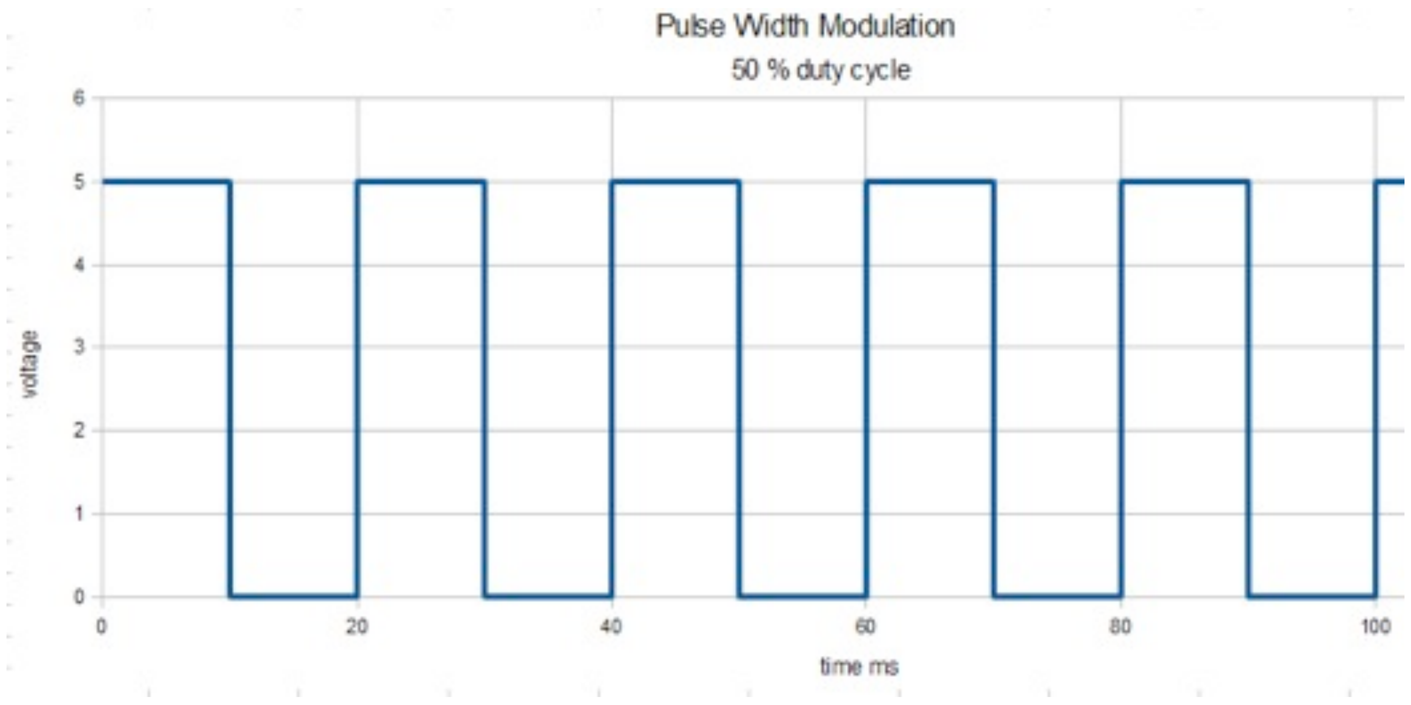
Pulse Width Modulation Basics

Pulse width modulation is a scheme that has two important uses

- Control power (at 100% efficiency)
- Send information or generate control signals

Conceptually we start with a power supply, a switch and a device to power.
Diagram:
Turn the switch on for some time, say .10 seconds (100 ms), then off for the same time.
We get:

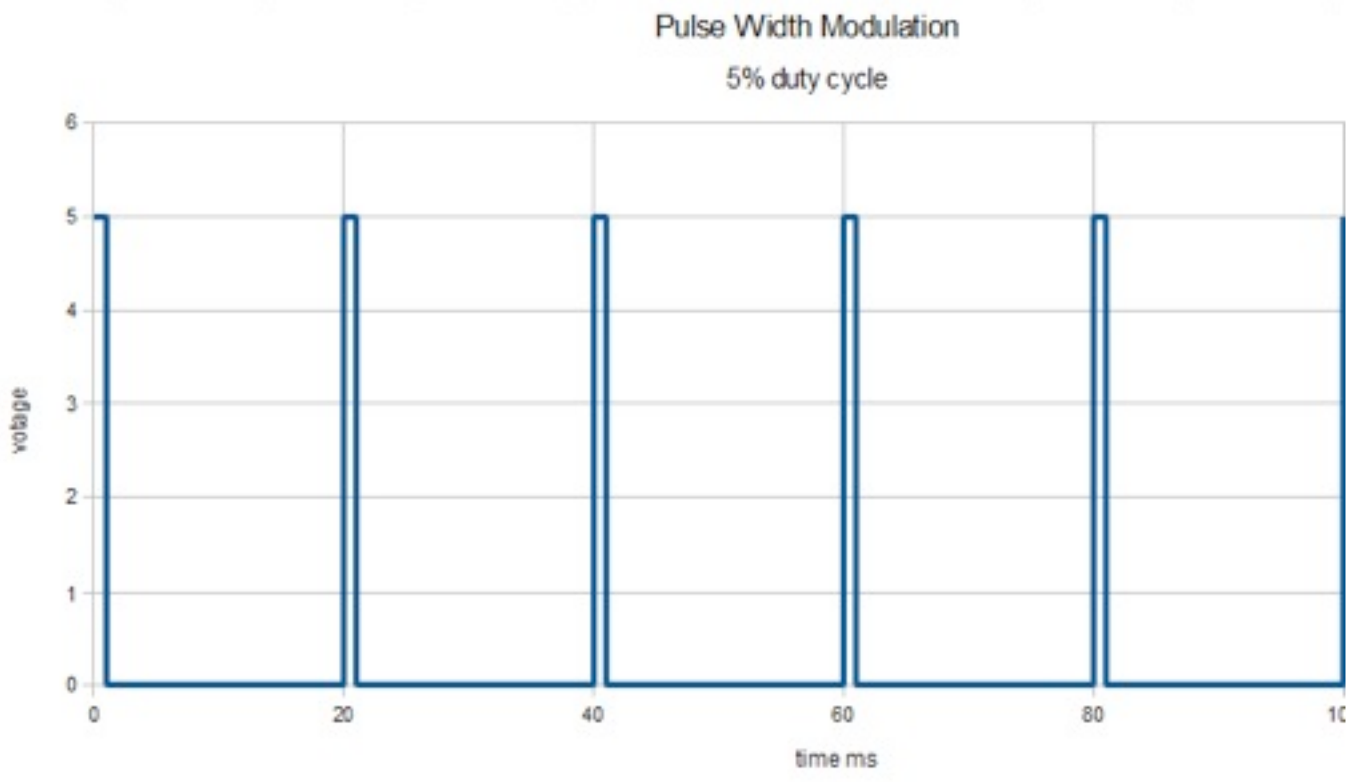


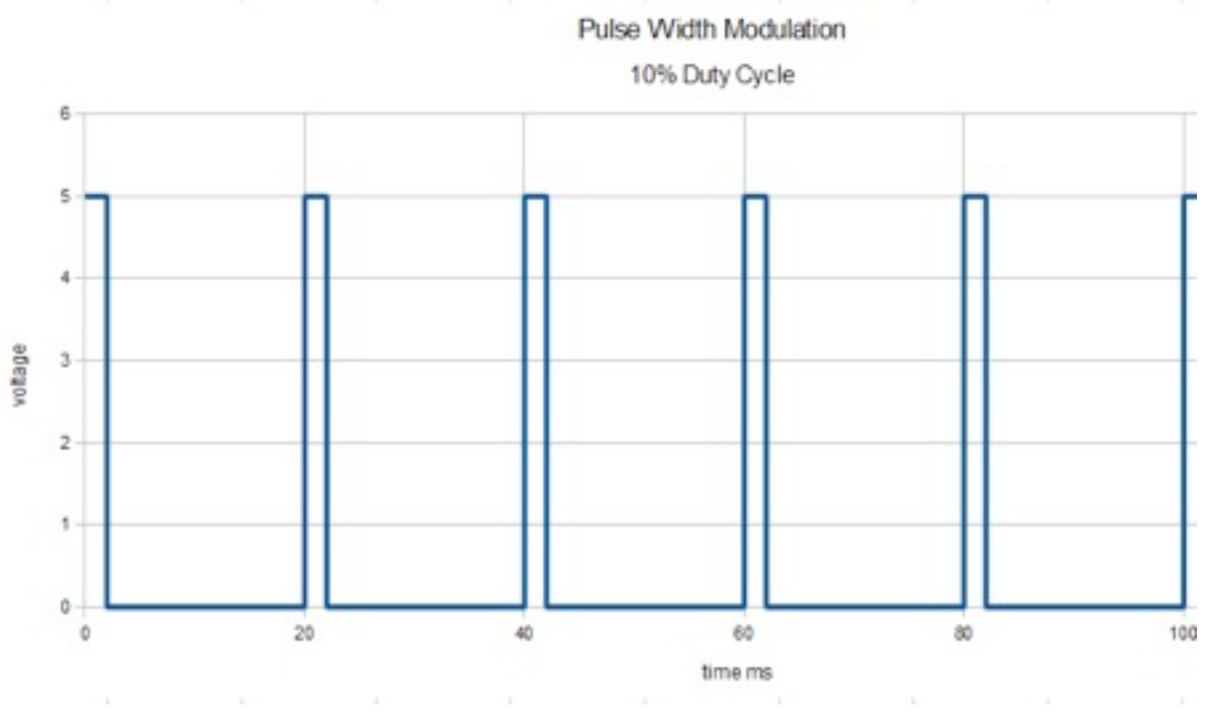
It is not hard to see that the power is about 50 percent of full power or on all the time. The whole wave has a period (the time to repeat) of .1 seconds, and a frequency of 10 waves per second or 10 Hz

The basic math of this is

- f frequency = waves per second
- p period = seconds per wave = 1/f
- duty cycle = time on/period or time on/(time on + time off)
  (note that this has nothing particular to do with either the frequency or period that is for any period we can have any duty cycle)

We can choose any duty cycle we want independentlY of the frequency or period. We often choose a frequency high enough so that we do not notice the pulsations of signal, just the overall effect of the reduction of power.
The next part of the game is to change the time on and the time off, but in such a way that the total adds up to the same number, this number is the period of the wave, so the period and the frequency of the wave stays the same, but the duty cycle varies:
The following diagrams show 5 and 10 percent duty cycle.





for Servos

For servos pulse width modulation does not control power but instead sends information. In this case the frequency is 50 hz corresponding to a period of 20 ms, the pulse is varied from 1 ms to 2 ms. The pulse width represents the amount of rotation we want from the servo. (The duty cycle is from 2% to 4% for a servo.)

Arduino and Pulse Width Modulation

There are many ways to do pulse width modulation with an arduino. These fall into 2 classes, hardware assisted and straight software.

Hardware
The arduino hardware includes a unit that will produce pulse width with essentially one command, and keep doing it with no attention from the program. However this is easiest if the frequency is kept to about 500 Hz and with 256 divisions of the duty cycle (in about .004 percent steps). It is not really useful for servos as the frequency is not right and the control over the 1 to 2 ms range is not very high.

Software
Software modulation keeps the controller busy but allows pretty much any frequency and duty cycle within a few micro seconds. It can fail if the controller gets too busy, and it can be hard for the controller to do other things at the "same time"