Pulse Width Modulation (PWM) – Controlling Analog Output with Digital Signals

What is PWM?

Pulse Width Modulation (PWM) is a technique used to simulate an **analog** output using **digital** signals. It achieves this by rapidly switching a digital signal **on and off** at a high frequency. By adjusting the proportion of time the signal is **on** versus **off** (called the **duty cycle**), PWM can effectively control the **average voltage** applied to a device, such as an LED, motor, or speaker.

How PWM Works

A **PWM signal** is a square wave that alternates between **high (on)** and **low (off)** states. The key parameters of a PWM signal are:

- **Period (T)**: The total time of one complete on-off cycle.
- **Frequency (f)**: The number of cycles per second (measured in **Hertz, Hz**).
- **Duty Cycle (%)**: The percentage of time the signal is **on** in each period.

The **duty cycle** determines the average output voltage:

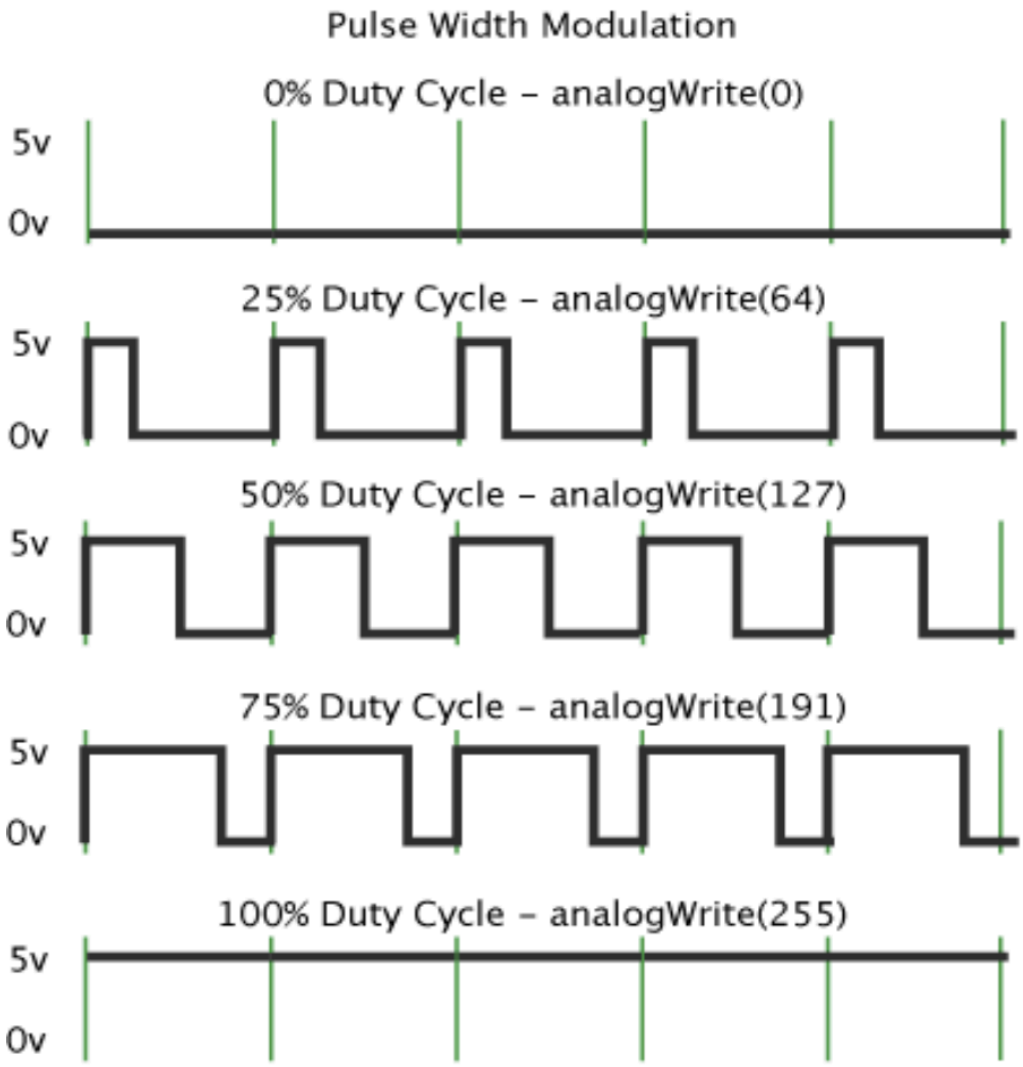
$$V_{avg} = V_{max} * \frac{Duty\ Cycle}{100}$$

where:

- V_{max} is the full supply voltage (e.g., 5V for Arduino).
- A **100% duty cycle** means the signal is always on (full voltage output).
- A **50% duty cycle** means the signal is on for half the time, simulating half the voltage.
- A **0% duty cycle** means the signal is always off (zero voltage output).

For example:

- **analogWrite(255)** → 100% duty cycle (always on, full brightness LED).
- **analogWrite(127)** → 50% duty cycle (half brightness LED).
- **analogWrite(0)** → 0% duty cycle (always off, LED off).



PWM in Arduino (Fading LED Example)

The **Fading** example in Arduino demonstrates PWM by gradually increasing and decreasing the brightness of an **LED**. It is available in:

File → Sketchbook → Examples → Analog → Fading

Arduino uses PWM to simulate an **analog output voltage** by changing the duty cycle of a digital signal. The **analogWrite()** function in Arduino allows setting PWM values between **0 and 255**, where:

- **0** is always off (0% duty cycle).
- **255** is always on (100% duty cycle).
- **127** is on for half the time (50% duty cycle).

PWM Frequency and Applications

The frequency of PWM determines how fast the signal switches on and off. In Arduino, PWM typically runs at **~500Hz**, meaning each period lasts **2 milliseconds**. At this speed, the human eye perceives the LED brightness as a smooth fade rather than flickering.

Common Applications of PWM

- **LED Brightness Control** (simulating different voltage levels).
- **Motor Speed Control** (controlling power delivered to motors).
- **Audio Signal Generation** (modulating sound frequencies).
- **Power Regulation** (switching regulators use PWM for efficient power conversion).

By using **PWM**, digital microcontrollers can effectively control devices that typically require an **analog** voltage, making it a powerful tool for electronics and embedded systems.

Arduino Tutorial 8: Understanding Pulse Width Modulation (PWM) and the Arduino Analog Write Command