

# Programando Para Arduino: Fundamentos

Elias de Souza Gonçalves

Faculdades Integradas de Caratinga - FIC  
Curso de Graduação em Engenharia Elétrica

08 de Junho de 2017

# Sumário

- 1 Introdução
- 2 Relembrando
- 3 Programação Para Arduino
- 4 Resumo da Aula
- 5 Atividades

## Direitos Autorais

Material elaborado com base no conteúdo disponível pelo site **Circuitar** [Chavier, 2017] e no Livro **Arduino Básico** [McRoberts, 2015].

Este material vai te ajudar a:

- Conhecer mais sobre o arduino;
- Entender como é o desenvolvimento de aplicações para arduino.

- Criar coisas legais;
- Possibilita a prática da eletrônica;
- Automação;
- Robótica;
- Internet das coisas.

## ① Fundamentos de Eletrônica

- Resistores;
- Lei de Ohm;
- Capacitores;
- Indutores.

## ② Eletrônica Digital

- Portas Lógicas;
- Tabela Verdade;
- Representação das Operações;
- Funções Lógicas Compostas.

# Sumário

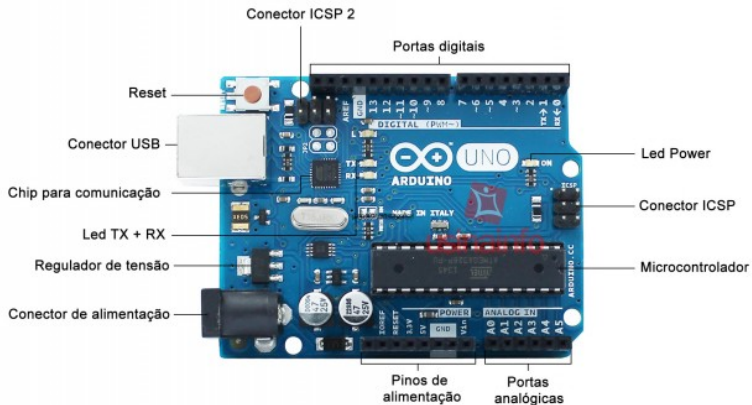
- 1 Introdução
- 2 Relembrando**
- 3 Programação Para Arduino
- 4 Resumo da Aula
- 5 Atividades

## O que é Arduino?

É um projeto que engloba *software* e *hardware* e tem como objetivo fornecer uma plataforma fácil para prototipação de projetos interativos, utilizando um microcontrolador. O *software* interage diretamente com o *hardware*, tornando possível integração fácil com sensores, atuadores e outros dispositivos eletrônicos.



# Placa Arduino Uno



1

<sup>1</sup><http://www.usinainfo.com.br/6333-infografico/>

# Sumário

- 1 Introdução
- 2 Relembrando
- 3 Programação Para Arduino**
- 4 Resumo da Aula
- 5 Atividades

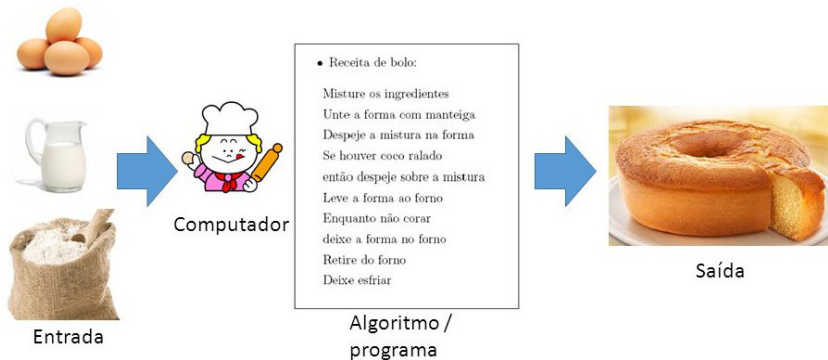
# Computador

Máquina que processa instruções. O processamento ocorre no **microprocessador**, dessa forma, entende-se que todo computador possui pelo menos um microprocessador.

O Arduino utiliza um **microprocessador** do modelo ATmega. Ele pode ser **programado** para diversas funções, mas faz apenas o que está no seu **programa**. Para executar qualquer outra função ele precisa ser **reprogramado**. Por conter todos os itens básicos de um computador em um único chip, microprocessadores como o ATmega também são chamados de **microcontroladores**.

# Programa de Computador

É uma lista de instruções para que o computador faça o que queremos.



# Linguagem de Máquina

Cada tipo de microprocessador entende um conjunto de instruções diferente, no seu próprio idioma, conhecido como **linguagem de máquina**. São as únicas linguagens que os computadores conseguem entender. (Alguns humanos também conseguem...)

