Student Name : W.K.G.K Jayawardana

Student Number : EC/2021/006

Partner Name : S.M.A.D.R Pabasarani

Partner Number : EC/2021/007

Date Performed : 23/05/2025

Date of Submission : 23/05/2025

Experiment No: 08

BECS 31421

**PULSE WIDTH MODULATION (PWM) WITH MICROCONTROLLER**

**DISCUSSION**

In this experiment, we successfully implemented Pulse Width Modulation (PWM) using the PIC16F628A microcontroller to control the brightness of an LED. By varying the duty cycle of the PWM signal, we could effectively simulate different brightness levels of the LED, demonstrating how PWM can control power delivered to analog devices using a digital signal.

The PWM was generated using the PWM1\_Init() and PWM1\_Set\_Duty() functions provided by MikroC PRO, which greatly simplified the coding process. By cycling the duty cycle from 0 to 255 and back in steps of 5, a gradual fade-in and fade-out effect was achieved, visible both in simulation and physical hardware. This confirmed the linear relationship between duty cycle and perceived brightness due to the human eye's persistence of vision.

One important observation was the significance of setting up the configuration bits correctly. Specifically, disabling the comparator module (CMCON = 0x07) was necessary to use the PWM pin properly. Incorrect setup here would result in the PWM signal not being generated or delivered correctly, causing the LED to stay off or behave erratically.

**SOURCE CODE**

// Step 1: Declare the main function

void main() {

// Step 2: Configuration settings: output and initialization

CMCON = 0x07; // Disable comparators (set all to digital)

TRISB = 0x00; // Set all port B pins as output

PORTB = 0x00; // Initialize PORTB to 0

// Step 3: PWM initialization and start

PWM1\_Init(5000); // Initialize PWM1 module with the frequency of 5kHz

PWM1\_Start(); // Start the PWM1 module operation

// Step 4: Define the infinite loop

while(1) { // Infinite loop

// Step 5: PWM duty cycle variation

unsigned short duty\_cycle; // Declare an unsigned short variable ‘duty\_cycle’

// Step 5.1: Increasing duty cycle loop

for (duty\_cycle = 0; duty\_cycle <= 255; duty\_cycle += 5) {

PWM1\_Set\_Duty(duty\_cycle); // Set the duty cycle

Delay\_ms(50); // Delay to observe the change in LED brightness

}

// Step 5.2: Decreasing duty cycle loop

for (duty\_cycle = 255; duty\_cycle >= 5; duty\_cycle -= 5) {

PWM1\_Set\_Duty(duty\_cycle); // Set the duty cycle

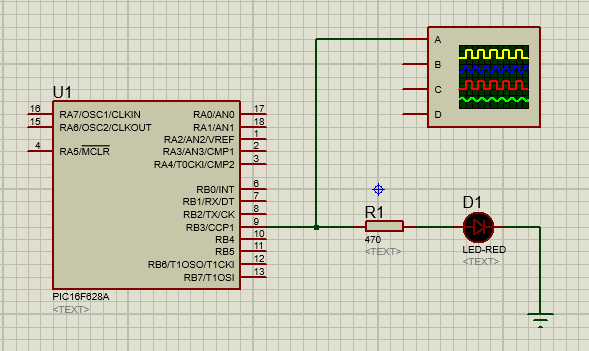
Delay\_ms(50); // Delay to observe the change in LED brightness

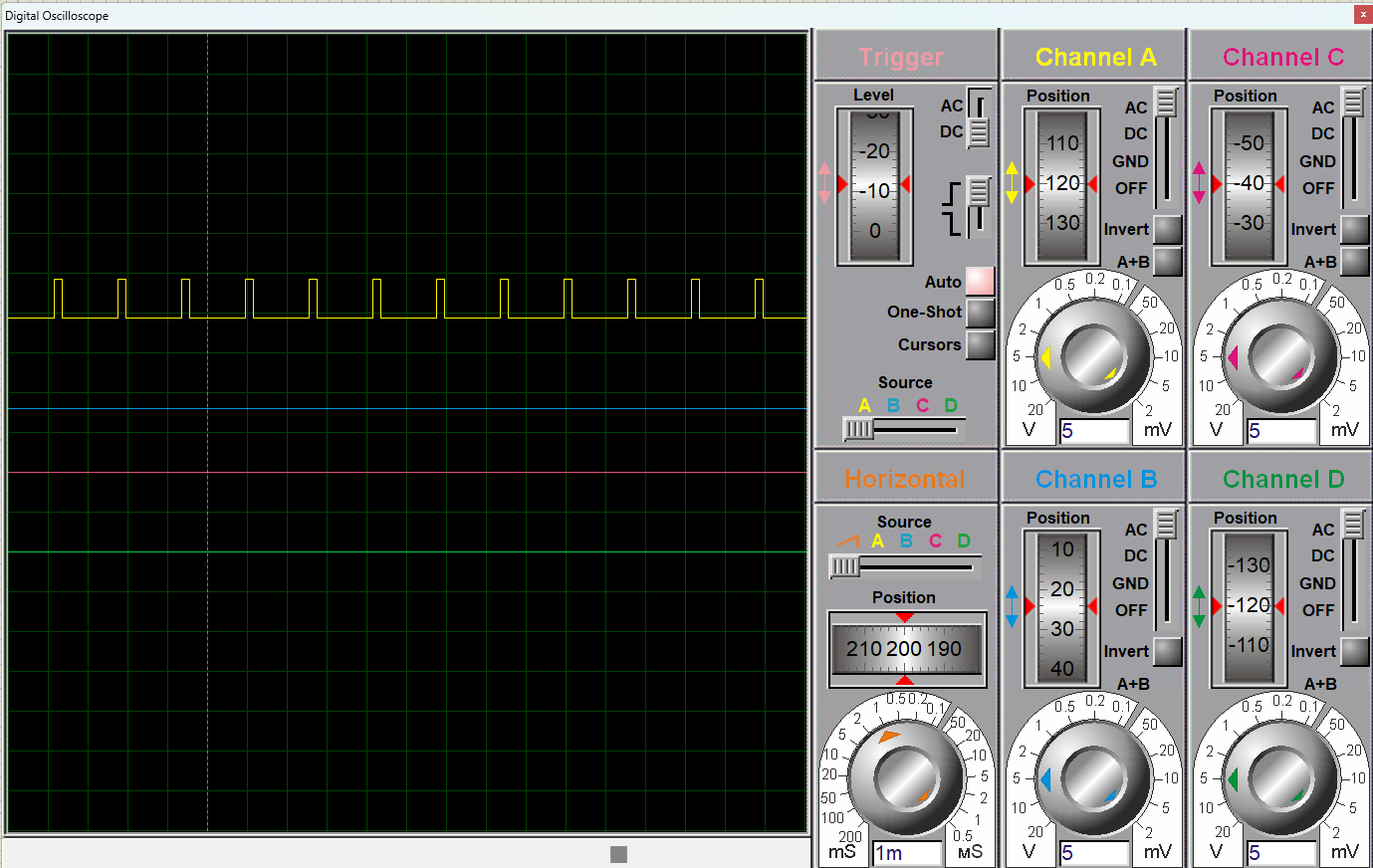
}

}

}

**SIMULATION SCREENSHOTS**



****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**