Tutorial Básico Arduíno

Allan Eduardo Feitosa

18 de dezembro de 2018

1 Introdução: O que é um microcontrolador?

Neste tutorial, vamos ter um primeiro contato com o Arduino Uno, um placa microcontroladora utilizada nas experiências e projetos das disciplinas Laboratório de Circuitos Elétricos (PSI3212) e Laboratório de Instrumentação Elétrica (PSI3214) da Poli-Elétrica.

Mas, num sentido geral, o que é uma placa microcontroladora? Primeiramente, a principal parte deste dispositivo é o *microcontrolador*, que segundo o site "TechTarget" é definido como

> "*um circuito integrado compacto desenhado para governar uma operação específica em um sistema embarcado*"[1].

Porém, esta definição pode nos trazer mais perguntas, como "o que é um circuito integrado?" ou "o que é um sistema embarcado?". Vamos simplificar, em prol do entendimento, a definição dada:

> "*um microcontrolador é um pequeno computador, capaz de realizar cálculos simples e utilizado para tarefas específicas.*"

Desta forma, microcontroladores são usados em um número imenso de atividades, tais como controle automático, telefones celulares, câmeras digitais, aparelhos micro-ondas, máquinas de lavar, robôs, aparelhos médicos, veículos e por aí vai. Estas atividades são realizadas pelo microcontrolador segundo uma programação, que é gravada em sua memória interna e executada por seus circuitos lógicos integrados[2]. Veremos como programar o Arduino Uno para realizar uma tarefa simples mais a frente no capítulo 3 deste tutorial.

Figura 1: Placa microcontroladora Arduino Uno.

Agora que já temos uma ideia do que é um microcontrolador, podemos falar da placa microcontroladora, que é, simplificadamente, o conjunto formado pelo microcontrolador e mais as

[1] Disponível em: https://internetofthingsagenda.techtarget.com/definition/microcontroller (14/12/2018).

[2] Segundo definição da Wikipedia (EN), um circuito integrado é um conjunto de circuitos eletrônicos em um pequeno substrato plano (ou "chip") de material semicondutor, normalmente silício.

funcionalidades que nos permitem utilizá-lo na prática, em todas aquelas tarefas possíveis. Estas funcionalidades representam geralmente partes físicas da placa, e por exemplo são: memória RAM, memória Flash, circuitos de entrada e saída (I/O), gerador de *clock*, interfaces de comunicação como Serial, entrada USB, entrada para microfone, conversores analógicos-digitais (AD) e/ou digitais analógicos (DA), e muito mais a depender da placa escolhida. Para o Arduino Uno, temos as seguintes funcionalidades:

- 32kB de memória Flash e 2kB de memória RAM.
- 14 pinos de entrada/saída digital, sendo 6 deles podem ser utilizados como saídas PWM[3];
- 6 entradas para sinais analógicos;
- 1 cristal de quartzo de 16MHz[4];
- 1 entrada USB;
- 1 botão de Reset.

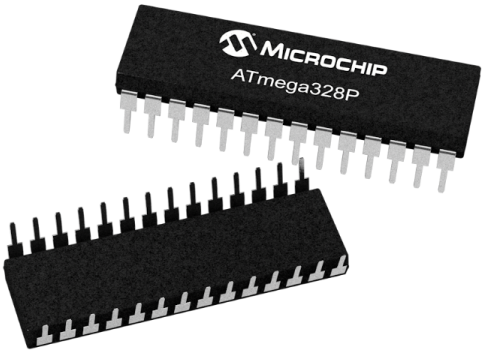

Figura 2: Microntrolador ATmega328P utilizado na placa Arduino Uno.

A Tabela 1 reproduz as especificações apresentadas para o Arduino Duo em seu site[5]. Nesta tabela encontramos as funcionalidades que citamos logo acima, juntamente as informações nominais de corrente e tensão; estas serão muito importantes quando utilizarmos a placa como parte de um circuito externo. Podemos observar também o nome do microcontrolador utilizado no Arduino Uno, "ATmega328P", e mais informações sobre ele podem ser consultadas no site de sua fabricante, a Microchip, em "https://www.microchip.com".

