02

Pulse Width Modulation

Transcrição

Agora que já conhecemos melhor sobre o **servo motor**, vamos entender sobre o **PWM** (*Pulse Width Modulation*).

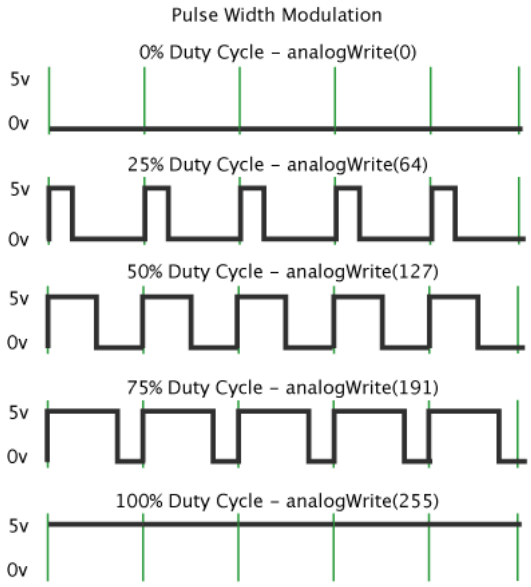
Entendendo o PWM

Dentro do nosso projeto, o servo motor será colocado em ângulos específicos, através de sinais de controle que iremos colocar nele. Mas como faremos isso?

Utilizaremos o **PWM** (*Pulse Width Modulation*, ou Modulação de Largura de Pulso), então vamos entender como ele funciona.

No mundo digital, temos o ***LOW***, que é o zero, e o ***HIGH***, que no nosso caso é o 5 volts (se estivéssemos trabalhando com o Raspberry Pi, seria 3.3 volts, por exemplo), que podemos ser interpretados como **0** (***LOW***) e **1** (***HIGH***).

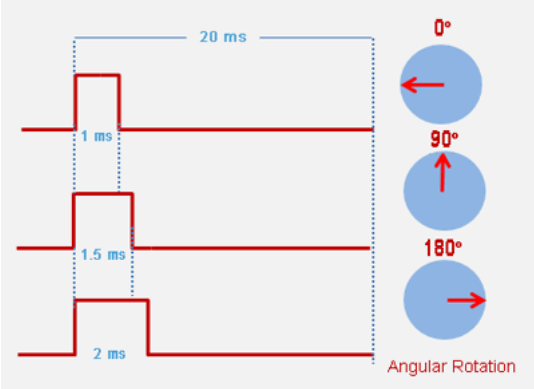
Se temos 0 e 1, como iremos traduzir isso para ângulos diferentes do motor? A resposta é que não tem como! É aí que entra o **PWM**:



A técnica consiste em variar, dentro de um período de tempo (na imagem, os períodos estão divididos pelas colunas verdes), o tempo em que o pulso se mantém em ***HIGH***. Ou seja, através da largura do pulso de uma onda quadrada, é possível o controle de potência ou velocidade.

Aplicando o PWM no servo motor

Ao olhar a especificação do nosso motor, vemos que podemos mandar um pulso de 1 a 2 milissegundos. Ao mandar um pulso de 1 milissegundo, a motor fica posicionado em 0°, se mandarmos um pulso de 1.5 milissegundos, ele fica posicionado em 90°, e se o pulso for de 2 milissegundos, o motor fica posicionado em 180°:

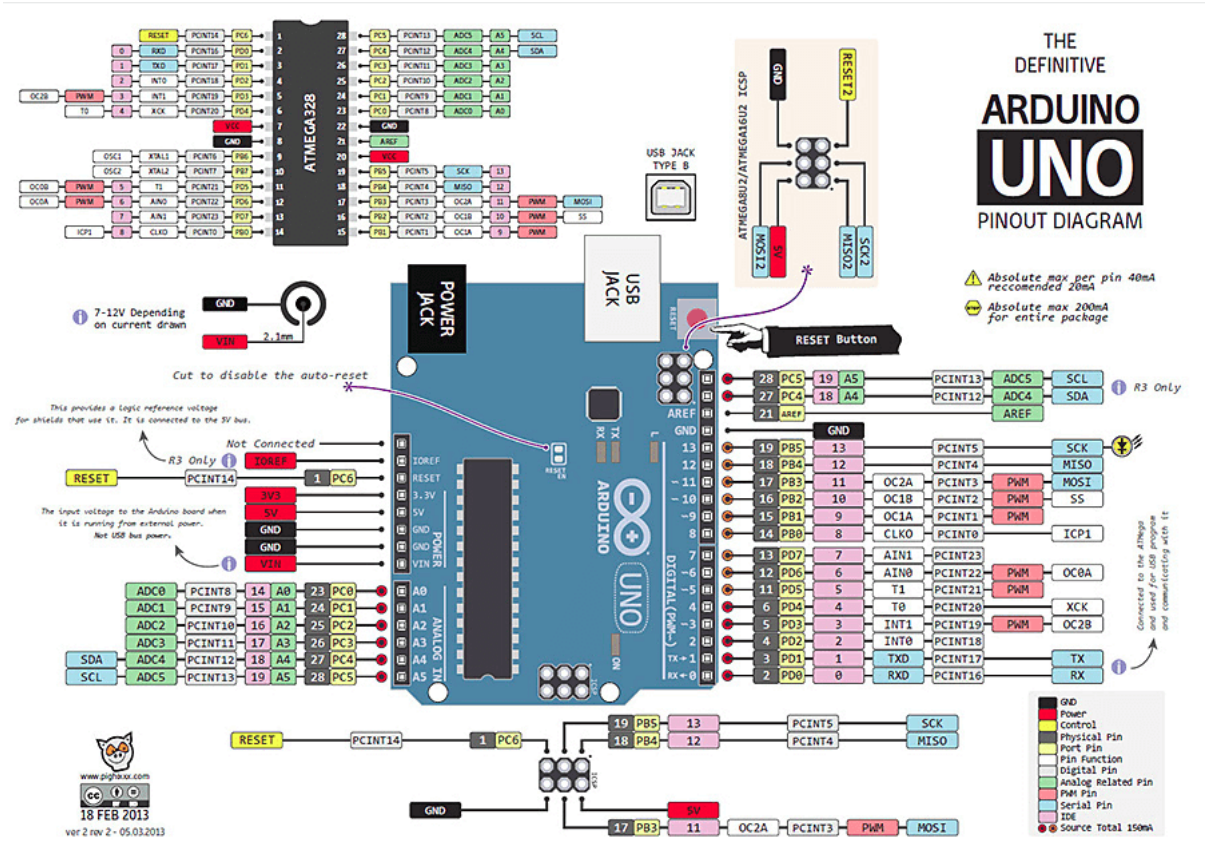


Logo, para cada valor do pulso, entre 1 e 2 milissegundos, o motor ficará em uma angulação diferente. Por exemplo, para posicionar o motor em 45°, mandamos um pulso de 1.25 milissegundos.

Essa variação do tempo, no qual colocamos o sinal em *HIGH*, será traduzida (no caso do servo motor) no ângulo em que o servo motor ficará posicionado. Mas se queremos o motor em 37°, por exemplo, precisamos aplicar uma regra de três para saber a duração do tempo que devemos mandar? Felizmente não, pois a biblioteca do servo motor no Arduino, quando colocamos o ângulo que queremos trabalhar, já sabe o tempo em que irá trabalhar. Logo, só precisamos escrever o ângulo que quisermos.

Relação de pinos PWM do Arduino

Por fim, é importante sabermos quais e quantos pinos PWM o Arduino possui:



Ele possui 6 pinos PWM, são eles: 3, 5, 6, 9, 10 e 11. E são exatamente esses pinos que utilizaremos para controlar o nosso motor.