**Introdução ao Arduíno**

**Prática : Ajustando o brilho de um LED por PWM com Arduino**

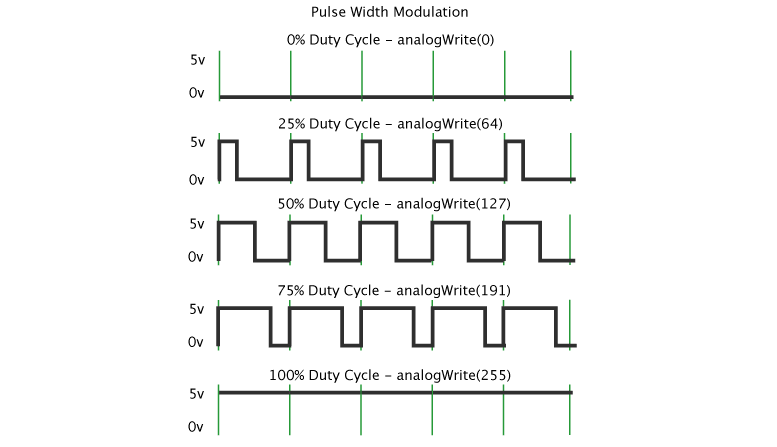
**I - Objetivo**

Nesta aula tem como objetivo controlar o brilho de um LED utilizando a técnica de PWM (Pulse Width Modulation) com Arduino.

**II – Introdução**

A modulação por largura de pulso (MLP) - mais conhecida pela sigla em [inglês](https://pt.wikipedia.org/wiki/L%C3%ADngua_inglesa) PWM (Pulse-Width Modulation) - de um sinal ou em [fontes de alimentação](https://pt.wikipedia.org/wiki/Fonte_de_alimenta%C3%A7%C3%A3o) envolve a modulação de sua [razão cíclica](https://pt.wikipedia.org/wiki/Raz%C3%A3o_c%C3%ADclica) (duty cycle) para transportar qualquer informação sobre um canal de comunicação ou controlar o valor da alimentação entregue à carga.

A modulação por largura de pulso é utilizada em várias aplicações como telecomunicações, áudio, reguladores de tensão, etc. Pode ser também utilizada para variar o valor da transferência de potência entregue a uma carga sem as perdas ocorridas normalmente devido à quedas de tensão por recursos resistivos. Em um sistema PWM, a chave de estado sólido (normalmente IGBT, MOSFET ou transistor bipolar) usada para controlar o fluxo de corrente: ora não conduzindo corrente, ora conduzindo, mas provocando uma queda de tensão muito baixa; como a potência instantânea dissipada pela chave é o produto da corrente pela tensão elétrica a um dado instante, isso significa que nenhuma potência é dissipada se a chave fosse uma chave "ideal".