Devido a dificuldade do ser humano em entender as linguagens de máquinas, adotamos o uso de **linguagens de programação**. Elas surgiram para que as pessoas consigam transmitir suas ideias para o computador realizar o processamento e cumprir as tarefas desejadas.



rede de ensino  
**DOCTUM**

# Compilador

Converte o programa escrito em uma **linguagem de programação** para a **linguagem de máquina**. A ação de converter o programa para a linguagem de máquina é chamada **compilar**.



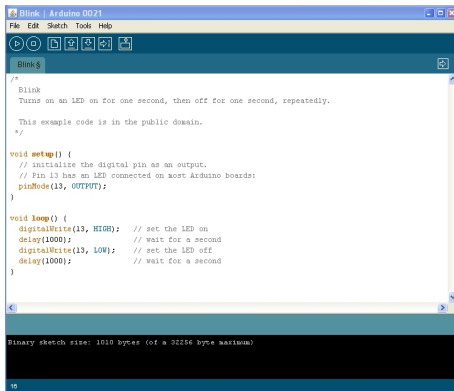
# IDE - *Integrated Development Environment*

Normalmente utilizamos um **ambiente de desenvolvimento integrado** que já possui um compilador e um editor de texto onde é possível escrever o programa numa linguagem de programação específica e compilá-lo.

No caso do Arduino, a linguagem de programação adotada é C++ e o ambiente de desenvolvimento é o Arduino IDE.

# Código Fonte

É o nome dado ao arquivo que contém o texto do programa escrito em uma linguagem de programação.

A screenshot of the Arduino IDE interface. The title bar reads 'Blink | Arduino 0021'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu is a toolbar with icons for running, saving, opening, and other sketch-related actions. The main text area contains the following code:

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
  */  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH);  // set the LED on  
  delay(1000);             // wait for a second  
  digitalWrite(13, LOW);   // set the LED off  
  delay(1000);             // wait for a second  
}
```

At the bottom of the window, a status bar indicates 'Binary sketch size: 1010 bytes (of a 32256 byte maximum)'. The file name 'Blink.ino' is visible in the bottom left corner.

<sup>4</sup><http://www.hobbytronics.co.uk/image/data/tutorial/arduino-install/>

# Programa/Algoritmo

É a forma de dizer para um computador o que ele deve fazer. Normalmente são escritos na forma que nós humanos entendemos, em linguagens de programação. No Arduino, um programa é conhecido como ***sketch***.

# Blink

Vamos analisar o programa *Blink*, disponível na IDE do Arduino.

**Veja, é uma sequência de comandos!**

```
int led = 13;

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

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

# Variável

Representa uma região da memória do computador (Arduino) usada para armazenar uma determinada informação<sup>5</sup>.

**Para usar uma variável no programa declara-se a variável abaixo:**

```
int led = 13;
```

Nesse caso a variável declarada é do tipo **int** e recebeu o nome de **led**

---

<sup>5</sup>Pode ser um número, um caracter ou uma sequência de texto.

# Tipo de Dado

Significa o tipo de informação que se pode armazenar na variável.

**No arduino, os tipos de dados mais comuns são:**

**boolean** : valor verdadeiro (**true**) ou falso (**false**)  
**char** : um caractere  
**byte** : um byte, ou sequência de 8 bits  
**int** : número inteiro de 16 bits com sinal (-32768 a 32767)  
**unsigned int** : número inteiro de 16 bits sem sinal (0 a 65535)  
**long** : número inteiro de 32 bits com sinal (-2147483648 a 2147483647)  
**unsigned long** : número inteiro de 32 bits sem sinal (0 a 4294967295)  
**float** : número real de precisão simples (ponto flutuante)  
**double** : número real de precisão dupla (ponto flutuante)  
**string** : sequência de caracteres  
**void** : tipo vazio (não tem tipo)

6

<sup>6</sup><https://www.circuitar.com.br/tutoriais/programacao-para-arduino-primeiros-passos/>

# Atribuição

Significa armazenar o valor na variável para usar posteriormente. O comando de atribuição em C++ é o =

No exemplo, o valor 13 é atribuído à variável **led**:

```
int led = 13;
```

O pino 13 do Arduino será utilizado para acender o LED, e armazenar essa informação para usar depois, ao longo do programa.

Os **valores fixos** usados no programa, como o valor 13 acima, são chamados de **constantes**, pois, diferentemente das variáveis, o seu valor não muda.

