



**UNIVERSIDADE FEDERAL  
DE SANTA CATARINA**

# Arduino: Módulo Básico

# Apresentação

Material produzido para projeto Escola de Automação e Robótica, projeto de extensão pela Universidade Federal de Santa Catarina, campus Blumenau.

## Contatos

Brunno Vanelli

Guilherme Moser Manerichi

Alex Sandro Roschildt Pinto

[brunno.v@grad.ufsc.br](mailto:brunno.v@grad.ufsc.br)

[guilherme.manerichi@grad.ufsc.br](mailto:guilherme.manerichi@grad.ufsc.br)

[a.r.pinto@ufsc.br](mailto:a.r.pinto@ufsc.br)

Escola de Automação e Robótica

[facebook.com/earUFSC](https://facebook.com/earUFSC)



[github.com/bvanelli/eAR](https://github.com/bvanelli/eAR)



# Apresentação

1. O que é Arduino?
2. Portas Digitais
3. Portas Analógicas
4. Portas PWM
5. Comunicação Serial
6. Desvios Condicionais
7. Usando Bibliotecas

A thick yellow diagonal stripe runs from the top right corner towards the bottom left, separating the white background on the left from the solid yellow background on the right.

# 1.

Introdução

O que é Arduino?



“

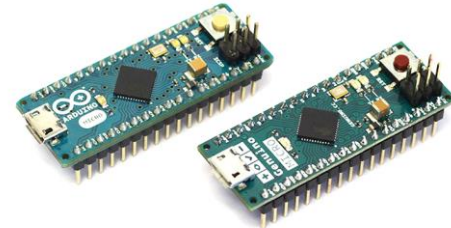
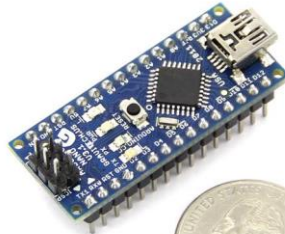
Arduino é uma plataforma de desenvolvimento open-source com *hardware* e *software* fáceis de utilizar.

[Arduino.cc](https://www.arduino.cc)

- ▶ Desenvolvido em 2005 na cidade de Ivrea, na Itália.
- ▶ O Arduino é uma plataforma voltada para **prototipagem e desenvolvimento** de circuitos eletrônicos.
- ▶ Placas baseadas em microcontroladores **Atmel AVR**, e ambiente de programação baseado em **C++**.
- ▶ Hardware e Software livres (open-source).

# Tipos de Arduino

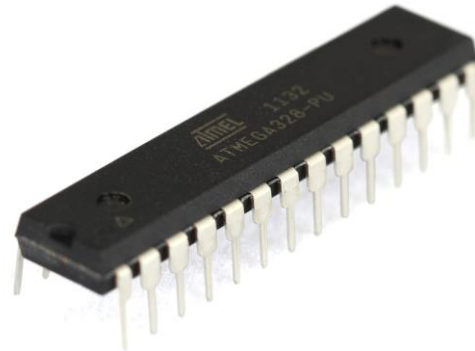
- ▶ Arduino Uno
- ▶ Arduino Leonardo
- ▶ Arduino Mega
- ▶ Arduino Nano
- ▶ Arduino Due
- ▶ Arduino Micro
- ▶ Arduino Yún



# Especificações Técnicas

## ARDUINO UNO

- ▶ Microcontrolador **ATMega328P**
- ▶ Tensão de operação: **5V**
- ▶ Alimentação Recomendada: **7V - 12V**
- ▶ **14** pinos digitais, **6** pinos analógicos
- ▶ Corrente máxima por porta: **20mA - 40mA**
- ▶ Velocidade do Clock: **16 MHz**
- ▶ Memória Interna: **32 KB**
  - SRAM: **2 KB**
  - EEPROM: **1 KB**

