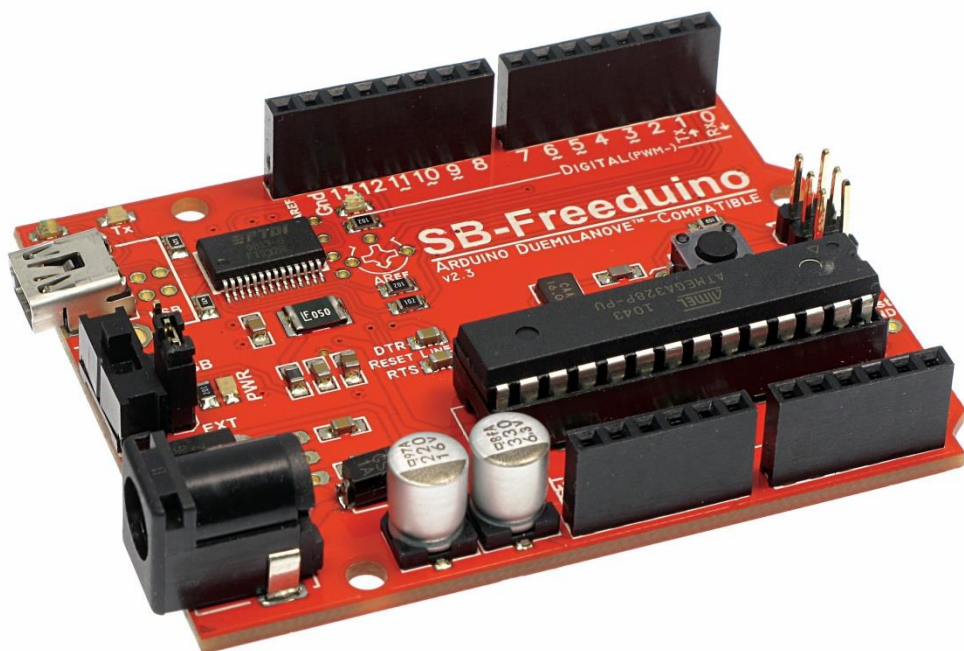
<http://nicegear.co.nz/arduino-shields/gsm-cellular-shield-with-sm5100b/>



