CI - Lap 001 - DC Motor Control

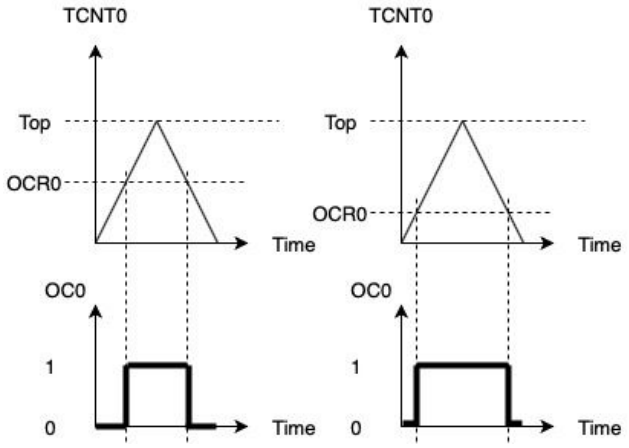
Lab Target:

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 lab, 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.

Theory:

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: $Duty\ Cycle = \frac{(0xFFFF - OCR1)*2}{0xFFFF*2}$.

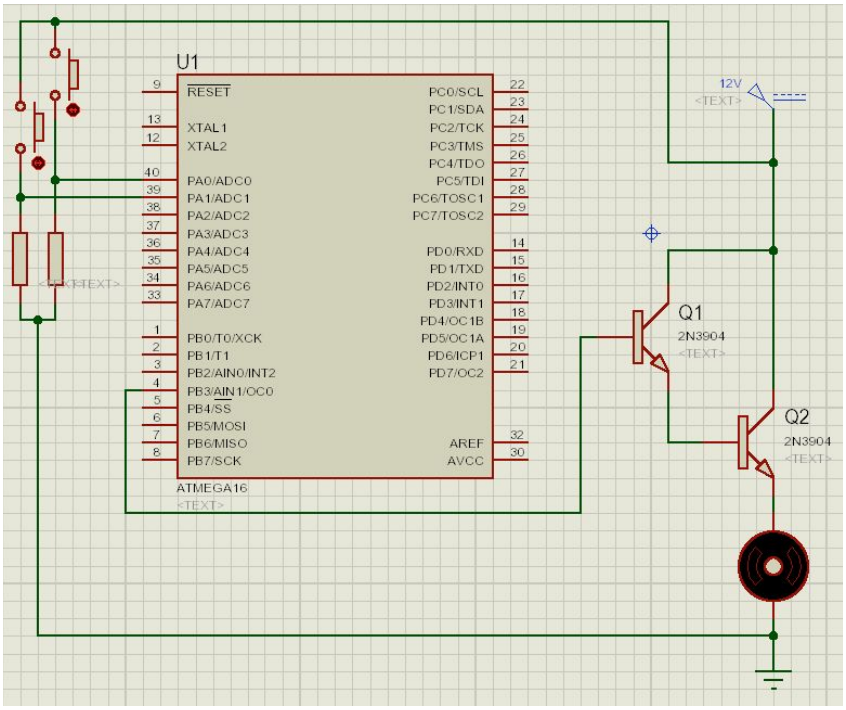
Required Registers with Values:

```
TCCR0 = 0b01110101; //Configure TCCR0 as explained in the article
TCCR0 = 0b00000000; //Disable all interrupts.
TCCR0 = 255; // Set OCR0 to 255 so that the duty cycle is initially 0 and the motor is not rotating
```

Experiment:

Using proteus simulator connect two push buttons on **PINA0-1**, where these buttons will control the motor speed. **PINA0** will increment the OCR1 by delta value (5) while **PINA1** will decrement the OCR1 with the same value. Regarding the DC motor will be connected on **OC1** for PWM generation. Since the µC will not be able to drive the motor directly we will use 2 transistors **2N3904** as an electronic switch.

Circuit Diagram:



Code:

```
#include <avr/io.h>

#define get_bit(reg,bitnum) ((reg & (1<<bitnum))>>bitnum)
#define DELTA 5
int main(void)

{
   DDRA=0b11111000; // set the first 3 pins of PORTD to be inputs to read from the push buttons
   DDRB=0b11111111; // ensure that Pin3 in POrt B is output as this is the OC0 pin that wll produce the PWM.
   PORTD=0b00000000; // Initialize PORTD to zeros
   TCCR0=0b01110101; //Configure TCCR0 as explained in the article
   TIMSK=0b00000000;
   OCR0=255; // Set OCR0 to 255 so that the duty cycle is initially 0 and the motor is not rotating

   volatile float duty_cycle = 0;

   while(1)
   {
      if ((get_bit(PINA,0)==1)) {
         duty_cycle -= 0.1;
      }

      if ((get_bit(PINA,1)==1)) {
         duty_cycle += 0.1;
      }

      OCR0 = (uint8_t)(255 * (1 - duty_cycle));
      DDRC  = 255;
      PORTC = OCR0;
   }
   return 0;
}
```

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