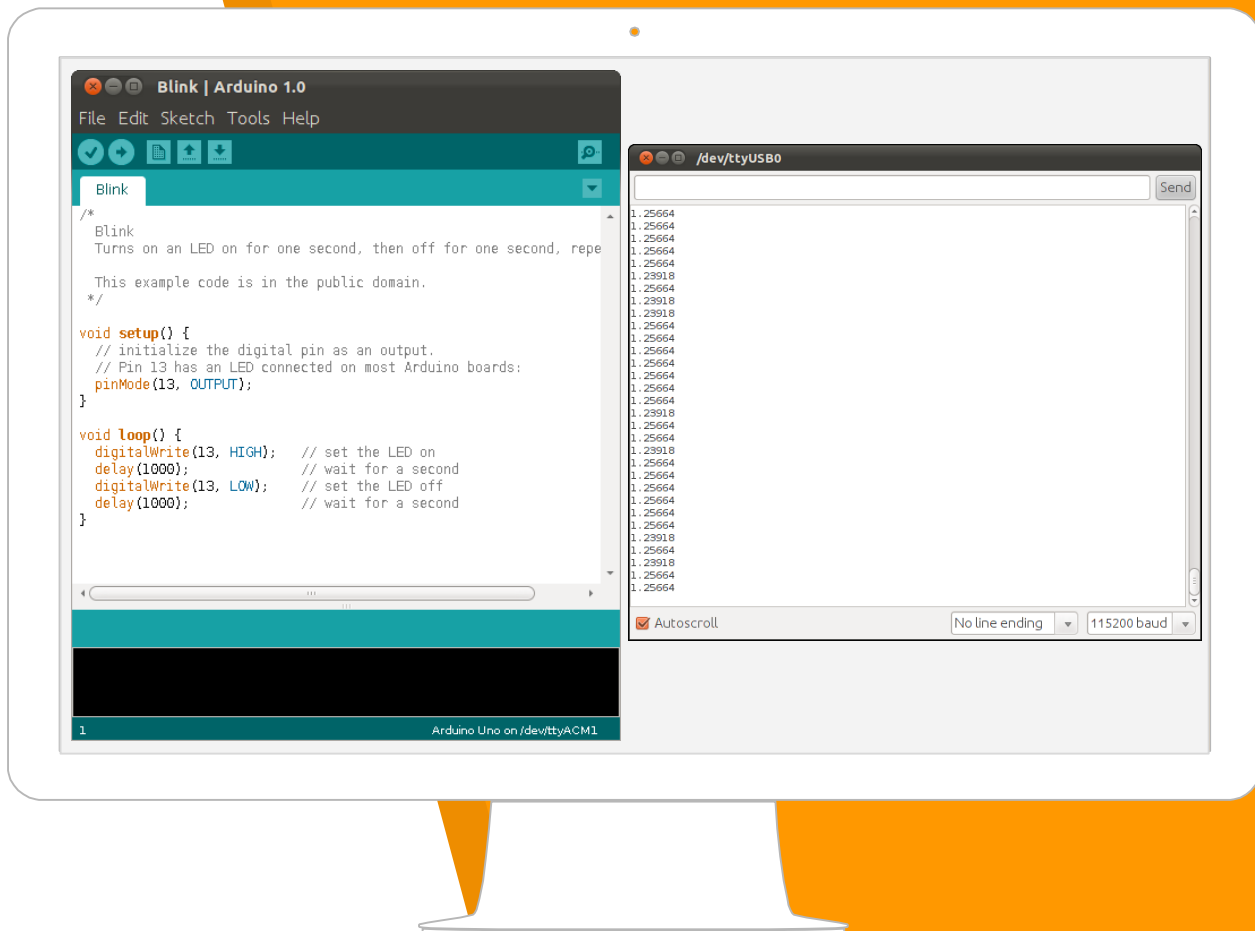






## INTERFACE ARDUINO IDE

- ▶ Salvar/Abrir
- ▶ Carregar/Compilar
- ▶ Monitor Serial
- ▶ Ferramentas
  - Placa
  - Porta
  - Bibliotecas



## 2. Experimentos

## Funções Básicas

### void setup()

Usada para declaração de pinos e objetos.  
Executa uma única vez.

### void loop()

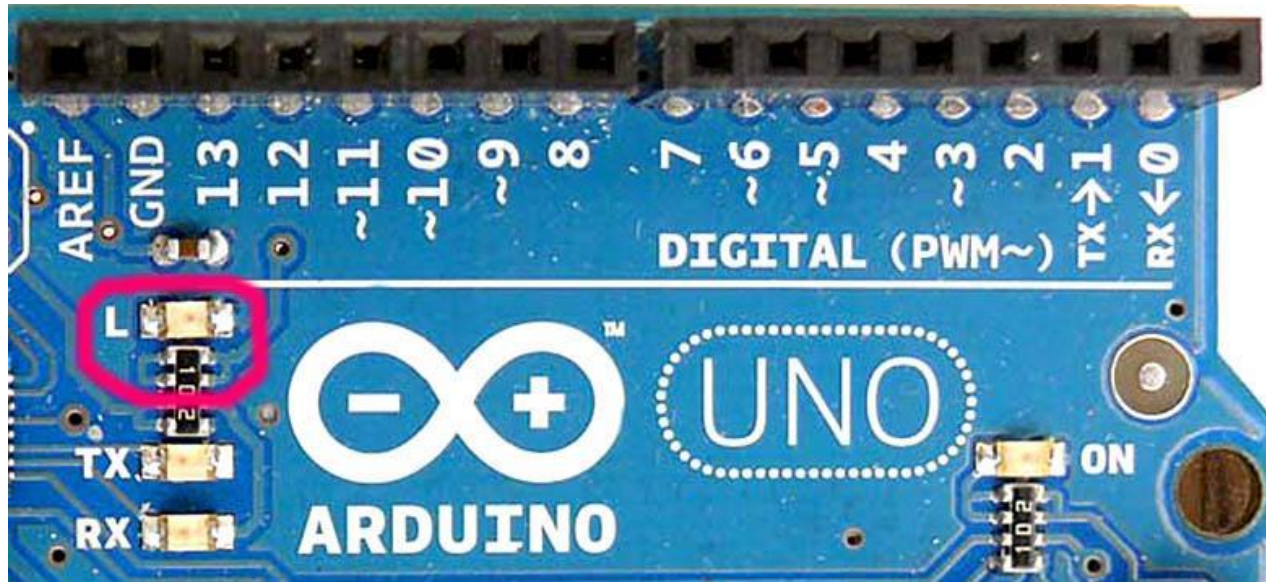
Usada para executar o código do usuário.  
Executa indefinidamente.

### Exemplo:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

# Resultado

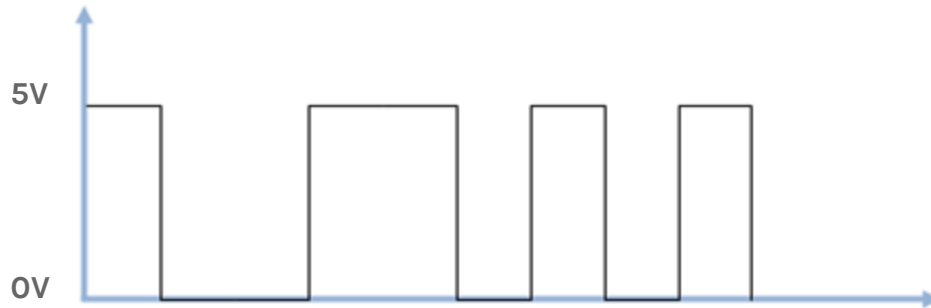


O LED L da placa deve piscar com intervalos de 1 segundo.

# Portas Digitais

Leem ou escrevem valores definidos binários (**HIGH/LOW**)

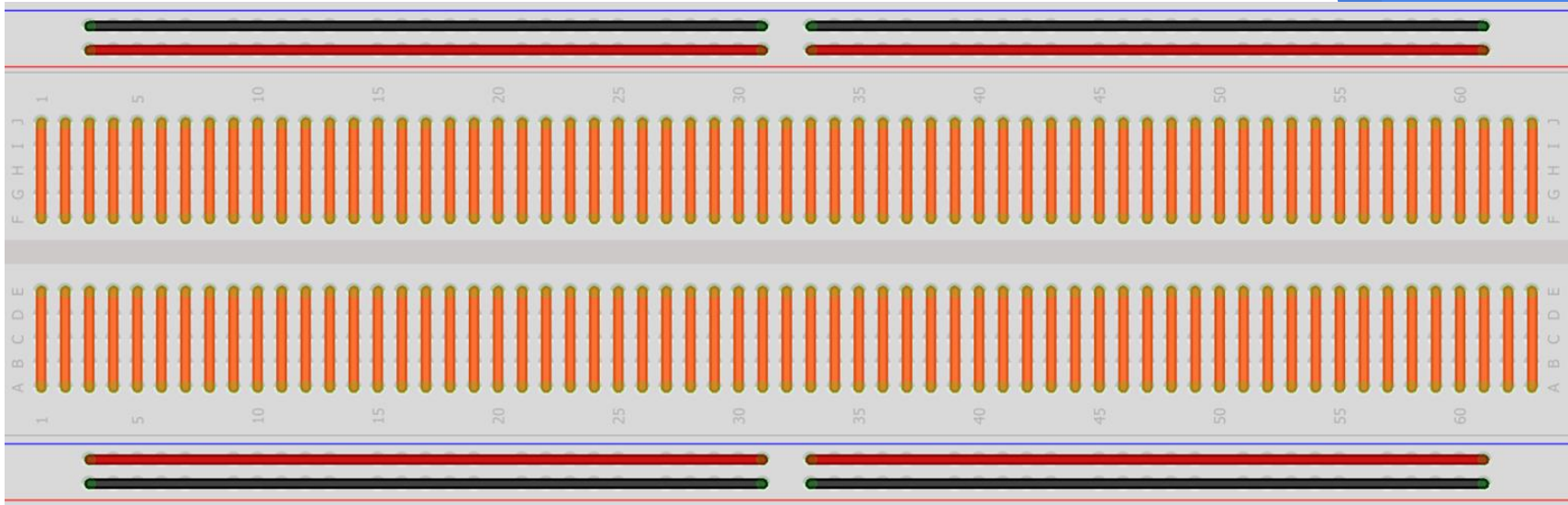
- ▶ **HIGH**: 2.2V - 5V
- ▶ **LOW**: 0V - 0.8V
  
