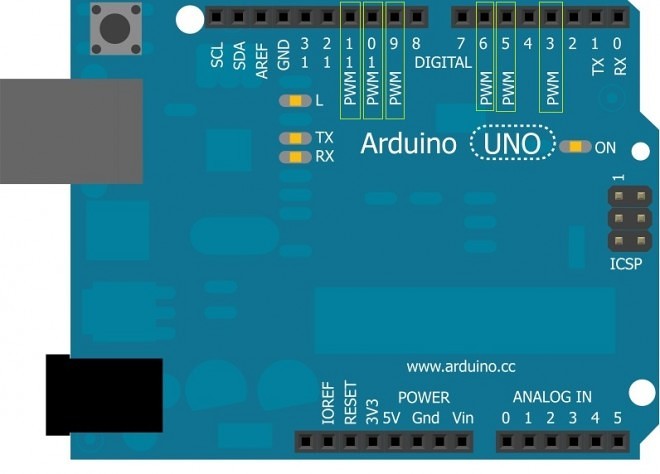
# Arduino - Saídas PWM

por [Fábio Souza](http://www.embarcados.com.br/author/fabio-souza/) em sexta-feira, 10 de janeiro de 2014.



Continuando a sequência de artigos básicos sobre a plataforma Arduino, vamos aprender como utilizar sinais PWM na placa Arduino UNO.

## O que é PWM?

PWM, do inglês Pulse Width Modulation, é uma técnica utilizada por sistemas digitais para variação do valor médio de uma forma de onda periódica. A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto. Esse tempo é chamado de duty cycle, ou seja, o ciclo ativo da forma de onda. No gráfico abaixo são exibidas algumas modulações PWM:



Arduino PWM

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

Analisando as formas de onda nota-se que a frequência da forma de onda tem o mesmo valor e varia-se oduty cycle da forma de onda. Quando o duty cicle está em 0% o valor médio da saída encontra-se em 0 V e consequentemente para um duty cycle de 100% a saída assume seu valor máximo, que no caso é 5V. Para um duty cycle de 50% a saída assumirá 50% do valor da tensão, 2,5 V e assim sucessivamente para cada variação no duty cycle. Portanto, para calcular o valor médio da tensão de saída de um sinal PWM pode-se utilizar a seguinte equação:

Vout = (duty cycle/100)\* Vcc

Onde:

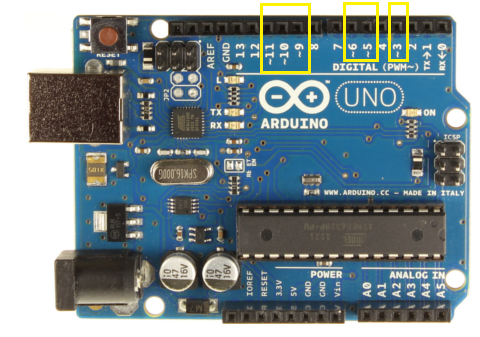
* Vout - tensão de saída em V;
* duty cycle - valor do ciclo ativo do PWM em %;
* Vcc - tensão de alimentação em V.

PWM pode ser usada para diversas aplicações, como por exemplo:

* controle de velocidade de motores;
* variação da luminosidade de leds;
* geração de sinais analógicos;
* geração de sinais de áudio.

## Arduino PWM

A placa Arduino Uno possui pinos específicos para saídas PWM e são indicados pelo caracter ‘~’ na frente de seu número, conforme exibido a seguir:



Saídas PWM na placa Arduino UNO

Observa-se na figura acima, que a Arduino Uno possui 6 pinos para saída PWM (3,5,6,9,10,11). Para auxiliar na manipulação desses pinos a plataforma possui uma função que auxilia na escrita de valores de duty cyclepara esses pinos.

### Função analogWrite()

A função **analogWrite()** escreve um valor de PWM em um pino digital que possui a função PWM. Após a chamada dessa função, o pino passa a operar com uma onda quadrada de frequência fixa e com duty cycleconforme valor passado pela função. A frequência dessa onda, na maioria dos pinos é em tordo de 490 Hz, porém, os pinos 5 e 6 da Arduino UNO operam em 980 Hz.

Para utilizar a função **analogWrite()** , deve-se configurar o pino correspondente como saída digital. É interessante notar que essas saídas não são conversores digital-analógico como o nome sugere, e estes pinos não estão relacionados às entradas analógicas.