<http://www.jarenhavell.com/Projects/projects/arduino/>



# MODELOS CLONES

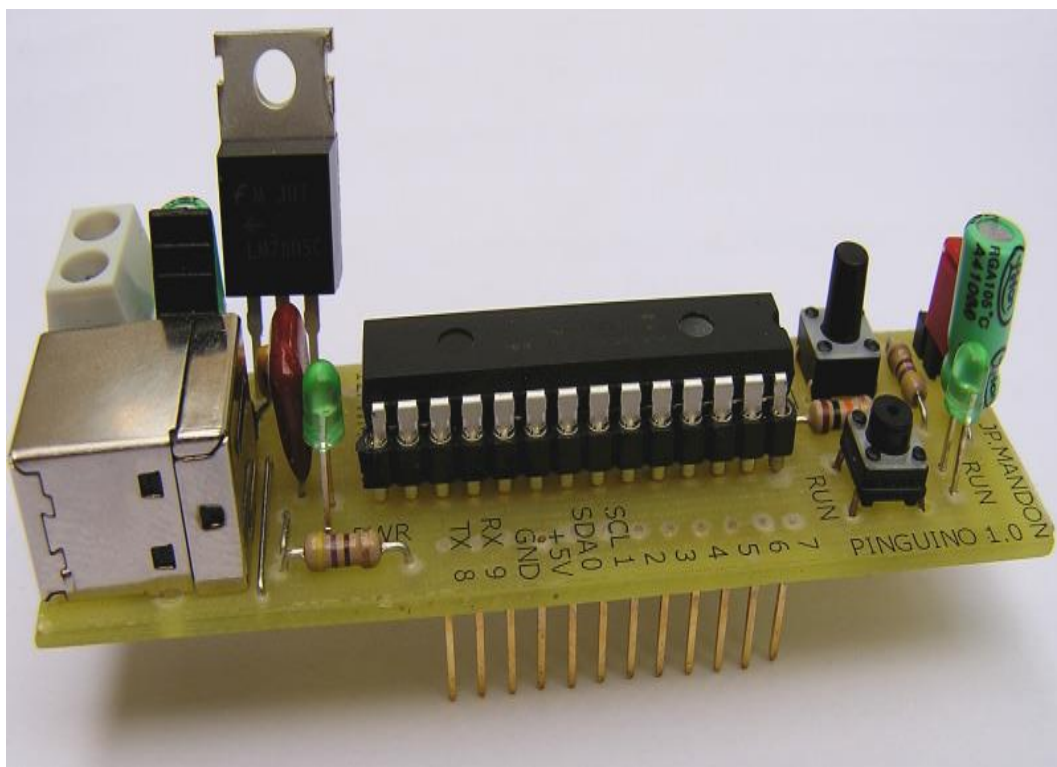


<http://www.solarbotics.com/products/28920/>

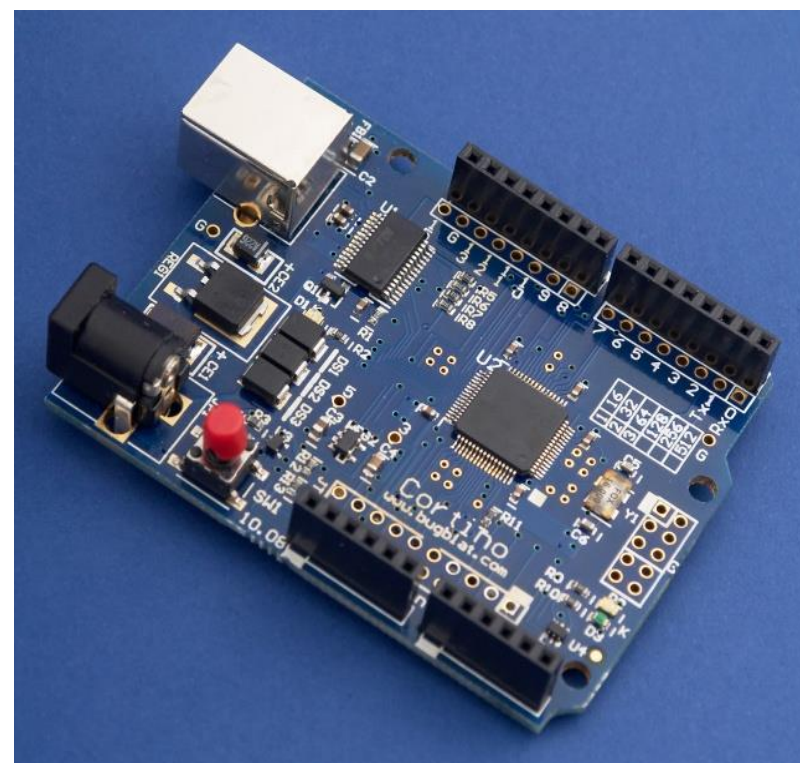


<http://brasuino.holoscopio.com/>

# MODELOS USANDO OUTRO MICROCONTROLADOR



<http://www.zeitounian.com.br/pinguino/> (PIC)



<http://www.bugblat.com/products/cor.html> (ARM M3)

