PWM Pin

PWM (Pulse Width Modulation) Pins on Arduino

PWM (Pulse Width Modulation) allows you to control the intensity of LEDs, motor speed, and other analog-like outputs using digital pins.

PWM Pins on Arduino Uno:

- The Arduino Uno has **6 PWM pins**: **3, 5, 6, 9, 10, and 11**.
- These pins can be used with analogWrite(pin, value);, where **value** ranges from **0 (0% duty cycle)** to **255 (100% duty cycle)**.

Key Concepts of PWM:

1. Signal Modulation:

- PWM involves switching the pin on and off rapidly, which results in a square wave signal.

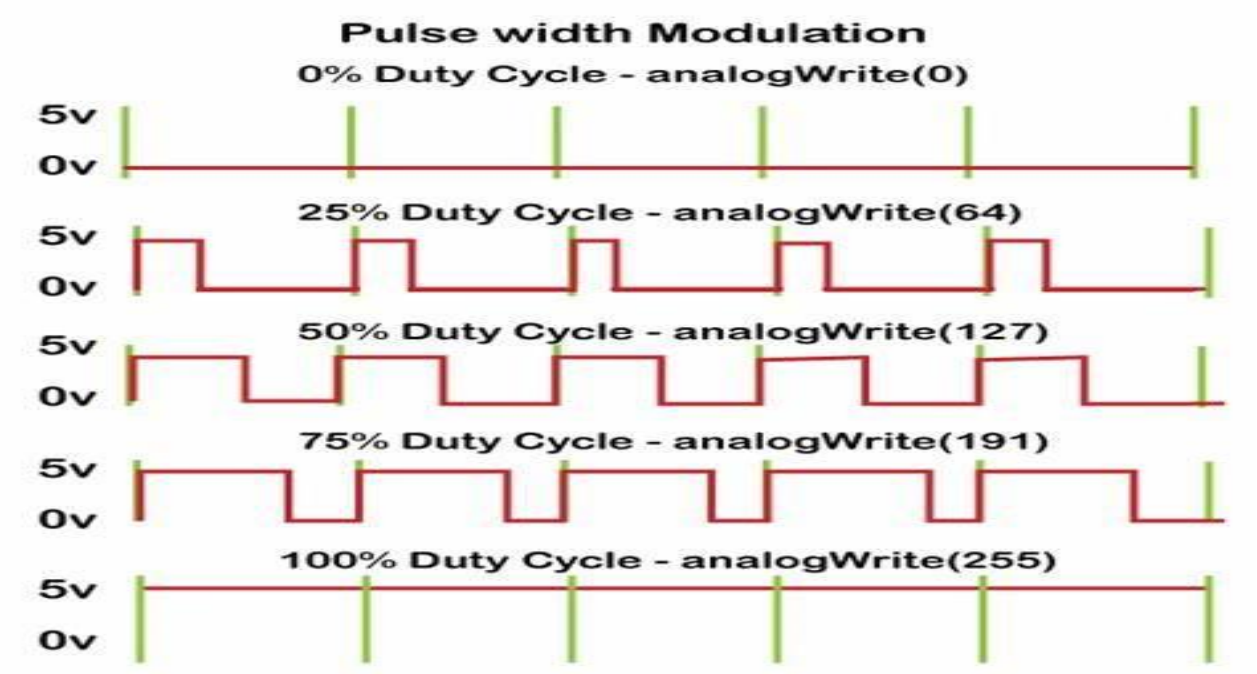
- The duty cycle of the signal (the proportion of time the signal is "on") determines the average power delivered to the device.

2. Duty Cycle:

- A 100% duty cycle means the signal is always on, delivering full power.

- A 0% duty cycle means the signal is always off, delivering no power.

- Any value in between modulates the power linearly.



3. Arduino PWM Pins:

- Not all digital pins on an Arduino can perform PWM. Typically, the pins marked with a tilde (~) support PWM.

4. Using PWM in Arduino:

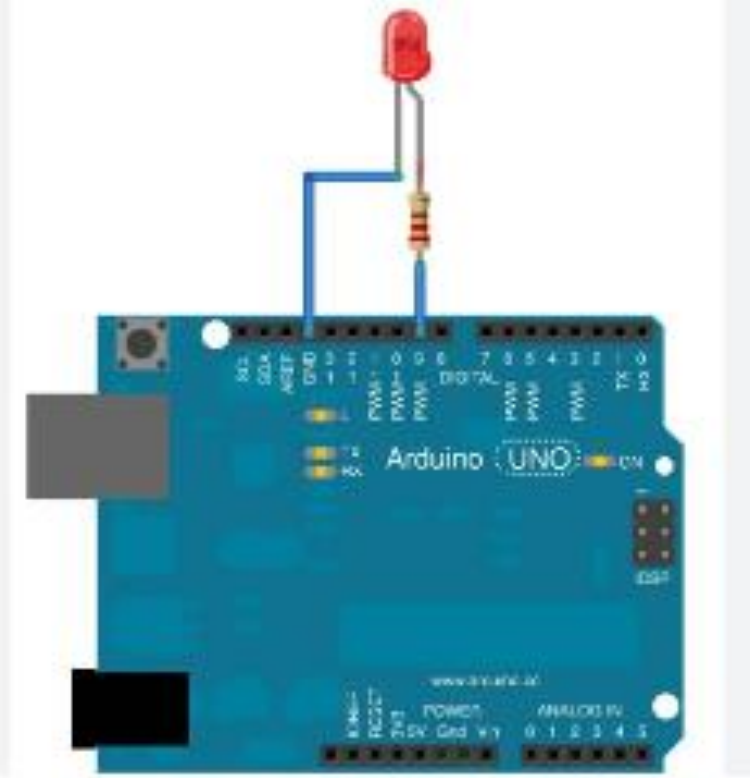
- The `analogWrite(pin, value)` function is used, where `value` ranges from 0 to 255, corresponding to 0% to 100% duty cycle.

Example Code:

Here's an example of using PWM to control the brightness of an LED:

```
int ledPin = 9; // PWM pin connected to LED

void setup() {

pinMode(ledPin, OUTPUT);

}

void loop() {

for (int brightness = 0; brightness <= 255; brightness++) {

analogWrite(ledPin, brightness); // Set the brightness

delay(10); // Wait for 10 milliseconds

}

for (int brightness = 255; brightness >= 0; brightness--) {

analogWrite(ledPin, brightness); // Set the brightness

delay(10); // Wait for 10 milliseconds

}

}
```

Circuit Diagram Representation



Application:

PWM is widely used in applications where variable power is required, such as dimming LEDs, controlling motor speed, and adjusting signal levels.

For reference, here is a table image showing the PWM pins on an Arduino board: