PWM (Pulse Width Modulation)

- PWM enable to vary duty cycle of clock (0 to 100%) and effectively power delivered to the load (device).
- duty cycle = (Ton / (Ton + Toff)) * 100
- It works like DAC, if clock frequency is too low.
- There are few devices whose behaviour depends on power supplied to them like DC motor.
    - DC motor speed depends on power supplied to that.
    - DC motor direction depends on polarity of the voltage.

Watchdog Timer

- It detects timeout in any activity and reset the processor to handle that problem.
- Resetting system is helpful many times to recover from the problem.
- WDT is a downcounter. It can be programmed for delay of few ms/sec. When counter reach 0, it resets the processor. If counter is disabled before that time, processor continue to function normally.
- Usage:
    - step1: Decide the time duration required for any activity/task.
    - step2: Calculate WDT count for that duration (based on clock).
    - step3: Load the count into the WDT and start the timer.
    - step4: Initiate the activity/task.
    - step5: At the end of task, disable the WDT.
        - If task is completed within duration, WDT will be disabled.
        - If task is not completed within duratin, WDT will raise interrupt and reset the processor.
    - step6: Upon reset, check if reset is due to WDT and take necessary action.
- Typical application

```
main() {
    check if reset due to WDT & take action accordingly
    while(1) {
        init WDT
        // do task
        stop WDT
    }
}
```

AVR WDT

- MCUCSR
- WDTCR

```
int main() {
    if(MCUCSR & BV(WDRF)) {
        // reset is due to WDT
        // do error handling
    }

    while(1) {
        // init WDT to wait for 2sec
        WDTCR = BV(WDE) | BV(WDP2) | BV(WDP1) | BV(WDP0);

        // do the task -- time = 2sec

        // stop WDT
        WDTCR = BV(WDTOE) | BV(WDE);
        WDTCR = 0x00;
    }
}
```