# INTERFACE DE PROGRAMAÇÃO



<http://www.openframeworks.cc/>



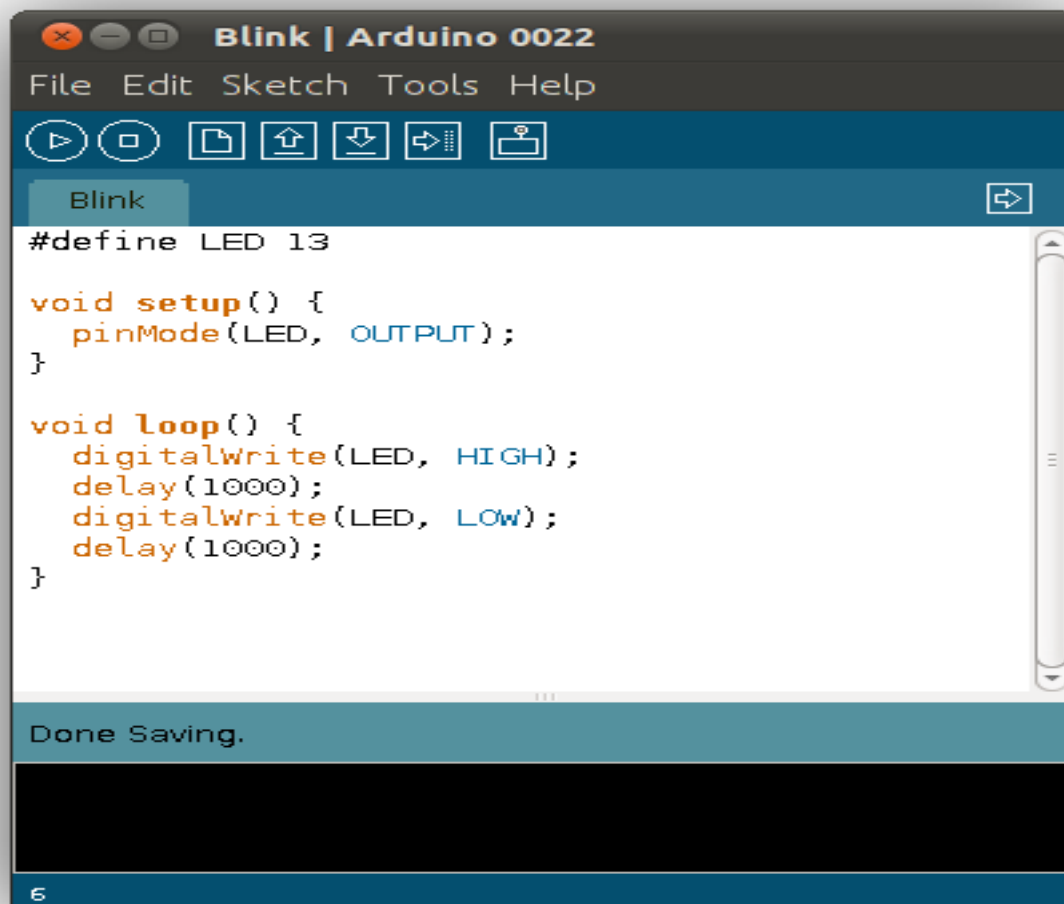
[http://pt.wikipedia.org/wiki/Processing\\_%28linguagem\\_de\\_programa%C3%A7%C3%A3o%29](http://pt.wikipedia.org/wiki/Processing_%28linguagem_de_programa%C3%A7%C3%A3o%29)