- ▶ A função escrita é dada por **digitalWrite(pino, VALOR)**
- ▶ A função leitura é dada por **digitalRead(pino)**



# Protoboard

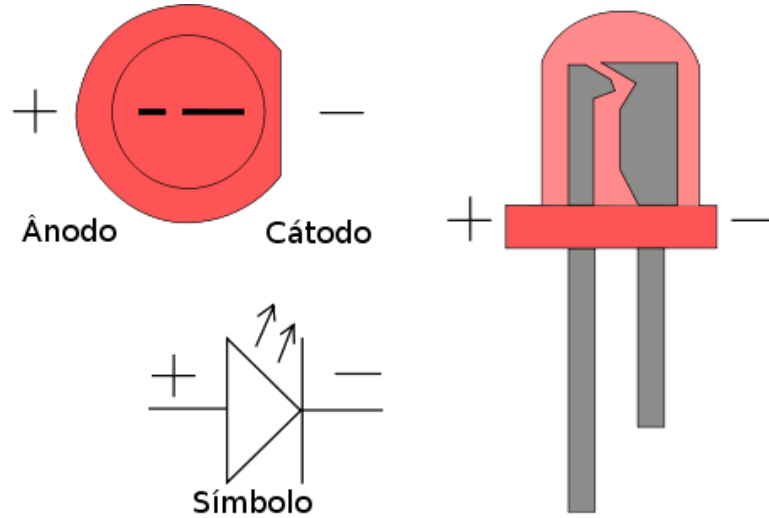
Utilizada para prototipagem de circuitos eletrônicos.

Trilhas adjacentes estão conectadas.



## ACENDENDO UM LED

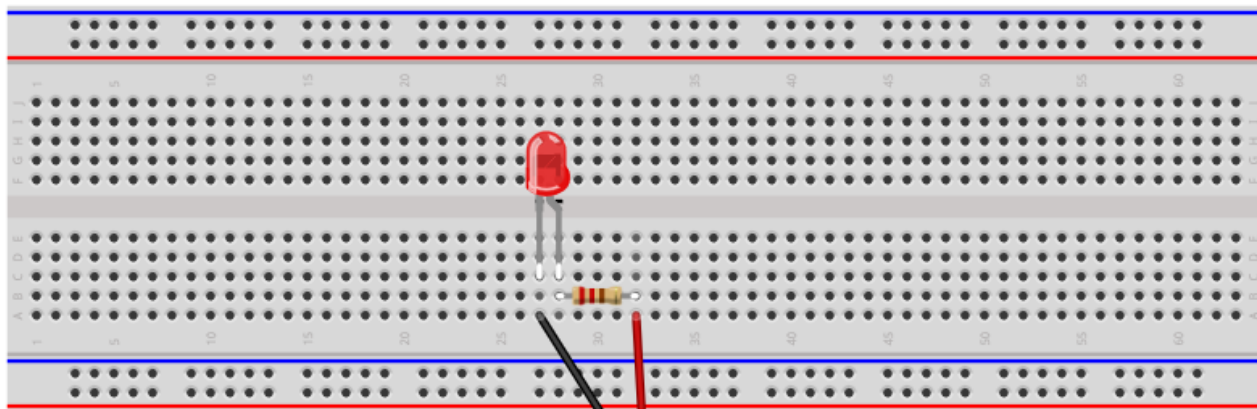
- ▶ Diodo emissor de luz (Light Emitting Diode)
- ▶ Possui **polaridade** (pernas positivas e negativas)
- ▶ Utilizado para acompanhar os processos realizados pelo Arduino
- ▶ **Não apresentam resistência interna**, sendo necessária a presença de resistores.



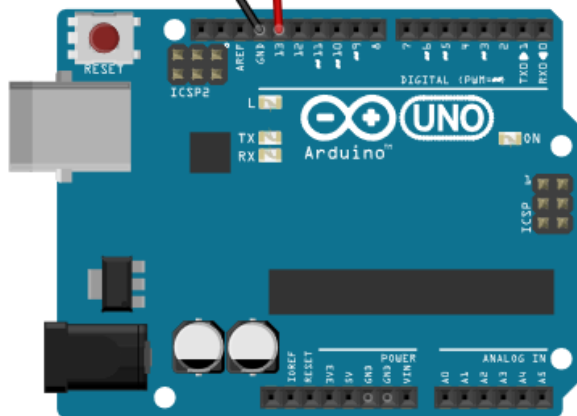
Polaridade LED

## ACENDENDO UM LED

É interessante organizar os fios de acordo com as funções de cada um. É normal usar vermelho para indicar o positivo e preto para indicar o negativo.