# Função


É uma sequência de comandos que pode ser reutilizada várias vezes ao longo de um programa. Fazemos uma declaração de função para criar e dizer o que ela faz.

## Declarando uma função:

```
void setup() {  
  pinMode(led, OUTPUT);  
}
```

A função declarada chamada `setup()`<sup>7</sup> executa os comando de outra função, chamada `pinMode()` (declarada automaticamente no arduino).

---

<sup>7</sup>Chamada uma vez só e depois é chamada a função `loop()` repetidamente até que o arduino seja desligado ou reiniciado. 



# Parâmetros e Retorno

## Parâmetros

Servem para enviar algum dado para a função quando ela é chamada.

## Valor de Retorno

A palavra chave que vem antes do nome da função na declaração define o tipo do valor de retorno da função. Toda vez que uma função é chamada, ela é executada e devolve ou retorna um determinado valor.

# Comentário

Trecho de texto no seu programa que serve apenas para explicar o código, sem executar nenhum tipo de comando no programa.

Na linguagem C++, um comentário pode ser escrito de duas formas:

- **Comentário de linha:** inicia-se com os caracteres `//` e deixa todo o resto da linha atual comentada.
- **Comentário de bloco:** inicia-se com os caracteres `/*` e termina com os caracteres `*/`. Todo o texto entre o início e o término se torna um comentário, pode ser composto de várias linhas.

# Exemplo

```
/*  
Declaracao da variavel "led"  
Indica que o LED esta conectado no pino digital 13 do  
    Arduino (D13).  
*/  
int led = 13;  
  
/*  
Declaracao da funcao setup()  
Esta funcao e chamada apenas uma vez, quando o Arduino  
    e ligado ou reiniciado.  
*/
```

## Exemplo (cont.)

```
void setup() {  
    // Chama a funcao pinMode() que configura um pino  
    // como entrada ou saida  
    pinMode(led, OUTPUT); // Configura o pino do LED  
    // como saida  
}  
  
/*  
Declaracao da funcao loop()  
Apos a funcao setup() ser chamada, a funcao loop() e  
    chamada repetidamente ate  
o Arduino ser desligado.  
*/
```

## Exemplo (cont.)

```
void loop() {  
  // Todas as linhas a seguir sao chamadas de funcao  
  // com passagem de parametros  
  // As funcoes sao executadas em sequencia para fazer  
  // o LED acender e apagar  
  digitalWrite(led, HIGH); // Atribui nivel logico  
  // alto ao pino do LED, acendendo-o  
  delay(1000);             // Espera 1000  
  // milissegundos (um segundo)  
  digitalWrite(led, LOW);  // Atribui nivel logico  
  // baixo ao pino do LED, apagando-o  
  delay(1000);             // Espera 1000  
  // milissegundos (um segundo)  
  
  // Apos terminar a funcao loop(), ela e executada  
  // novamente repetidas vezes,  
  // e assim o LED continua piscando.  
}
```



## Estruturas de Controle (Repetição)

São blocos de instruções que alteram o fluxo de execução do código de um programa. Elas permitem que sejam executados comandos diferentes, conforme uma **condição** ou **repetir** comandos várias vezes.

- A estrutura **while** executa um conjunto de comandos repetidas vezes enquanto uma determinada condição for verdadeira.
- A estrutura **for** nada mais é do que um **while** acrescido de um comando de inicialização e um comando de finalização.

```
while(condição) {  
    ...  
}
```

(a) Formato do while.

```
for(inicialização; condição; finalização) {  
    ...  
}
```

(b) Formato do for.

## Exemplo while

```
int i = 0; // Variavel para contar o numero de vezes
           que o LED piscou
// Pisca o LED tres vezes
while(i < 3) {
    digitalWrite(led, HIGH); // Atribui nivel logico
                             alto ao pino do LED, acendendo-o
    delay(1000);              // Espera 1000
                             milissegundos (um segundo)
    digitalWrite(led, LOW);  // Atribui nivel logico
                             baixo ao pino do LED, apagando-o
    delay(1000);              // Espera 1000
                             milissegundos (um segundo)
    i = i + 1;                // Aumenta o numero de
                             vezes que o LED piscou
}
delay(5000);                 // Espera 5 segundos para
                             piscar o LED de novo
```

## Exemplo for

```
int i; // Variavel para contar o numero de vezes que
      o LED piscou

// Pisca o LED tres vezes
for(i = 0; i < 3; i++) {
    digitalWrite(led, HIGH); // Atribui nivel logico
    alto ao pino do LED, acendendo-o
    delay(1000);              // Espera 1000
                              milissegundos (um segundo)
    digitalWrite(led, LOW);  // Atribui nivel logico
    baixo ao pino do LED, apagando-o
    delay(1000);              // Espera 1000
                              milissegundos (um segundo)
}
delay(5000);                  // Espera 5 segundos para
                              piscar o LED de novo
```



