

Lab 9

Generating a PWM Waveform



Spring 2025

Submitted by: Mohsin Sajjad

Registration No: 22pwsce2149

Class Section: A

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

A handwritten signature in black ink that reads "Mohsin Sajjad".

Student Signature: _____

Submitted to:

Engr. Faheem Jan

Month Day, Year (11 05, 2025)

Department of Computer Systems Engineering

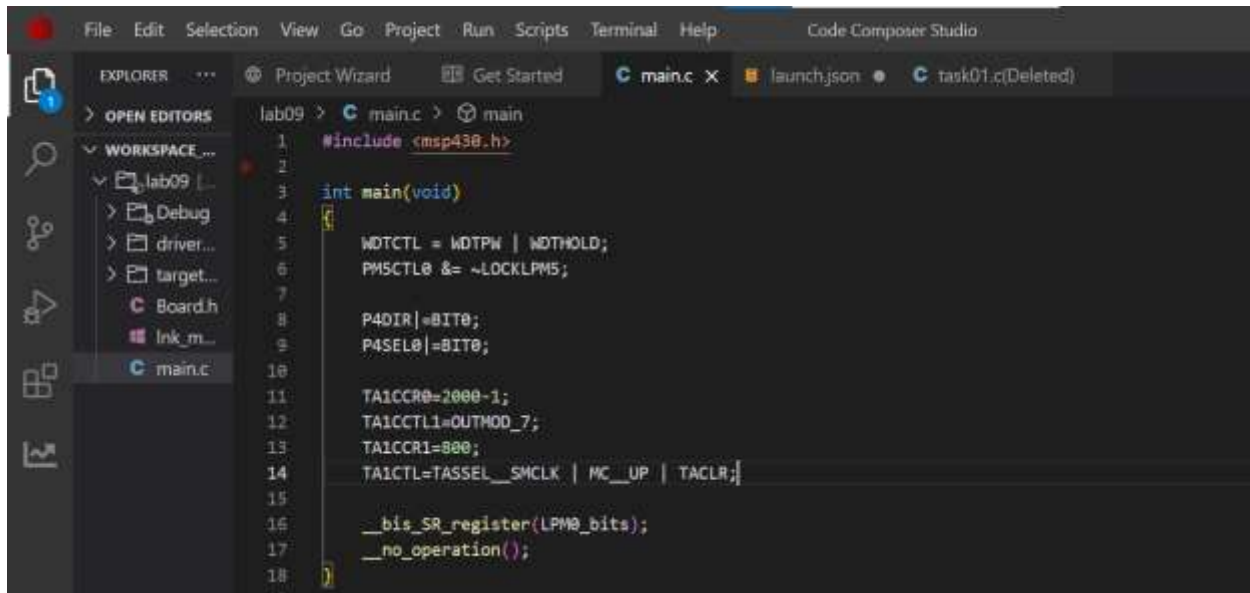
University of Engineering and Technology, Peshawar

Generating a PWM Waveform

TASKS:

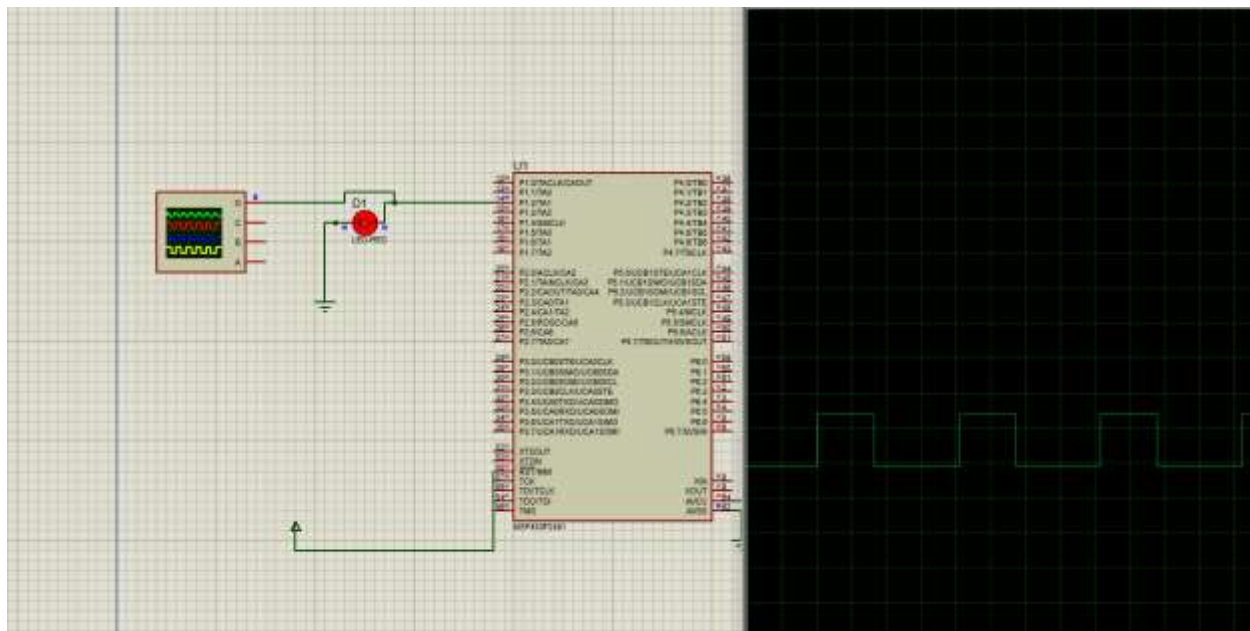
- 1) Generate a signal of 500Hz with 40% duty cycle.

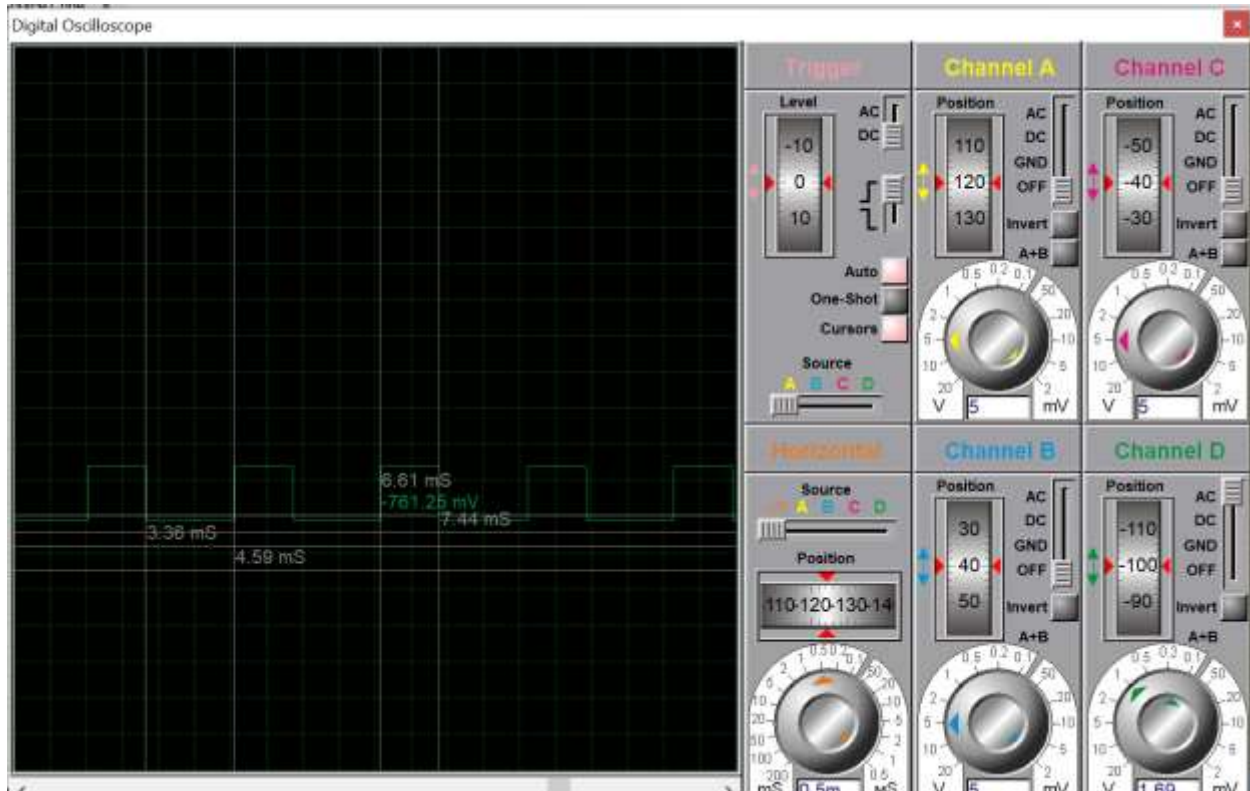
CODE:



```
File Edit Selection View Go Project Run Scripts Terminal Help Code Composer Studio
lab09 > C main.c > main
1 #include <msp430.h>
2
3 int main(void)
4 {
5     WDTCTL = WDTPW | WDTHOLD;
6     PM5CTL0 &= ~LOCKLPM5;
7
8     P4DIR |= BIT0;
9     P4SEL0 |= BIT0;
10
11     TA1CCR0=2000-1;
12     TA1CCTL1=OUTMOD_7;
13     TA1CCR1=800;
14     TA1CTL=TASSEL_SMCLK | MC_UP | TACLK;
15
16     __bis_SR_register(LPM0_bits);
17     __no_operation();
18 }
```

