Experiment 1 - DC Motor Control

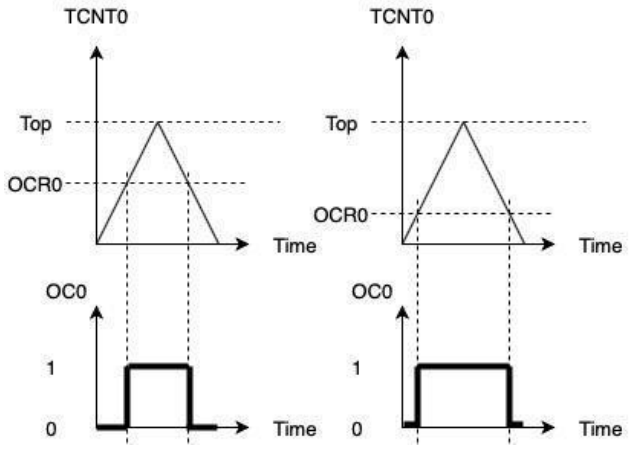
Objective:

Using PWM (Pulse Width Modulation) to control a device is a common practice in embedded systems; for example, you can use it to control the light intensity of a LED or control the speed of a DC motor.
In this experiment, we will explain how to get a PWM from the AVR Atmega16 and we shall apply the output PWM to a small DC motor to vary its speed.

Introduction:

In order to get the PWM from AVR, we need to use the **timer/counter** module of the AVR. This module can be used in several modes to generate different PWM signals of different characteristics; here we shall explain how to use the counter in the "Phase Correct PWM" mode. Atmega16 has 3 timer/counters and we are using **timer/counter 0**.

The phase correct mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x00) to TOP and then from TOP to BOTTOM. The Output pin **(OC0)** is set when the counter reaches a certain value called the "**Compare value**" while up counting, and is cleared when the counter reaches the same value while down counting. This compare value is set by the software in a register called **OCR0** (Output Compare Register), while the value of the counter itself is contained in a register called **TCNT0**. When the value of TCNT0 matches the OCR0, it's called a Compare Match. The below timing diagram explains the operation.



Therefore the duty cycle can be calculated as:

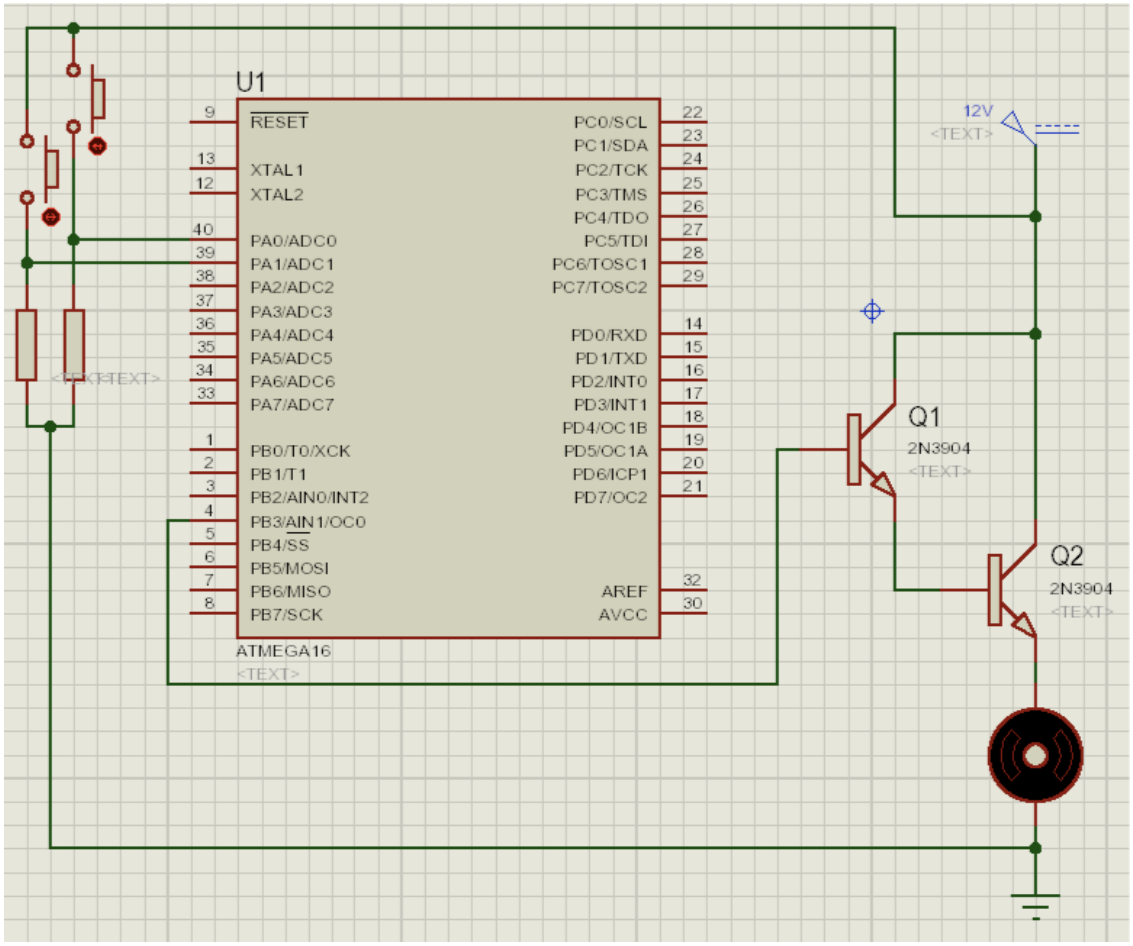
$$Duty\ Cycle = \frac{255 - OCR0}{255}$$

Experiment Procedure:

Using proteus simulator

- Connect two push buttons on **PINA0 and PINA1**, where these buttons will control the motor speed.
    - **PINA0** will increment the OCR0 by delta value (for example 5)
    - **PINA1** will decrement the OCR0 with the same value.
- Connect the DC motor to **OC0** pin for PWM generation through transistor.
    - Microcontrollers are not able to drive sufficient current to DC motors directly, then we will use 2 transistors **2N3904** as an electronic switch.

Circuit Diagram:



Code:

```
#define get_bit(reg,bitnum) ((reg & (1<<bitnum))>>bitnum)

void main() {
	float duty_cycle = 0;


	DDRA=0b11111100; // Set the first 2 pins of PORTA as inputs to read push buttons
	DDRB=0b11111111; // Set Pin3 (OC0) in Port B as output


	TCCR0=0b01110101; // Configure TCCR0 as explained in the article
	OCR0=255; // Set OCR0 to 255 for initial duty cycle = 0 and the motor is not rotating

	while(1) {

		if ((get_bit(PINA,0) == 1)) {
			// Place your code here (Increase the duty cycle)
		}

		if ((get_bit(PINA,1) == 1)) {
			// Place your code here (Decrease the duty cycle)
		}

		// Place your code here (Apply the change in duty cycle to OCR0)
		delay_ms(100);
	}

}
```

Note: don't forget to take care of the overflow that will occur due to continuous increment or decrement without boundary checking.

Lab Deliverables:

DC Motor control system:

AVR based system to control the speed of a DC motor using PWM signal. The system will have two push buttons to control motor speed up or down.

Extended Features (Mandatory):

Add push buttons to control the motor rotation direction. (Hint: Use H-Bridge)