[3]PWM significa *Pulse-width Modulation*, e simplificadamente é uma forma de descrever um sinal $s(t)$ em um dado instante de tempo t como um pulso $p(t)$ que se mantém em uma amplitude constante (diferente de zero) durante um intervalo de tempo T de tal forma que o seu valor médio neste intervalo é igual ao valor médio do sinal $s(t)$ no mesmo intervalo. É uma técnica bastante utilizada, por exemplo, para controlar a potência em um LED ou mesmo de um motor elétrico. Você pode consultar um pouco mais sobre PWM neste curto resumo disponível em https://www.citisystems.com.br/pwm/.

[4]O cristal de quartzo, quando submetido à um certo campo elétrico, produz uma vibração cuja frequência é extremamente precisa, e este valor é utilizado pelo microcontrolador para controlar o tempo — pense no cristal de quartzo como o pêndulo de um relógio antigo.

[5]Disponível em: https://store.arduino.cc/usa/arduino-uno-rev3. (14/12/2018).

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Tabela 1: Especificações da placa microcontrolador Arduino Uno.

A Figura 3 a seguir nos mostra onde estão os principais componentes da placa Arduino Uno que possibilitam utilizar suas funcionalidades.

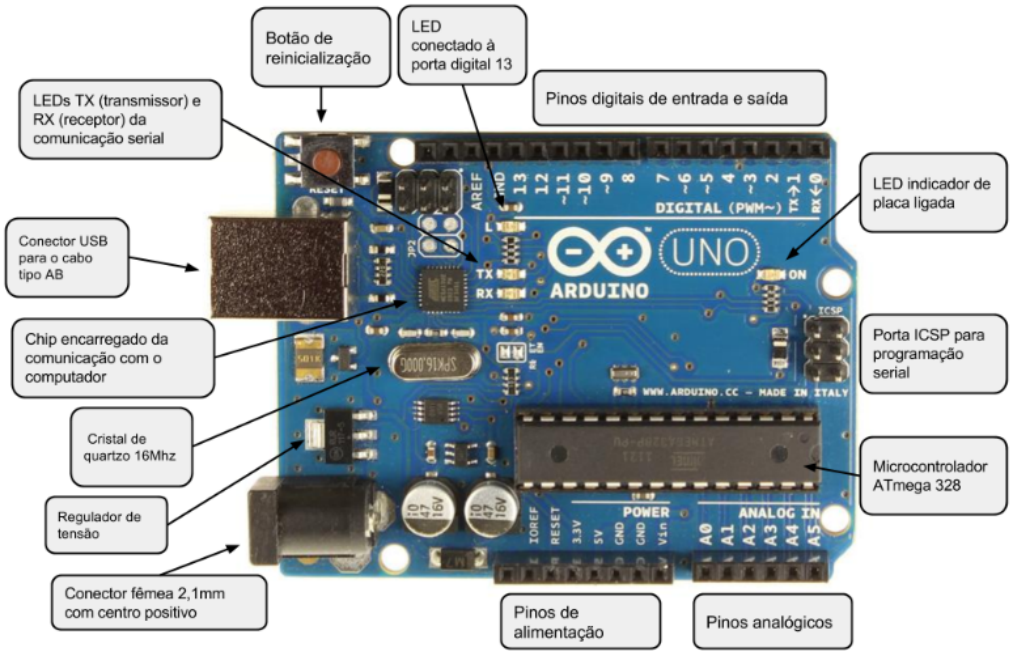

Figura 3: Principais componentes da placa Arduino Uno.

A seguir, vamos ver como podemos programar a placa Arduino Uno para executar alguma tarefa que desejamos.

2 IDE: a interface entre microcontrolador e usuário

Vimos algumas características de uma placa microcontroladora e suas funcionalidades. Mas como é que podemos utilizar todas essas coisas pra realizar alguma tarefa interessante? A resposta simples é: programar o microcontrolador para ele fazer o que a gente quer que ele faça e utilizar o que for necessário. Mas o real problema é outro; como fazemos pra programar um microcontrolador? Que linguagem se utiliza para isto?