- ▶ Resistor de  $220\Omega$
- ▶ LED
- ▶ Protoboard



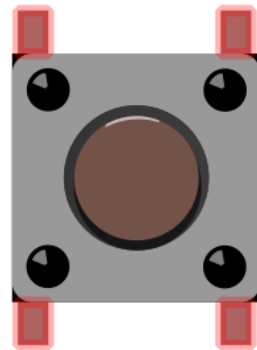
Esquemático Ligação



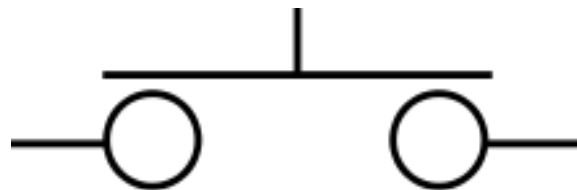
## ACENDENDO UM LED

```
void setup() {  
    // inicializa pino 13 como output  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);    // HIGH liga o LED do pino 13  
    delay(1000);               // Aguarda um segundo  
    digitalWrite(13, LOW);     // LOW desliga o LED do pino 13  
    delay(1000);               // Aguarda um segundo  
}
```

## LENDO UM BOTÃO



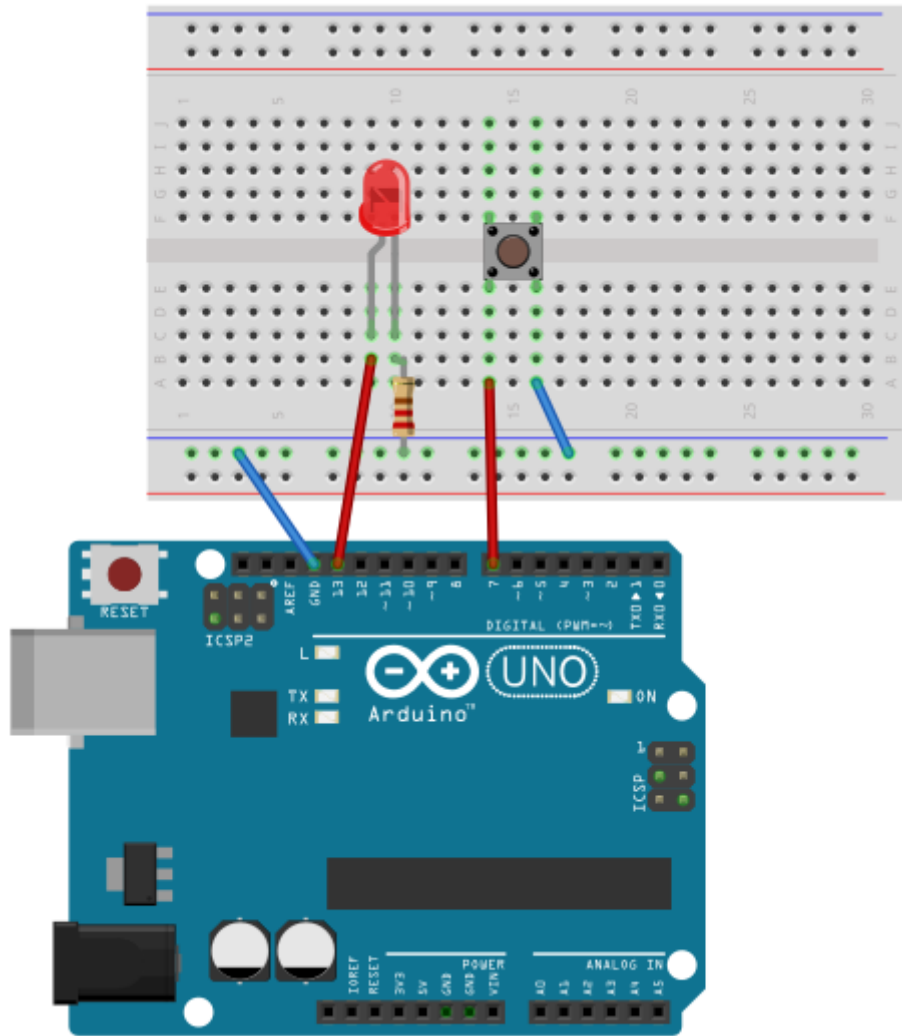
- ▶ Botões funcionam abrindo ou fechando uma ligação física ao serem pressionados
- ▶ Utiliza-se `pinMode(porta, INPUT_PULLUP);`
- ▶ A outra extremidade é ligada ao **GND** comum
- ▶ Lê **LOW** quando estiver pressionado e **HIGH** quando estiver solto.



Esquemático Botão

## LENDO UM BOTÃO

- ▶ Resistor de  $220\Omega$
- ▶ LED
- ▶ Protoboard
- ▶ Botão



## LENDO UM BOTÃO

```
int botao = 7, led = 13;

void setup() {
  pinMode(led, OUTPUT);          // Declara porta 13 como OUTPUT
  pinMode(botao, INPUT_PULLUP); // Declara porta 7 como INPUT
}

void loop() {
  if(digitalRead(botao) == HIGH)
  {
    digitalWrite(led, LOW);      // Desligar o LED
  }
  else
  {
    digitalWrite(led, HIGH);     // Ligar o LED
  }
}
```

# Variáveis

Utilizadas para armazenar **valores** (Ex: número de portas, leituras de sensores). Os valores são armazenados na memória do Arduino.

Tipos de variáveis mais utilizadas:

Tipo	Tamanho	Comentários
boolean	TRUE ou FALSE	Valor digital
char	-128 a 127	Declaração de Caracteres
int	-32 768 a 32 767	Uso geral
long	-2,147,483,648 a 2,147,483,647	Maior que int
float	-3.4E+38 a -3.4E+38	Precisão decimal

As variáveis possuem seu próprio escopo, e só funcionam dentro dele. Variáveis declaradas antes do setup() são globais.

# Variáveis

Ex:

```
int led = 13;
```

```
float pi = 3.14159265;
```

```
char letra = 'X';
```

```
boolean estado = FALSE;
```

```
unsigned char valor = 230;
```

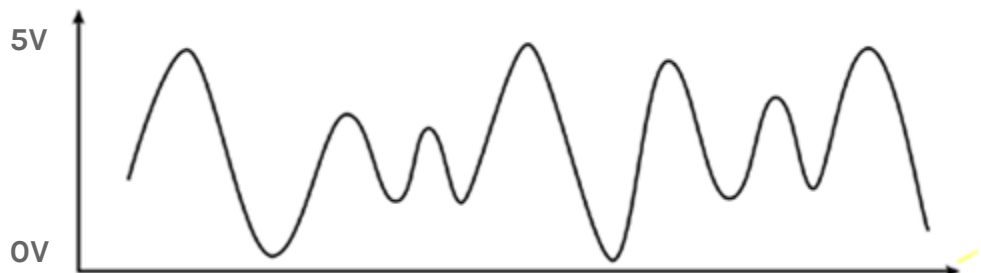
```
#define _SPEED 105
```

Os MACROS são declarados como #define MACRO valor e são pedaços de texto que são substituídos no código durante a compilação.