OUTPUT:





Task 2:

Generate a signal of 600Hz with 60% duty cycle on P1.3 Hint: use timer.

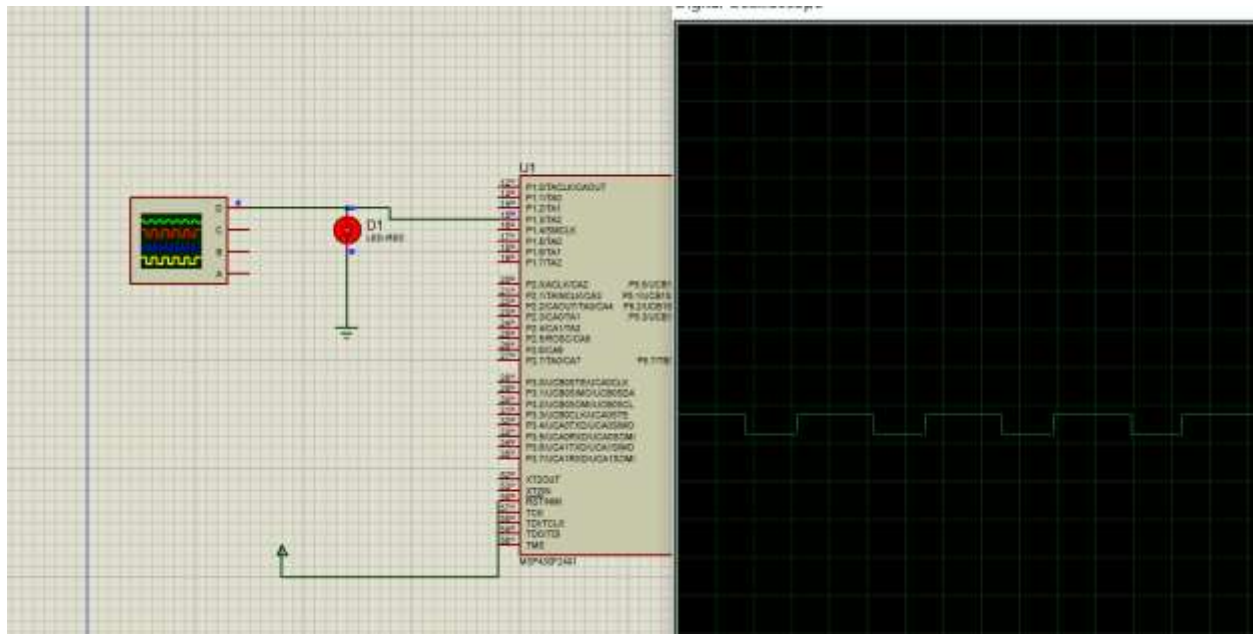
CODE:

```

File Edit Selection View Go Project Run Scripts Terminal Help Code Composer Studio
EXPLORER: ... Project Wizard Get Started main.c X launch.json task01.c(Deleted)
> OPEN EDITORS lab09 > C main.c > ...
WORKSPACE ...
  > lab09 [ ... ]
    > Debug
    > driver...
    > target...
    C Board.h
    Ink_m...
    C main.c
1  #include <msp430.h>
2
3  int main(void)
4  {
5      WDTCTL = WDTPW | WDTHOLD;
6      PMSCTL0 &= ~LOCKLPM5;
7
8      P4DIR |= BIT0;
9      P4SEL0 |= BIT0;
10
11     TA1CCR0=1660-1;
12     TA1CCTL1=OUTMOD_7;
13     TA1CCR1=996;
14     TA1CTL=TASSEL_SMCLK | MC_UP | TACLK;
15
16     __bis_SR_register(LPM0_bits);
17     __no_operation();
18 }