A função analogWrite deve ser utilizada da seguinte forma:

**Sintaxe:**

analogWrite(pino, valor);

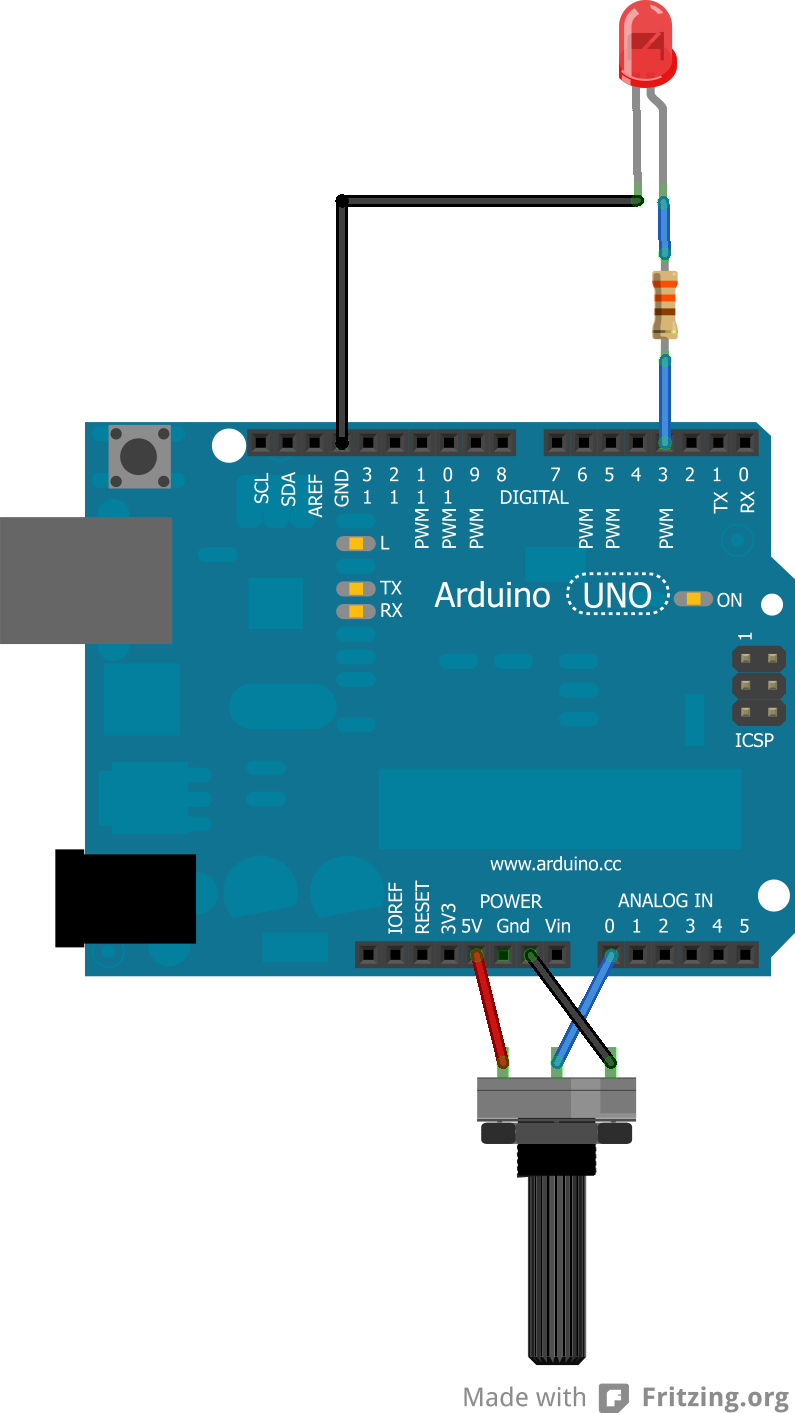
Onde:

* pino corresponde ao pino que será gerado o sinal PWM;
* valor corresponde ao duty cycle, ou seja, o valor que permanecerá em nível alto o sinal.

O valor deve ser de 0 a 255 onde com  0 a saída permanece sempre em nível baixo e 255 a saída permanece sempre em nível alto.

### Exemplo - Variando o brilho de um LED

Vamos utilizar a montagem a seguir para exemplificar o uso de um sinal  PWM para variação do brilho de um LED:



O circuito possui um LED ligado ao pino 3 (PWM) com seu devido resistor e um potenciômetro ligado à entrada analógica 0.  A ideia é controlar a intensidade do brilho do LED através da variação do valor do potenciômetro. Vejamos o sketch a seguir:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | /\*  PWM  controla a luminosidade de um led conforme o valor do potenciometro  \*/    int ledPin = 3; // pino do led  int analogPin = 0; // pino para leitura do potenciômetro  int val = 0; //variável para armazenar o valor lido    void setup()  {      pinMode(ledPin, OUTPUT); // configura pino como saída  }    void loop()  {      val = analogRead(analogPin);  // le o valor analógico      analogWrite(ledPin, val / 4); // aciona led com o valor analógico lido                                     //dividido por 4 para ajustar ao valor                                     //máximo que pode ser atribuído a função  } |

## Conclusões sobre Arduino PWM

A função **analogWrite()** fornece um modo simples para se trabalhar com sinais PWM, porém não fornece nenhum controle sobre a frequência do sinal aplicado ao pino. Em alguns casos a frequência do sinal é muito importante para o sistema, como por exemplo a frequência de acionamento de uma bobina de um motor. Em um artigo futuro vamos abordar como manipular os registradores do ATmega328 para alterar a frequência do sinal PWM.

Para início utilize o sketch apresentado para variar não só o brilho de LEDs, mas para varia a velocidade de motores de corrente contínua, criar cores em LEDs RGBs, etc. Use a imaginação em seus projetos.