<http://pyserial.sourceforge.net/>



# IDE ARDUINO



```
#define LED 13

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```

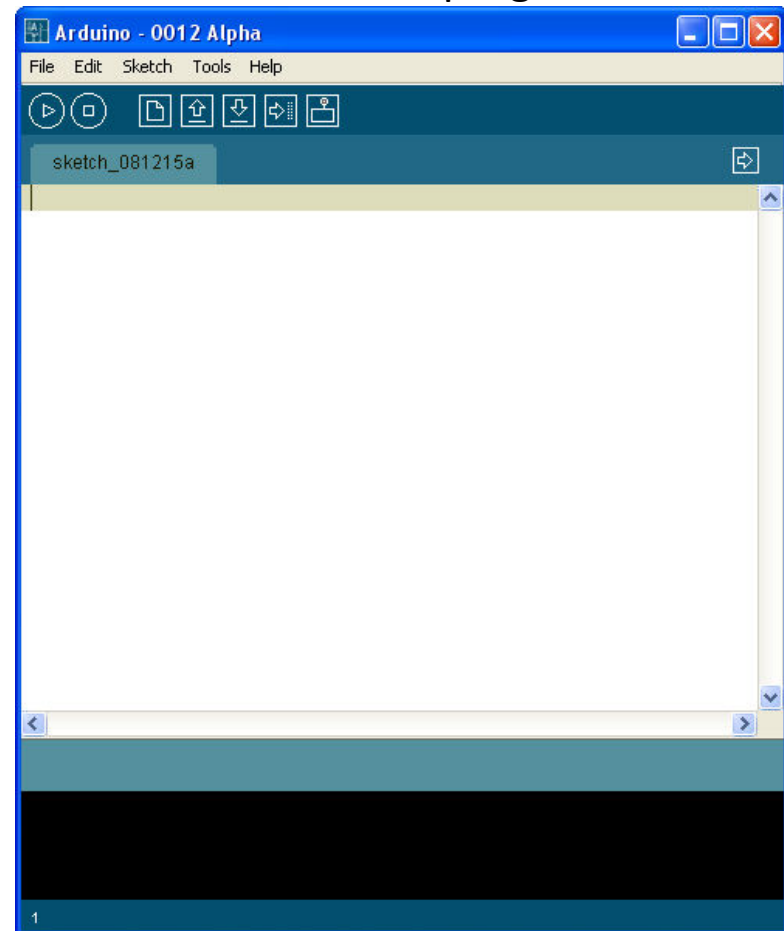
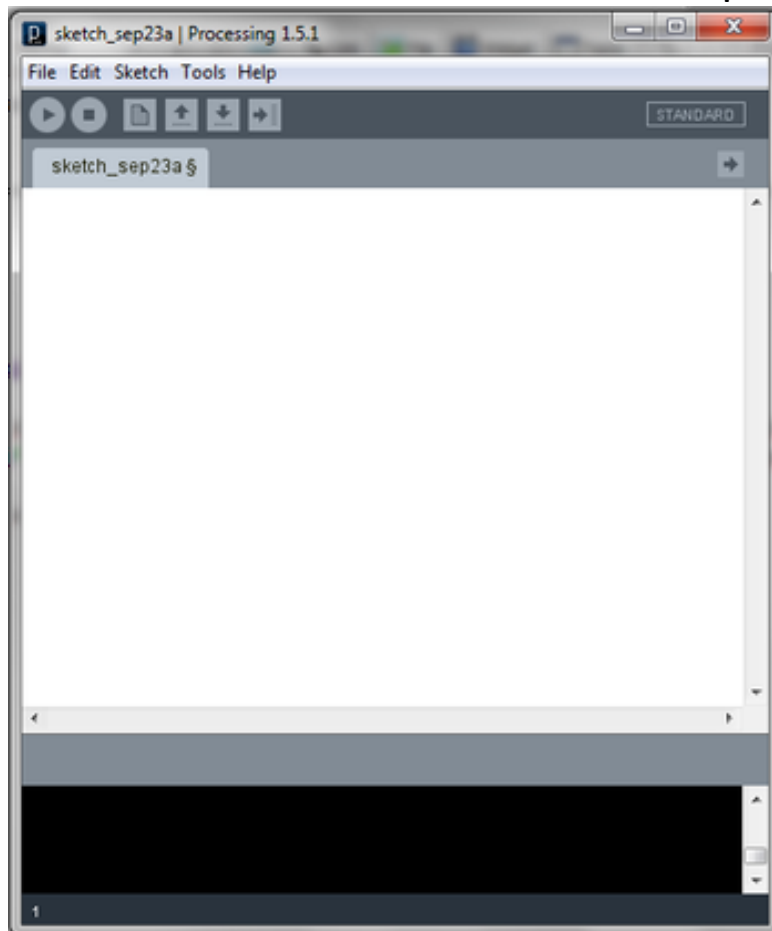
Done Saving.

Fonte Própria








É constituído pela Toolbar, Tab Menu e Menus (File, Edit, Sketch, Tools e Help).

O Tab Menu permite gerir documentos com mais do que um ficheiro, cada um aberto num tab independente. Esses ficheiros podem ser - **Ficheiros normais de código Arduino (sem extensão)** - **Ficheiros C (extensão .c), C++ (.cpp)**

**OBS:** Os Menus File, Edit e Help são semelhantes em todos os programas.



Conhecendo o menu ***Sketch***... Este contém os seguintes comandos:

<i>Verify/Compile</i> 	Verifica se o código tem erros
<i>Stop</i> 	Pára o serial monitor ou desactiva outros botões
<i>New</i> 	Cria um novo documento
<i>Open</i>	Abre uma lista dos documentos salvos e abre o que
	for seleccionado
<i>Save</i> 	Salva o documento
<i>Upload to I/O Board</i> 	Faz upload do código para a placa do <i>Arduino</i>
<i>Serial Monitor</i> 	Mostra a informação enviada pela placa do <i>Arduino</i>

# VARIÁVEIS E MODIFICADORES

Os nomes das variáveis apenas devem obedecer algumas regras: elas podem começar com letras ou sublinhado (\_) e não podem ter nome idêntico a alguma das palavras reservadas pelo programa ou de alguma biblioteca.

As variáveis ***devem ser declaradas antes de serem usadas***. Observe:

***tipo\_de\_variável*** *lista\_de\_variáveis*

Ex: `int ledPin`, *potenciometro*

# CLASSE DE VARIÁVEIS

## VARIÁVEIS LOCAIS

Quando uma variável é declarada dentro de uma função específica, ela é denominada variável local. Estas variáveis apenas existem enquanto o bloco onde está armazenada estiver sendo executado. A partir do momento em que o programa voltar à função principal, esta variável deixará de existir.

## VARIÁVEIS GLOBAIS

Uma variável global é aquela variável que é conhecida por todo o programa, ou seja, independente da função que estiver sendo executada ela será reconhecida e rodará normalmente.

## VARIÁVEIS ESTÁTICAS

Funcionam de forma parecida com as variáveis globais conservando o valor durante a execução de outras funções, porém, só são reconhecidas dentro da função onde é declarada.

## Algumas variáveis ...

**int** é a variável padrão do programa. Esta variável consegue memorizar dados de -32768 até 32767. Esta variável consegue memorizar números negativos através da propriedade matemática chamada **complemento de dois**.

*Exemplo: `int ledPin = 13; int nomeVariavel = valorVariavel;`*

**char** é um tipo de dado que dedica 1 byte de memória para armazenar o valor de um caractere.

*Exemplo: `palavra = 'Arduino';`*

**float**: Guarda um número real com certa precisão

*Exemplo: `12,8; 11,756; 666,666 ...`*

## Conceito de Condição (If - Else)

Podemos dizer que as estruturas de controle de fluxo são a parte mais importante da programação em Arduino, pois toda a programação é executada em torno delas.

**IF** testa se certa condição foi alcançada, como por exemplo, se uma entrada analógica obteve um número específico. Se o resultado da condição for 0 (falsa), ela não será executada, caso o resultado seja 1 (verdadeira), a função será executada. O formato desta estrutura de controle de fluxo é:

```
if (certaCondicao) {  
  
    // comandos... }
```



# IF... ELSE

Usar **if/else** permite um controle maior sobre a estrutura, sendo que poderá ser realizado vários testes sem sair de uma única função. Podemos pensar no **else** como sendo um complemento ao comando **if**.

```
if (certaCondicao) {  
  
    // comando A... }  
  
else {  
  
    // comando B... }
```

## OBSERVAÇÃO!

*Quando você estiver trabalhando com várias condições, onde não se restringe a apenas duas hipóteses, dentro da estrutura de controle **else**, você pode colocar mais funções **if/else** e assim por diante. Veja:*

```
if (Condicao1) {  
  
    // comando A... }  
  
else if (Condicao2) {  
  
    // comando B... }  
  
else {  
  
    // comando C }
```

# ESTRUTURA DE REPETIÇÃO (For, While)

## FOR

O estrutura for é uma das estruturas que se trabalha com loops de repetição. Esta estrutura normalmente é usada para repetir um bloco de informações definidas na função. Enquanto a condição for verdadeira, as informações contidas na função serão executadas. Sua forma geral é:

```
for (inicialização; condição; incremento) {  
    //instrução (ou instruções);  
}
```

## Vamos entender os blocos desta estrutura:

**INICIALIZAÇÃO:** a inicialização é o que ocorre primeiro e apenas uma vez.

**CONDIÇÃO:** o bloco **FOR** trabalha com loops de repetição, cada vez que a função for repetida, a condição será testada, se ela for verdadeira (1), executará o que se encontra dentro das chaves.

**INCREMENTO:** se a condição for verdadeira, o incremento será executado, caso contrário, se a condição for falsa, o loop é terminado.

## WHILE

A estrutura WHILE funciona como um loop de repetição. Enquanto a condição contida dentro dos parênteses ***permanecer*** verdadeira, a estrutura será executada. A partir do momento que a condição for falsa, o programa seguirá normalmente. Sua forma geral é:

```
while (condição) {  
    // instrução 1...  
    // instrução 2...  
}
```



# Bloco Principal

