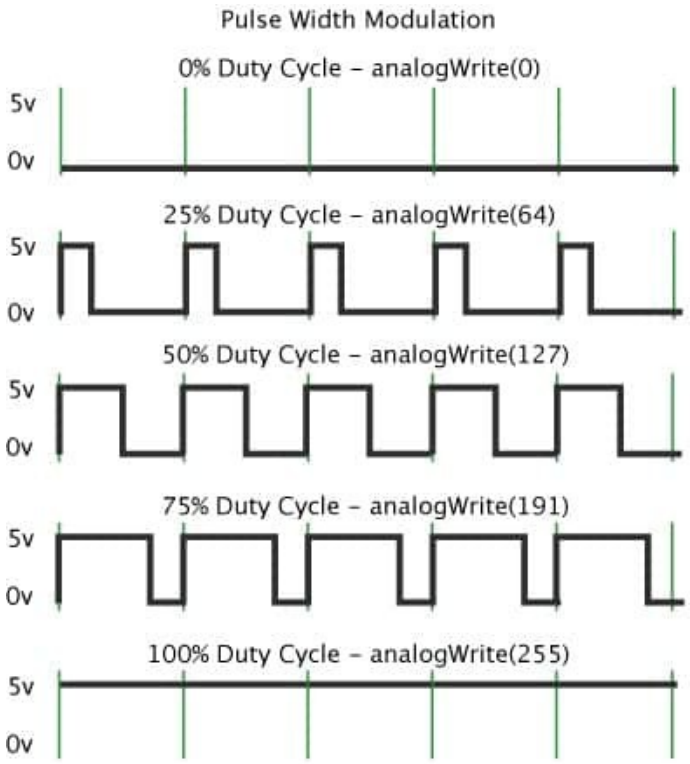
PWM

PWM, do inglês *Pulse Width Modulation* (modulação por largura de pulso), é uma técnica utilizada por sistemas digitais para simular uma saída analógica a partir da saída digital. A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto, variando assim a média do sinal de saída. Esse tempo em que o sinal fica em nível alto com relação ao período, é chamado de *duty cycle*, ou seja, o ciclo ativo da forma de onda. No gráfico abaixo são exibidas algumas modulações PWM:



5 exemplos de sinal digital com duty de 0, 25%, 50%, 75% e 100%

Analisando as formas de onda acima, nota-se que a frequência da forma de onda é a mesma para os 5 exemplos e varia-se o *duty 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.

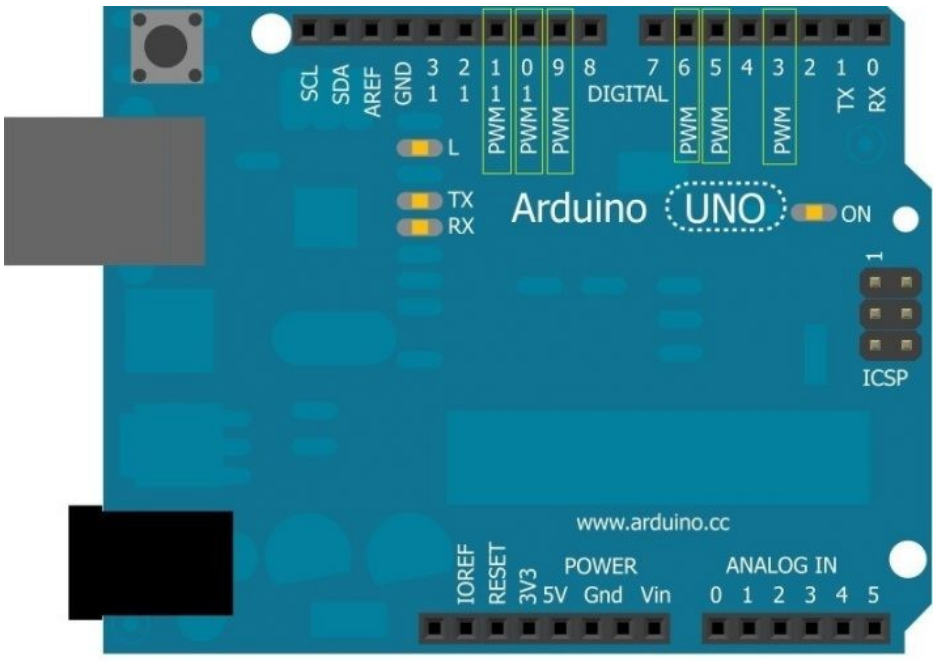
PWM pode ser usada para diversas aplicações, como por exemplo:

- controle de velocidade de motores;
- variação da luminosidade de LEDs;
- geração de sinais analógicos;
- geração de sinais de áudio.

Algumas aplicações não podem tirar proveito da técnica de PWM, como por exemplo aplicações que usem relés pela natureza do acionamento mecânico.