# Portas Analógicas

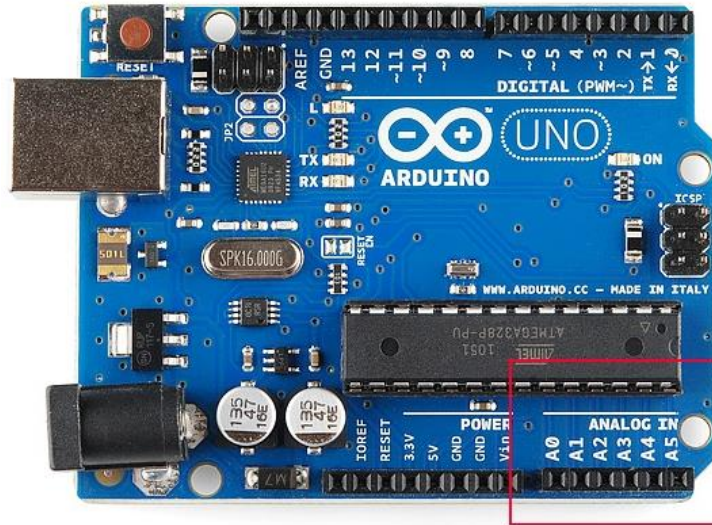
Possuem uma gama de valores de leitura analógica, variando a tensão de entrada. A resolução das portas é de **10 bits** (0 a 1024), aproximadamente **5 mV**.

- ▶ Podem ser utilizadas como saídas digitais.
- ▶ A função leitura é dada por **analogRead(pino)**



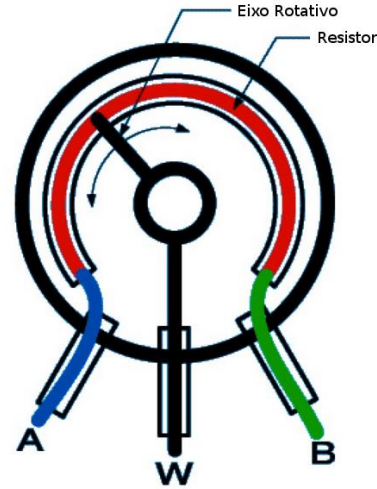
# Portas Analógicas

O Arduino **UNO** possui **6** pinos analógicos, nomeados de **A0** a **A5** (ou 14 a 19).





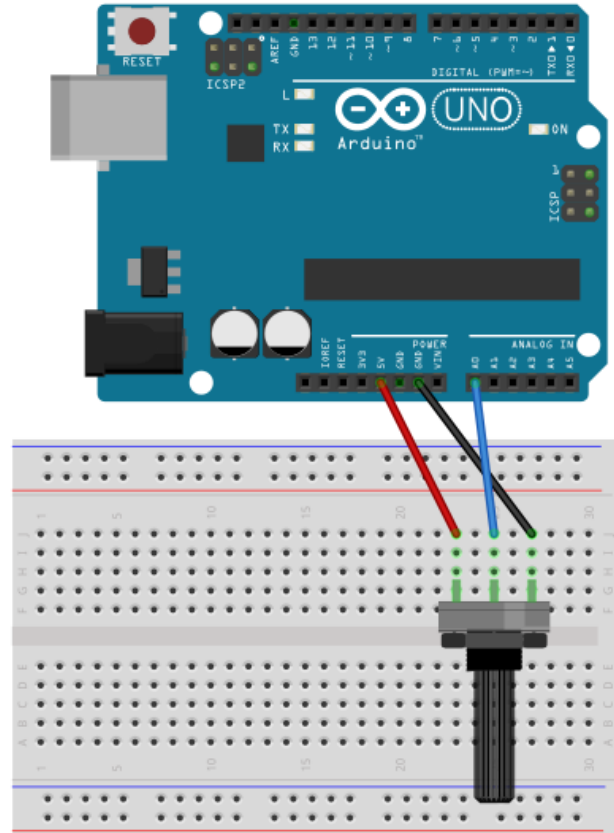
## FAZENDO A LEITURA DE UM POTENCIÔMETRO



- ▶ Potenciômetros são resistores variáveis que retornam a posição do eixo em forma de tensão.
- ▶ Utiliza-se `analogRead(porta);`
- ▶ Uma extremidade é ligada ao **5V** e a outra é ligada ao **GND** comum, em qualquer ordem. O pino do meio é conectado à porta.

## FAZENDO A LEITURA DE UM POTENCIÔMETRO

- Potenciômetro
- Protoboard



A ligação é simples, mas muitos tipos de sensores utilizam esse tipo de ligação para coletar dados, como alguns sensores de temperatura ou luminosidade.

## FAZENDO A LEITURA DE UM POTENCIÔMETRO

```
int sensorPin = A0;    // Pino do Potenciometro
int valor = 0;         // valor lido

void setup() {
    // Iniciar monitor Serial
    Serial.begin(9600);
}

void loop() {
    // Ler valor do sensor
    valor = analogRead(sensorPin);

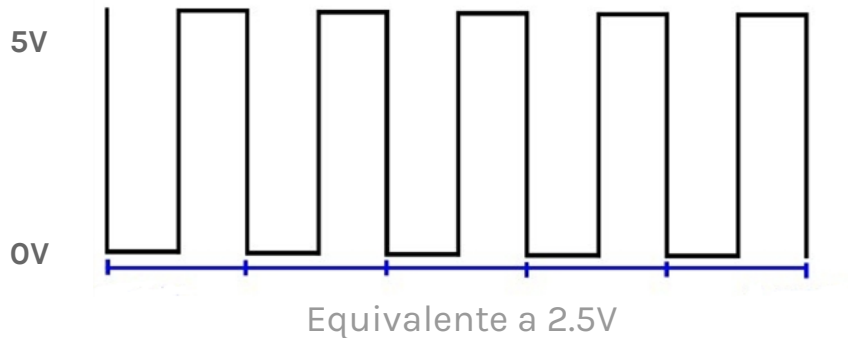
    // Mostrar valores na tela
    Serial.println(valor);

    delay(500);
}
```