# Estruturas de Controle (Decisão)

- A estrutura **if** significa "se" em inglês. Ela verifica uma expressão e, **apenas se ela for verdadeira**, executa o conjunto de comandos.
- A estrutura **if-esle** é uma extensão do comando **if**. **Else** em inglês significa "caso contrário". "**Se** isso for verdadeiro, então faça aquilo, **caso contrário**, faça outra coisa".

```
if(condição) {  
    ...  
}
```

(c) Formato do if.

```
if(condição) {  
    ...  
} else {  
    ...  
}
```

(d) Formato do if-else.

## Exemplo if

```
int i = 0; // Variavel para contar o numero de vezes
           que o LED piscou
void loop() {
    digitalWrite(led, HIGH); // Atribui nivel logico
                             alto ao pino do LED, acendendo-o
    delay(1000);              // Espera 1000
                             milissegundos (um segundo)
    digitalWrite(led, LOW);  // Atribui nivel logico
                             baixo ao pino do LED, apagando-o
    delay(1000);              // Espera 1000
                             milissegundos (um segundo)
    i++;                      // Incrementa o numero de "
                             piscadas"
```

## Exemplo if (cont.)

```
if(i == 3) {  
    delay(5000);           // Espera 5 segundos para  
        piscar o LED de novo  
    i = 0;                 // Reinicia o contador de  
        numero de "piscadas"  
}  
}
```

## Exemplo if-else

```
int i; // Variavel para contar o numero de vezes que
      o LED piscou
// Pisca o LED tres vezes
for(i = 0; i < 3; i++) {
    if(i == 2) {
        digitalWrite(led, HIGH); // Atribui nivel logico
                                   alto ao pino do LED, acendendo-o
        delay(200);                // Espera 200
                                   milissegundos (um segundo)
        digitalWrite(led, LOW);   // Atribui nivel logico
                                   baixo ao pino do LED, apagando-o
        delay(1800);              // Espera 1800
                                   milissegundos (um segundo)
    }
}
```

## Exemplo if-else (cont.)

```
} else {  
    digitalWrite(led, HIGH); // Atribui nivel logico  
        alto ao pino do LED, acendendo-o  
    delay(1000);              // Espera 1000  
        milissegundos (um segundo)  
    digitalWrite(led, LOW);  // Atribui nivel logico  
        baixo ao pino do LED, apagando-o  
    delay(1000);              // Espera 1000  
        milissegundos (um segundo)  
}  
}  
delay(5000);                  // Espera 5 segundos para  
    piscar o LED de novo
```

# O que aprendemos hoje?

- O **arduino** é um pequeno computador, que por sua vez é um **microprocessador** capaz de realizar o processamento de apenas um programa por vez, sendo também conhecido como **microcontrolador**;
- Ainda não é possível conversar com o arduino em linguagem natural, por isso escrevemos um **programa** com o que queremos que ele faça em uma **IDE** utilizando a **linguagem de programação C++** para que o **compilador** do arduino transforme as instruções contidas no **código fonte** em **linguagem de máquina**;
- As linguagens de programação, no geral são compostas de **variáveis**, **tipos de dados**, **funções** e **estruturas** que nos permitem dizer exatamente o que queremos que o computador faça e possibilita ao computador "entender" o que queremos.

# Preparar o Ambiente de desenvolvimento para Arduino

- ❶ Baixar e instalar o *software* **Arduino IDE** no seu computador:
  - **Onde baixar:** <https://www.arduino.cc/en/Main/Software>
  - **Como instalar (Windows, Linux e MAC):**  
<http://renatoaloi.blogspot.com.br/2011/10/instalando-arduino-guia-completo.html>
- ❷ Baixar e instalar o *software* **fritzing** no seu computador:
  - **Onde baixar:** <http://fritzing.org/download/>
  - **Como instalar (Windows, Linux e MAC):**  
<http://fritzing.org/download/>

# Referências |



Chavier, L. F. (2017).

Programação para arduino - primeiros passos.  
07 jun. de 2017.



McRoberts, M. (2015).

Arduino Básico - 2ª edição: Tudo sobre o popular microcontrolador Arduino.  
Novatec Editora.



# Programando Para Arduino:

## Fundamentos

Elias de Souza Gonçalves



falarcomelias@gmail.com



<https://github.com/eliasouza>