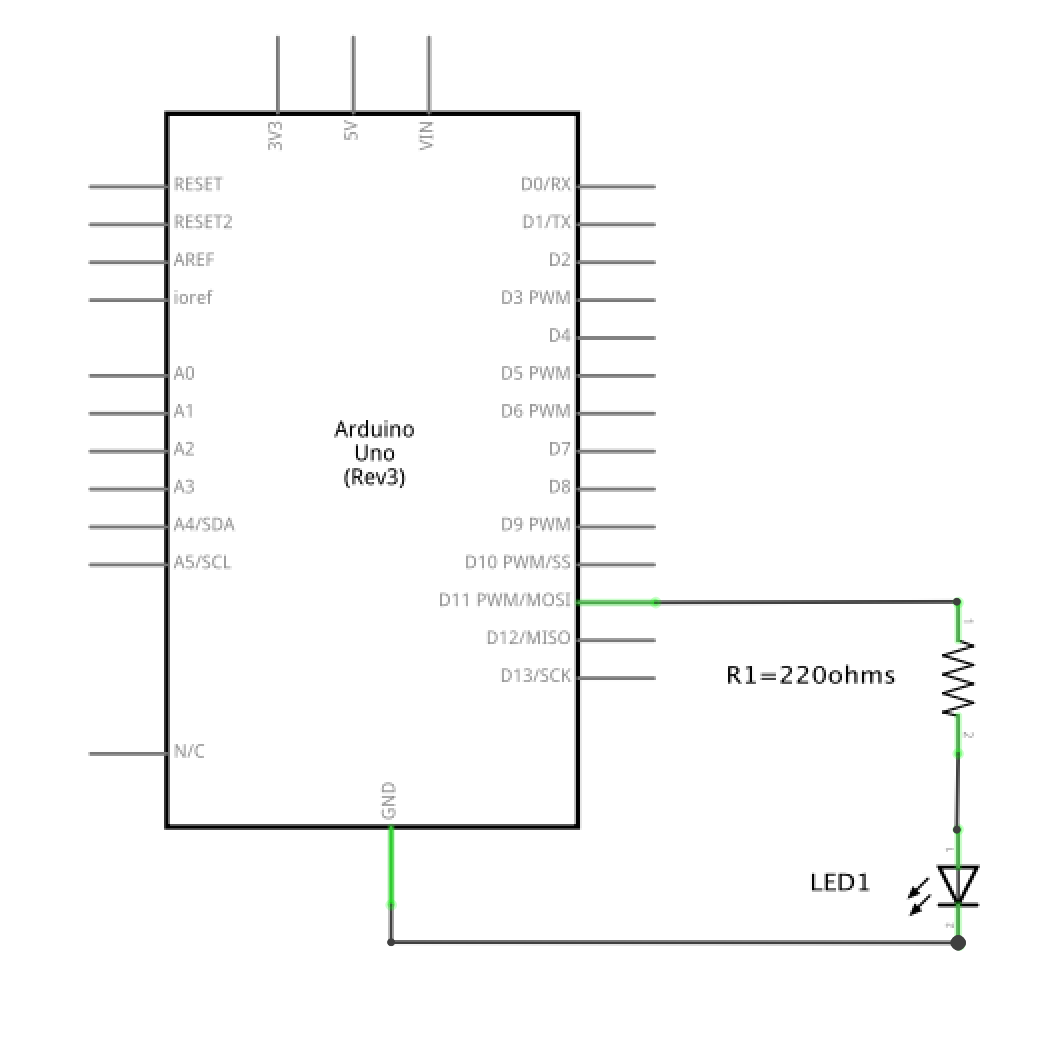


Além de controle de LEDs também é possível utilizar essa mesma técnica para controle de rotação de motores, servos e etc.