```
void setup(){  
  
}  
  
void loop(){  
  
}
```

# Instruções básicas

- **pinMode(pino de 0 a 13, INPUT ou OUTPUT);**

Se INPUT é entrada de dados. Se OUTPUT é saída de dados.

- **digitalRead(pino de 0 a 13);**

Faz a leitura do pino escolhido se está em nível lógico alto (HIGH ou 1) ou nível lógico baixo (LOW ou 0).

- **digitalWrite(pino, HIGH/LOW);**

Escreve no pino escolhido um nível lógico alto (HIGH ou 1) ou nível lógico baixo (LOW ou 0).

- **analogRead(pino);**

Lê uma entrada analógica (variável) no pino escolhido.

- **delay(ms);**

Determina o tempo que o Arduino irá esperar até executar a próxima linha de código.



# Leitura e escrita digital

## Digital

- `pinMode(num_do_pino, INPUT|OUTPUT);` Define o Pino como entrada ou saída
- `digitalWrite(num_do_pino, valor);` Envia um comando para o pino de saída.

**Obs:** Valor é LOW ou HIGH (0 ou 1, 0v ou 5v)

- `x = digitalRead(num_do_pino);` Faz leitura do pino de entrada.

# Leitura e escrita analógica

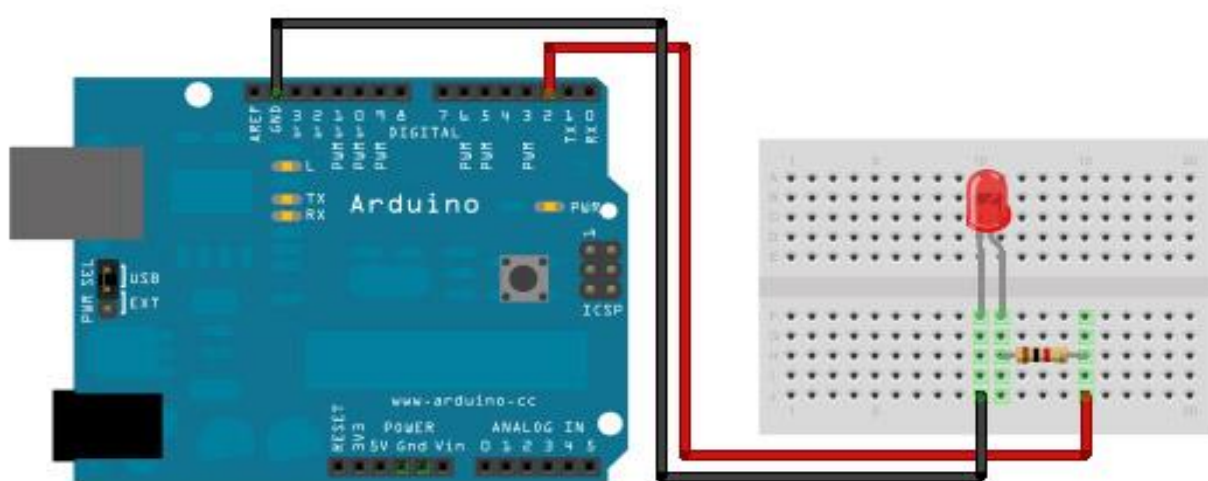
## Analógica

- `analogWrite(num_do_pino, valor);` Escreve um valor analógico no pino, valor ente 0 e 255.
- `y = analogRead(num_do_pino);` Ler valores associado ao pino
- `delay (milisegundos);` “Espera” um tempo...

# Liga led

```
void setup(){  
  pinMode(13,OUTPUT); // define que o pino 13 é saída  
  
}  
void loop(){  
  digitalWrite(13,HIGH); // coloco o pino 13 em nível lógico  
    alto  
}
```

# Liga led



<http://makebits.net/arduino-de-principiante-a-utilizador-avancado-aula-1/>



**UNIFACS**

UNIVERSIDADE SALVADOR

LAUREATE INTERNATIONAL UNIVERSITIES\*

# REFERÊNCIAS

<http://www.arduino.cc/>

<http://equipetechhunters.blogspot.com/>

<http://lusorobotica.com/>

<http://arduino-ce.blogspot.com/>