# Portas PWM

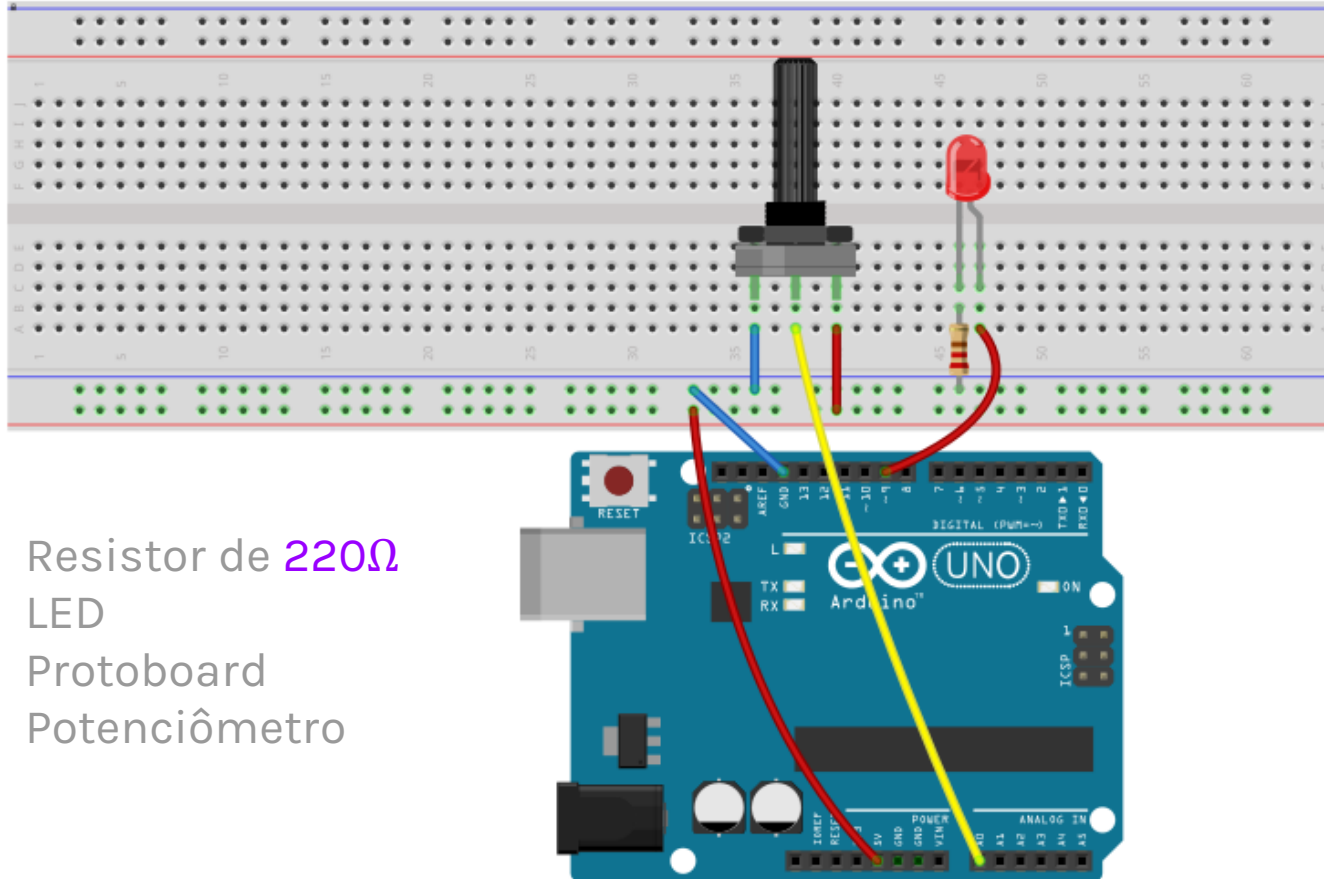
Possuem uma gama de valores de escrita analógica, devido à técnica de *Pulse Width Modulation*, variando a tensão de saída. A resolução das portas é de **8 bits** (0 a 255). São indicadas no Arduino por meio do símbolo ~.

- A função escrita é dada por `analogWrite(pino, VALOR)`



## CONTROLANDO O BRILHO DE UM LED

- ▶ Resistor de 220 $\Omega$
- ▶ LED
- ▶ Protoboard
- ▶ Potenciômetro



## CONTROLANDO O BRILHO DE UM LED

```
int potPin = A0, ledPin = 9, valor = 0;

void setup() {
}

void loop() {
    // Ler valor
    valor = analogRead(potPin);

    // Controlar brilho do LED
    analogWrite(ledPin, valor);

    delay(50);
}
```

## Monitor Serial



Utilizado para *debug* e *comunicação* entre PC e Arduino, além de comunicação entre Arduinos.

### Funções:

#### **Serial.begin(velocidade)**

Inicia a Serial com bit rate dado. Valor geralmente 9600/115200 bps.

#### **Serial.print(valor)**

Imprime variável, valor ou mensagem na tela.

#### **Serial.println(valor)**

Imprime variável, valor ou mensagem na tela e pula uma linha.

#### **Serial.write(valor)**

Envia dados binários para a Serial.

#### **Serial.available()**

Retorna verdadeiro se há dados disponíveis na Serial para serem lidos.

#### **Serial.read()**

Lê dados disponíveis na Serial (se houver).

# Monitor Serial

Ex:

```
Serial.println("Hello World");
```

```
char letra = Serial.read();
```

```
Serial.print(valor);
```

```
Serial.begin(115200);
```

```
Serial.println(analogRead(A0));
```



## Operadores

Servem para realizar **operações lógicas** ou **aritméticas**, dentre elas comparações, operações matemáticas, etc.

Operador	Função
+   -   *   /   %	Adição, subtração, multiplicação, divisão e resto.
&&        !	Bitwise AND e OR e NOT.
>   <   =   >=   <=   != ==	Maior, menor, igual, maior ou igual, menor ou igual e diferente.
++   --   +=   -=   /=   *=	Incrementos com atribuição.

# Operadores

Ex:

```
while(1<2)
```

```
cont++;
```

```
if( valor >= 3 && cont != 0 )
```

```
valor = num%10;
```

```
i += 10;
```

## Desvios Condicionais

### **IF(condição)**

Verifica uma condição e faz um desvio no código se for atendida.

Ex: `if(valor > 15)`

```
{  
    digitalWrite(13, LOW);  
}
```

### **WHILE(condição)**

Verifica uma condição e executa um código até que seja atendida.

Ex: `while(valor < 255)`

```
{  
    analogWrite(A0, valor);  
    valor++;  
}
```

## Desvios Condicionais

Variáveis podem ser declaradas dentro do laço for, no entanto essas variáveis só terão validade dentro do laço.

### FOR(variável; condição; incremento)

Laço de repetição que compara uma variável à condição e, se não atendida, repete o laço.