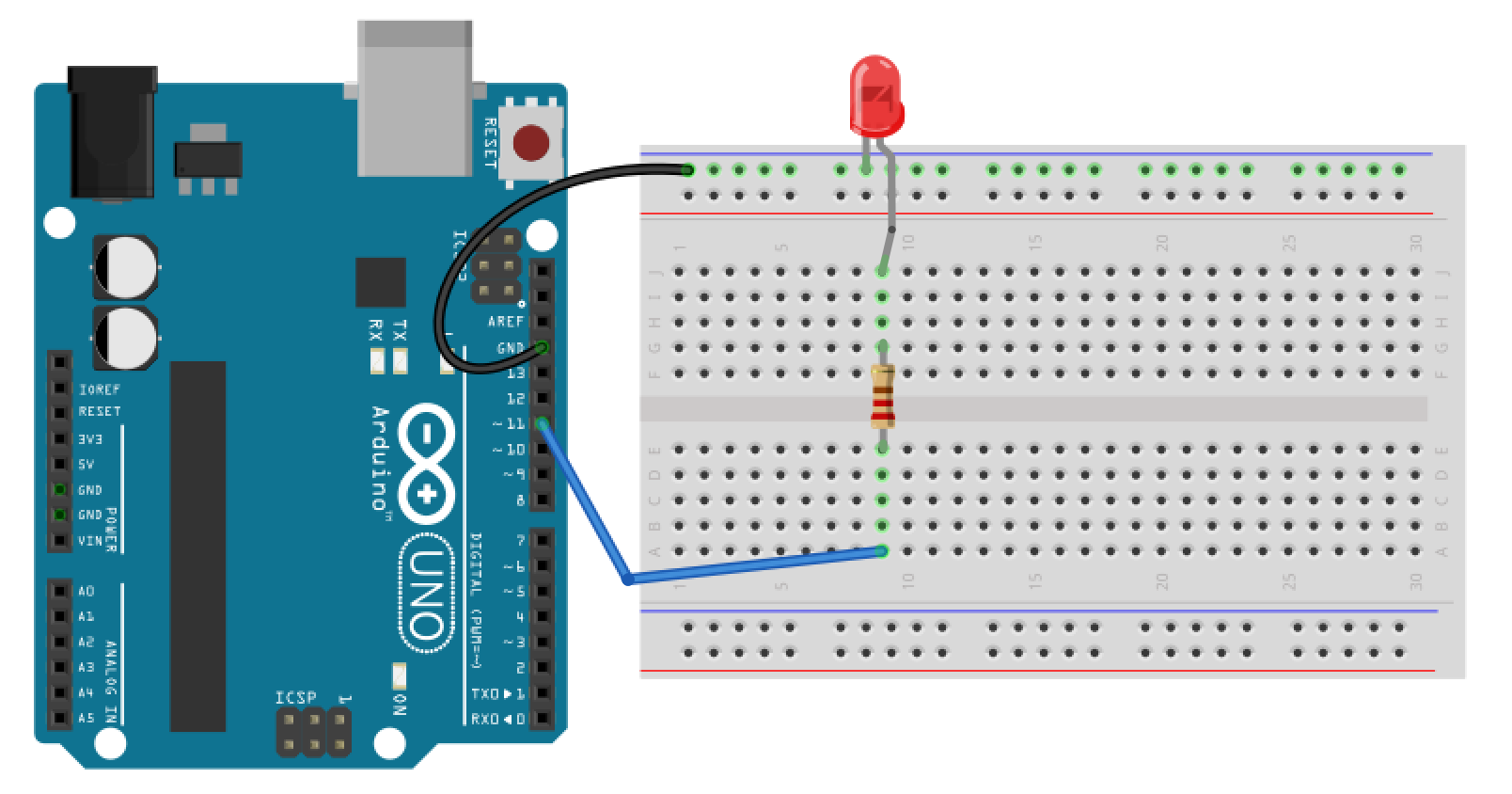
**III - Montagem**

O circuito do PWM (Pulse Width Modulation) com Arduino é bem simples e requer apenas um LED, resistor limitador de corrente, jumpers macho/macho e protoboard.

**O circuito**



## A montagem



## O programa Arduino

*// Projeto Controle de Brilho de LED - PWM*

*// Epaminondas Lage*

*int ledPin = 11;*

*float sinVal;*

*int ledVal;*

*void setup()*

*{*

*pinMode(ledPin, OUTPUT);*

*}*

*void loop()*

*{*

*for (int x=0; x<180; x++)*

*{*

*// converte graus para radianos e então obtém o valor do seno*

*sinVal = (sin(x\*(3.1412/180)));*

*ledVal = int(sinVal\*255);*

*analogWrite(ledPin, ledVal);*

*delay(50);*

*}*

*}*

## Código PWM com Arduino em detalhes

Primeiramente foram setadas as variáveis do **ledPin**, um *float* (floating point data type) para a onda seno, e um **ledVal** que será o valor inteiro a ser enviado para saída digital PWM Pin 11.

O conceito aqui é que estamos criando uma onda seno em que o brilho do LED a está seguindo em parte da onda, fazendo com que o valor do brilho seja alterado mais vagarosamente do que simplesmente brilho máximo e mínimo.

Como a função seno é somente positiva até 180 graus o laço for limita os valores entre 0 e 179, logo não haverá resultados negativos e o valor para o brilho somente poderá variar entre 0 e 255. Vale ressaltar que a função seno recebe valores apenas em radianos, logo é necessário converter de graus para radianos como mostra o código abaixo.

***sinVal = (sin(x\*(3.1412/180)));***

Os valores do seno serão multiplicados por 255 e o resultado será transformado em inteiro, trancando assim a casa decimal para que seja possível enviar para a saída digital PWM Pin 11.

***analogWrite(ledPin, ledVal);***

Este trancamento significa que você converteu o ponto flutuante float em um inteiro, eliminando assim possíveis casas decimais. Mas como posso enviar um sinal analógico por uma porta digital?

Se você olhar o Arduino verá que em alguns pinos digitais está escrito PWM, este pinos são diferentes dos outros pois são capazes de enviar um sinal PWM, que é uma técnica usada para digitalizar sinais analógicos.  Nestes pinos, o Arduino envia uma onda quadrada alternando o pino em ON/OFF rapidamente. A sequência destes ON/OFFs podem simular a variação de tensão entre 0 a 5v. Isto é feito trocando-se a quantidade de tempo que a saída fica em 1 (on) ou em 0 (off). A duração deste tempo é conhecida como pulse widht (largura de pulso).

Por exemplo, caso você envie o valor 0 pela saída digital PWM Pin 11 usando analogWrite(), o período em ON será zero, ou terá 0% de duty cycle (ciclo de trabalho). Se enviar o valor 64 (25% do valor máximo 255) o pino ficará em ON por 25% do tempo e 75% em OFF, logo duty cicle de 25%. O valor 191 terá o teu duty cicle de 75% e o valor 255 de 100%.

Compile o código e grave em seu Arduino. Note que o LED começa com um brilho bem fraco, vai aumentando seu brilho suavemente até alcançar seu brilho máximo. O olho humano não vê, mas o que está acontecendo é um processo de ligar/desligar o LED rapidamente variando assim a tensão e consequentemente o brilho.

## IV - Bibliografia

<https://www.filipeflop.com/blog/ajustando-brilho-led-pwm-com-arduino/>