Pois bem, a única linguagem que um circuito integrado entende (como um microcontrolador) é o *código de máquina*, uma sequência de bits que representam os comandos que o circuito deve realizar[6]. Embora seja possível realizar essa programa em código de máquina diretamente no microcontrolador, não é muito recomendável, afinal escreveríamos um código inteiro de 0's e 1's. Por nossa sorte, é possível utilizar linguagem de alto nível, tais como C, Python entre outras, para descrever o que queremos que o microcontrolador faça; para isto, necessitamos de um tradutor da linguagem de alto nível, própria dos seres humanos, para o código de máquina, próprio para o nível de circuito. Este "tradutor" é chamado de *compilador*, e é a primeira coisa que precisamos utilizar para programar nosso Arduino.

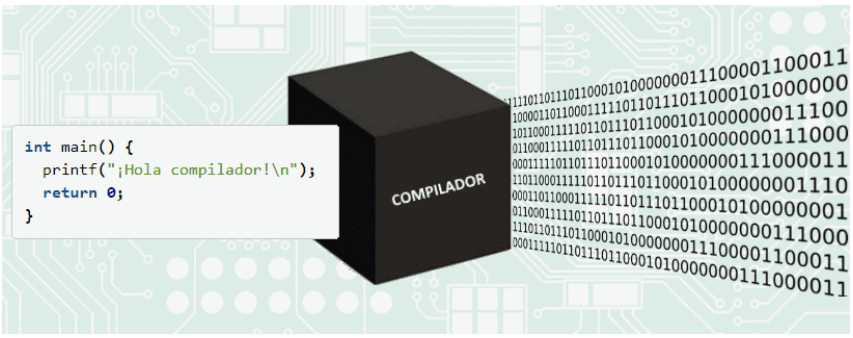

Figura 4: Ideia básica de um compilador.

Temos uma outra boa notícia! Tanto o compilador quanto o ambiente para escrever o código em alto nível estão disponíveis numa IDE — *Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado —, que é um software utilizado, entre outras coisas, para programação de dispositivos. O IDE também conta com uma vantagem: ela simplifica a interface entre usuário e o microcontrolador; sem um IDE, utilizar os componentes como entradas/saídas ou LEDS pode se tornar uma tarefa bastante complicada.

Utilizamos o IDE do próprio Arduino, disponível para download no site do Arduino[7], juntamente com as instruções para sua instalação (há também a opção de utilizar o IDE online disponibilizado). A Figura 5 abaixo mostra como vemos esse IDE no nosso computador. Ins-

[6]Existe uma versão de códigos de máquina legível para seres humanos, e esta se chama *Assembly*. Veja https://en.wikipedia.org/wiki/Assembly_language para saber um pouco mais sobre esta linguagem.

[7]Link para download: https://www.arduino.cc/en/Main/Software. O Arduino IDE também pode ser baixado no Microsoft Store.

talado o IDE no seu computador (Figure 5), vamos fazer nosso primeiro teste de programar o Arduíno.

3 Primeiro programa teste: piscando um LED

Para nos acostumar com o procedimento de gravar um código utilizando o IDE do Arduino, vamos fazer com que ele realize uma tarefa bastante simples: piscar um LED. Vamos a seguir apresentar um passo a passo de como fazer tudo corretamente.

Figura 5: Ambiente da IDE para programar o Arduino.

1. Primeiramente, abra o IDE do Arduino instalado no seu computador. Você deverá ver uma janela muito parecida com a da Figura 5.

2. Vá no menu Arquivo → Exemplos → 01.Basics → Blink. Isto abrirá uma nova janela chamada ”Blink”, como um código-exemplo já pronto (Figura 6). Este código é um dos muitos exemplos básicos disponível para utilização do Arduíno. Vamos pausar um pouco nosso passo a passo para entender um pouco mais sobre este código.

    - Na primeira parte do código temos o cabeçalho, que não faz parte do código em si, mas nos trás algumas informações úteis sobre a utilização do código, procedência, autoria, data etc.