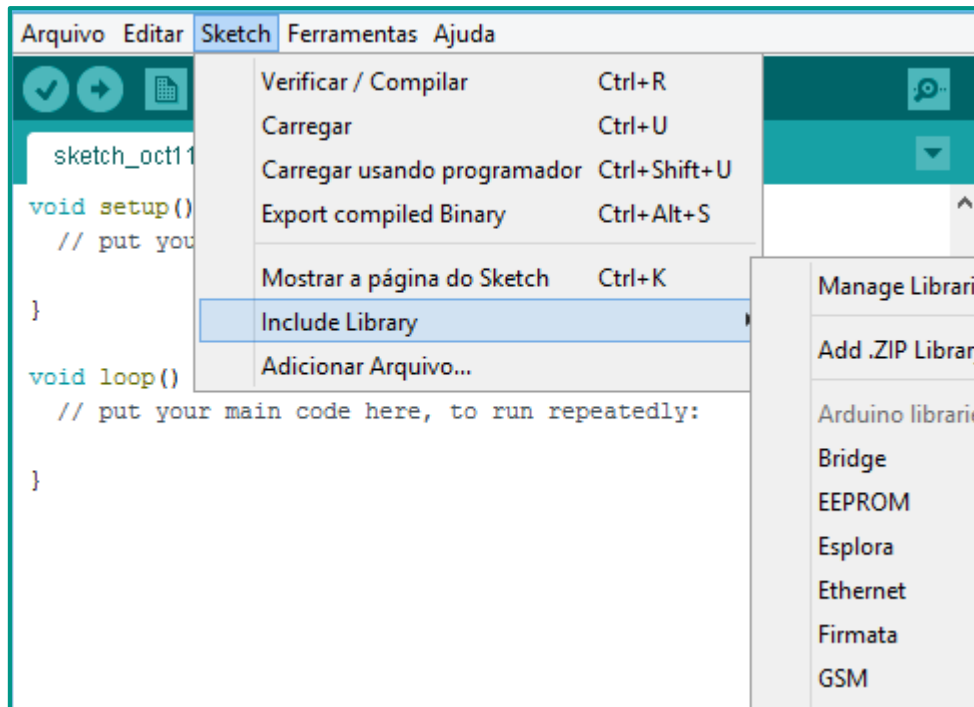
Ex: `for(i = 0; i<255; i++)`

```
{  
    analogWrite(A0, i);  
}
```

```
for(int num = 0; num < 10; num++)  
{  
    Serial.println(num);  
}
```

## USANDO BIBLIOTECAS

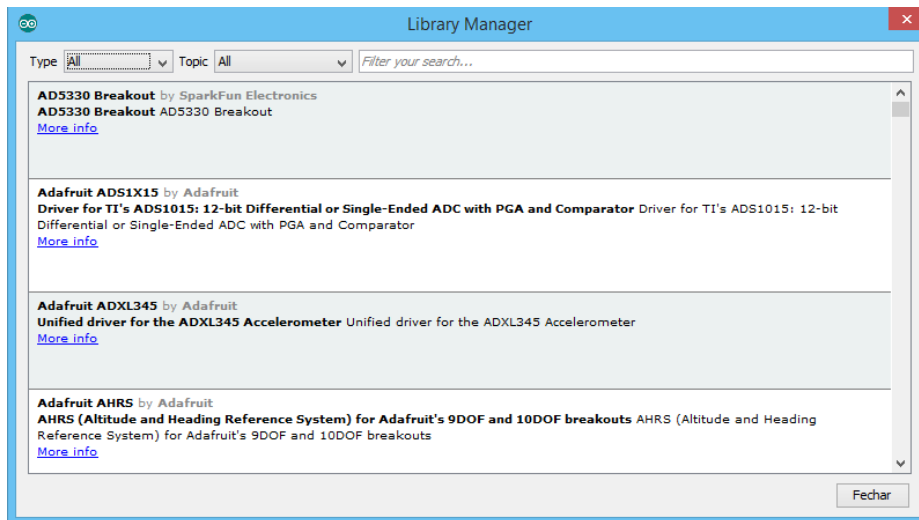
Bibliotecas são códigos já prontos desenvolvidos pela comunidade que **facilitam a implementação** e controle de diversos sensores/atuadores.



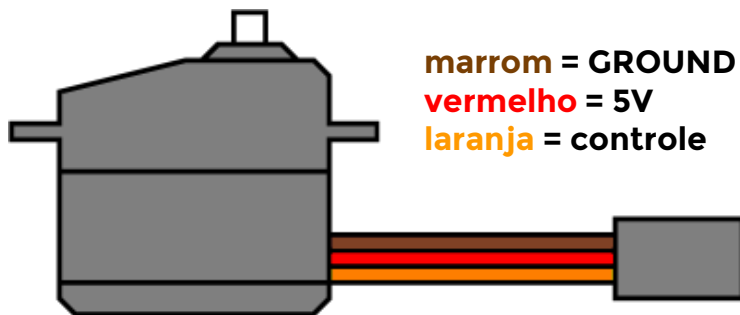
## USANDO BIBLIOTECAS

Bibliotecas podem ser incluídas com `#include <Biblioteca.h>` no topo do código (antes do setup).

Essas bibliotecas podem ser importadas tanto por arquivos .zip disponibilizados pelos autores quanto pelo repositório do Arduino ([Library Manager](#)).