PWM do Arduino

A placa Arduino Uno possui pinos específicos para saídas PWM e são indicados pelo caractere ‘~’ na frente de seu número, conforme figura abaixo:



Saídas PWM na placa Arduino UNO – marcadas como PWM, na prática são indicados com ~

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

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 cycle conforme valor passado pela função. A frequência dessa onda, na maioria dos pinos é em torno 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:

*analogWrite(*pino, valor*);*

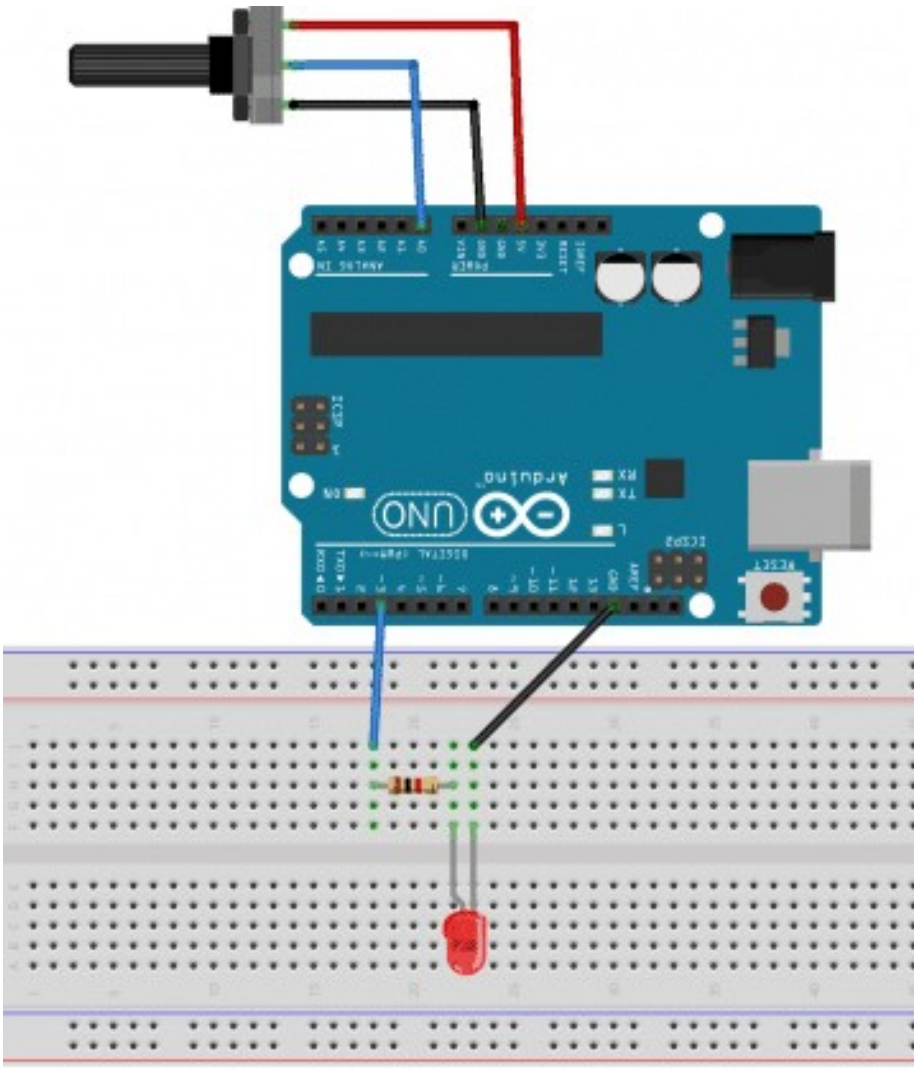
pino corresponde ao pino que será gerado o sinal PWM;

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 0 corresponde a 0% de *duty* e 255 corresponde um *duty* de 100%, sendo os outros valores proporcionalmente distribuídos.

Exemplo: Variando o brilho de um LED usando PWM do Arduino

Vamos utilizar a montagem a seguir para exemplificar o uso de um sinal PWM para variação do brilho de um LED, com a placa Arduino UNO:



Circuito para teste de PWM com Arduino

O circuito possui um LED ligado ao pino 3 (~:PWM) com um resistor (apesar do desenho, use 330Ω ou próximo) e um potenciômetro ligado à entrada analógica 0 (use linear de 10K). A proposta é controlar a intensidade do brilho do LED através da variação do valor do potenciômetro.

Vejam que o potenciômetro retorna um valor de 0 a 1023 e a variação que a função do PWM recebe um valor de 0 a 255, por isso a divisão por 4 no programa.

Programa:

```
#define LED 3   // pino do led
#define POT A0  // pino para leitura do potenciômetro

void setup()
{
  pinMode(LED, OUTPUT);  // configura pino como saída
}

void loop()
{
  int val;                // valor lido vai de 0 a 1023
  val = analogRead(POT);  // le o valor analógico
  analogWrite(LED, val/4);//aciona led com o valor analógico lido
                          //dividido por 4 para ajustar ao valor
                          //máximo (255) que pode ser atribuído a
                          //função
}
```