- A segunda parte realiza a inicialização do pino do LED como saída. O *setup* é executado apenas quando o botão de reset é apertado ou após a placa ser alimentada. Perceba que existem comentários com informações úteis, explicando o que este trecho faz.
- Na terceira parte temos o *loop* principal, que é executado indefinidamente, só parando quando o Arduino é desligado ou forçado a parar de outra maneira. Dentro desse laço, é executado uma sequência de LIGAR/DESLIGAR o LED, utilizando o comando "digitalWrite" para isto. Perceba como é necessário dois argumentos para este comando, "LED_BUILTIN" e "HIGH" (para "ligado") ou "LOW" (para ''desligado"). Perceba também que entre os comandos com argumentos "HIGH" e "LOW", existe um comando "delay" com argumento setado para 1000. Este comando acrescenta uma espera na execução do código em milissegundos, e com o argumento "1000" temos 1 segundo de espera. Atente também como ao lado de cada comando, há um comentário que nos ajuda a entender cada passo executado neste laço. Bons códigos sempre são escritos dessa maneira! ;)

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

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

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

Figura 6: Código Exemplo para piscar o LED da placa Arduino, com frequência de 1 segundo.

3. Agora que já compreendemos um pouco melhor a estrutura do código e como ele funciona, o próximo passo é compilá-lo e gravá-lo. Lembre-se que compilar significa traduzir o código escrito no ambiente da IDE para um código de máquina. Mas antes de tudo, precisamos conectar o Arduino ao nosso computador via USB para gravar o código, e vamos utilizar um cabo como da Figura 7. Faça as conexões devidas, e o Arduino deve "acordar" (o led "ON" na placa deve acender); ou seja, o próprio cabo USB é utilizado para alimentar a placa.

4. Vamos informar à IDE qual é a placa que estamos utilizando. Vá em Ferramentas → Placa e selecione "Arduino/Genuino Uno".

5. A seguir, precisamos informar a porta serial[8] para comunicação com o Arduino. Vá em Ferramentas → Porta e selecione a porta COM que contenha a informação "Arduino/Genuino Uno".

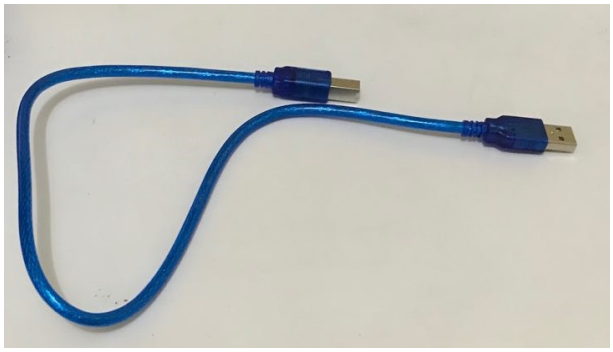

Figura 7: Cabo USB utilizado para comunicação entre o computador pessoal e a placa microcontroladora.

6. Com isso, estamos prontos para gravar este código no Arduino. Clique no botão "Carregar" (segundo botão do IDE, como na Figura 8), e o código começará a ser compilado, e em sequência é realizada a gravação via USB. Perceba que uma barra de progresso aparecerá na região abaixo do código no IDE, junto com a frase "Compilando Sketch". Se tudo correu bem, as mensagem "Carregando" e "Carregado" deveram aparecer, significando que o código já está gravado, e o LED já deverá estar piscando na taxa de 1 segundo.

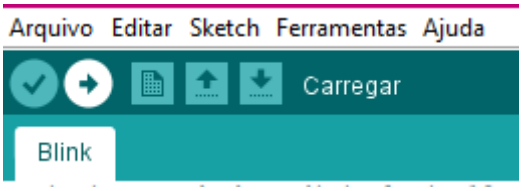

Figura 8: Botão "Carregar" da IDE.

[8]Bem simplificadamente, as portas serias do seu computador são formas de comunicação com alguns periféricos externos, tal como o microcontrolador que estamos utilizando.

Note que na parte inferior da IDE, com o fundo preto, podemos observar informações sobre o uso da memória do Arduino e a capacidade máxima.

Estes foram os passos para programar corretamente o Arduino utilizando seu IDE. Se houvesse algum problema com o código escrito (por exemplo, um ponto e vírgula faltando no fim de um comando), o IDE pararia o processo de compilação, não gravaria o código e informaria o usuário qual o erro e onde ele se encontra no código. Não se preocupe, você terá várias oportunidades pra ver isso acontecendo. =P

Podemos brincar um pouco mais com este código, diminuindo os valores dentro do comando "delay" para deixar a frequência do LED maior ou menor. Teste colocar 500, 100, 50, 10. Com 10 você não conseguirá mais ver o LED piscar; ele parecerá sempre acesso, porém a intensidade luminosa estará menor. Você consegue explicar por quê?