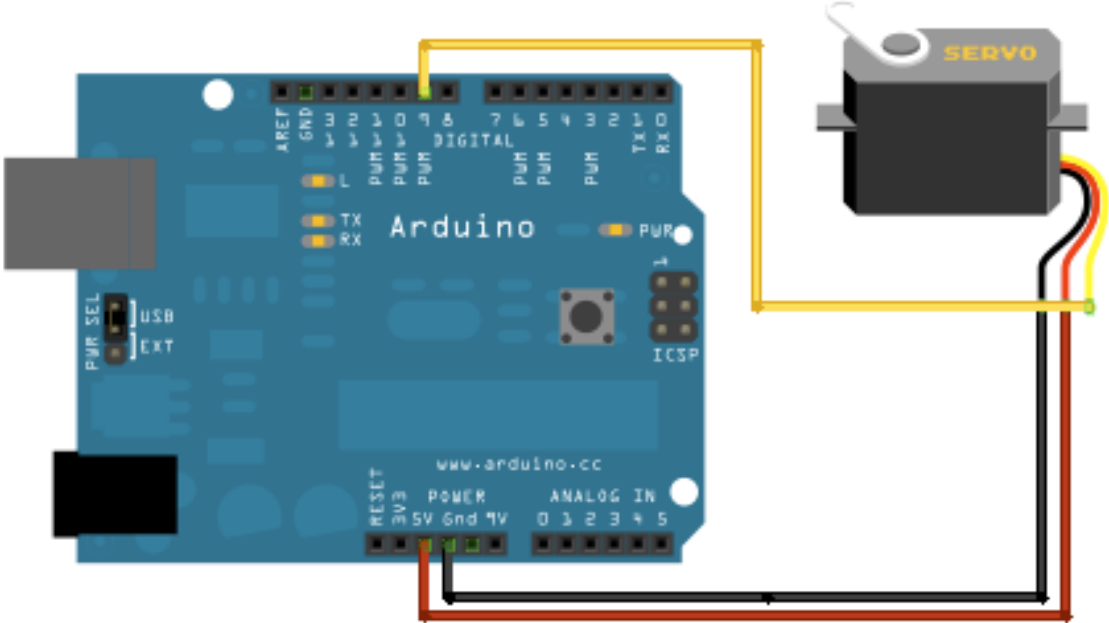


## USANDO BIBLIOTECAS



- ▶ **Servos** são atuadores rotacionais que permitem bom controle de velocidade angular e precisão.
- ▶ Utiliza-se a biblioteca **Servo.h** para gerir os servos conectados.
- ▶ O fio vermelho é ligado ao **5V** e o marrom ou preto ligado ao **GND** comum. O pino restante pode ser conectado a uma porta **PWM**.

USANDO BIBLIOTECAS





## USANDO BIBLIOTECAS

```
#include <Servo.h>

Servo myservo;  // Cria objeto Servo

int pos = 0;    // Posição do servo

void setup() {
  myservo.attach(9);  // Liga o Servo à porta 9
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(30);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(30);
  }
}
```

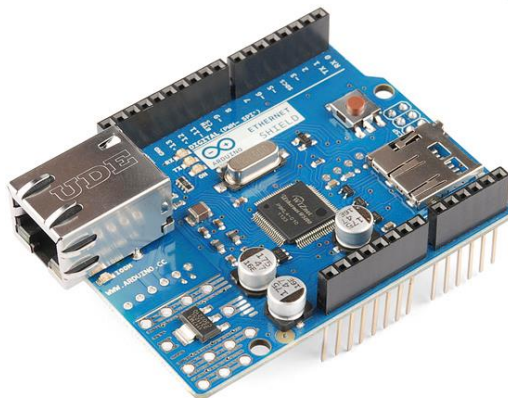
## UTILIZANDO SHIELDS

Shields são placas, conectadas no topo do Arduino, que estendem a funcionalidade do Arduino, **facilitando a conexão entre Arduino e periféricos** e tornando a aplicação mais **compacta** e **barata**, além de mais **segura**.

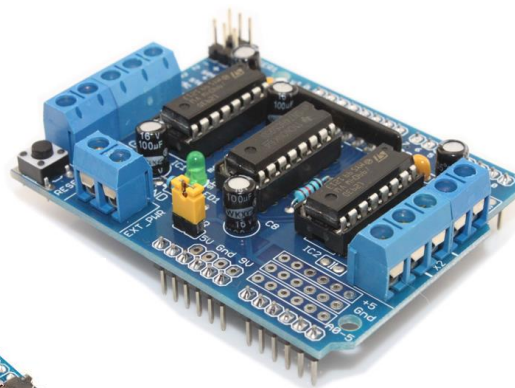
Shield “stackam” uma no topo das outras, sendo possível a utilização de mais de uma por vez.



SD Card Shield



Ethernet Shield



Motor Shield

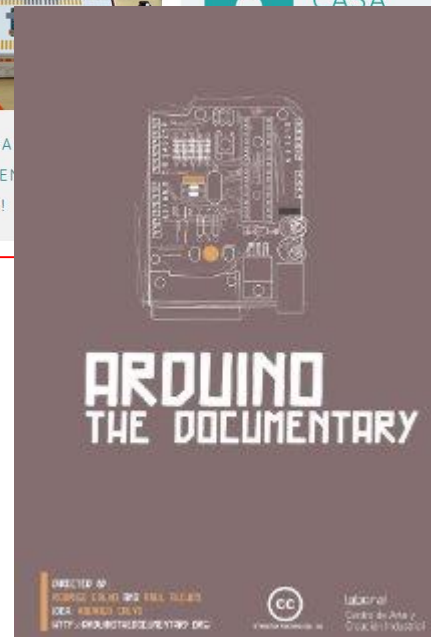
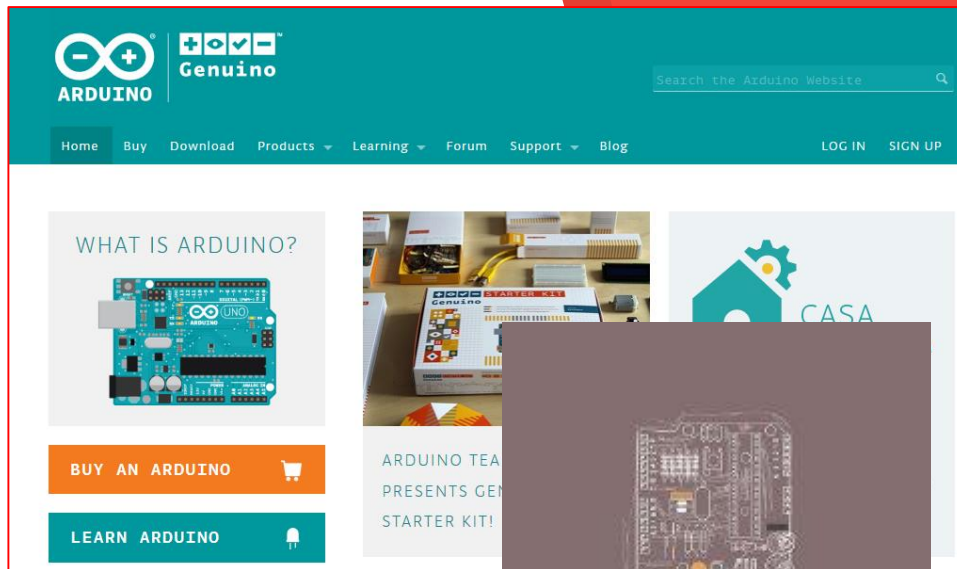
# Referências

Site oficial do Arduino:

[Arduino.cc](https://www.arduino.cc)

Documentário sobre o Arduino:

Arduino: The Documentary





**Obrigado!**

# Perguntas?

Dúvidas, críticas ou sugestões podem ser enviadas para [brunno.v@grad.ufsc.br](mailto:brunno.v@grad.ufsc.br).