```

OUTPUT:

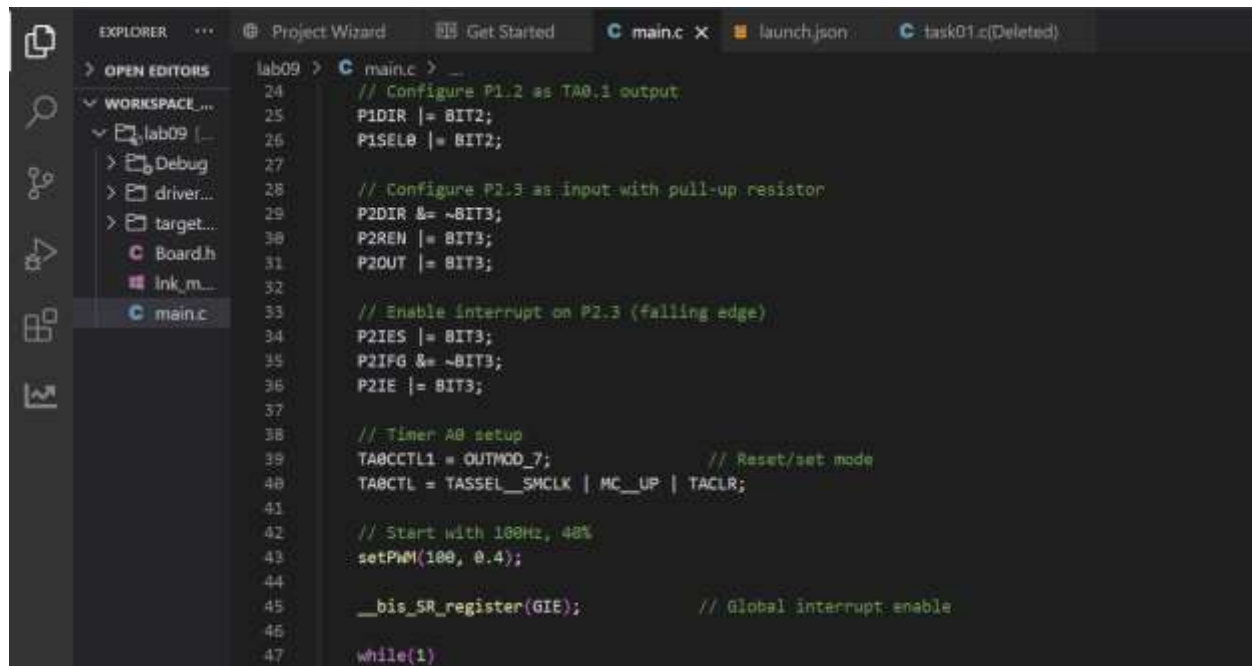


TASK 03:

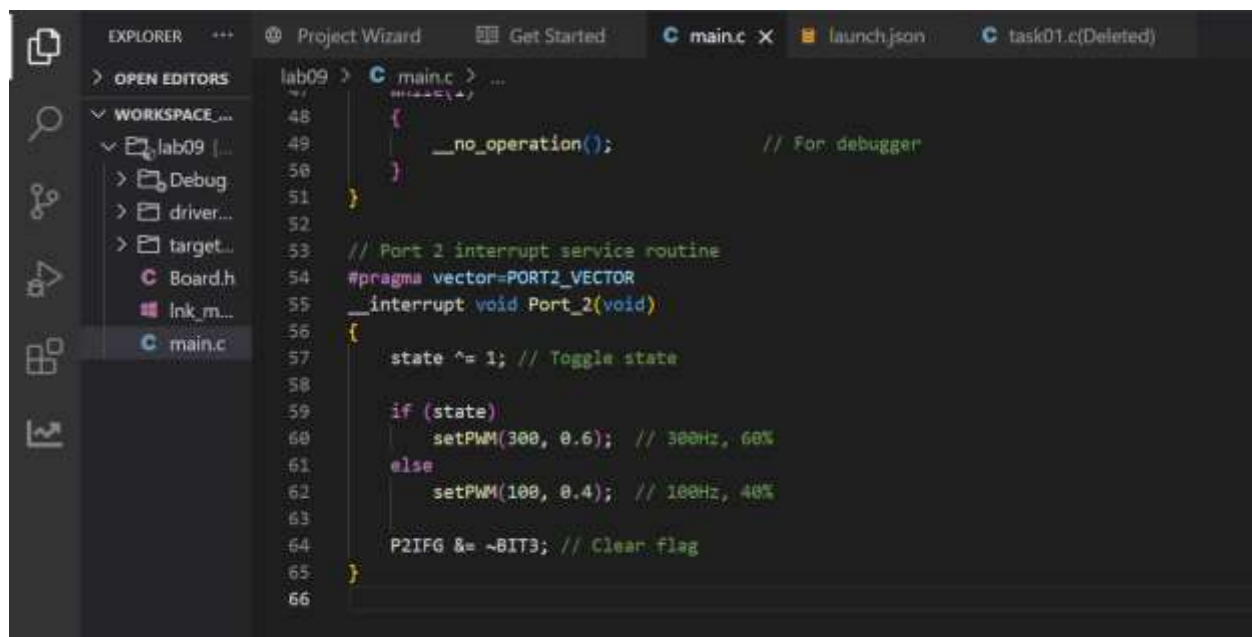
Generate a signal of 100Hz with 40% duty cycle on P1.2 When a user presses a button at P2.3 the signal change to 300Hz with 60% duty cycle. Button pressed means Press and release.

CODE:

```
File Edit Selection View Go Project Run Scripts Terminal Help Code Composer Studio
EXPLORER Project Wizard Get Started main.c x launch.json task01.c(Deleted)
WORKSPACE...
lab09 > C main.c > ...
1 #include <msp430.h>
2
3 unsigned char state = 0;
4
5 void setPWM(unsigned int freq, float duty)
6 {
7     if (freq == 100)
8     {
9         TACCR0 = 10000 - 1; // 100Hz
10        TACCR1 = (unsigned int)(10000 * duty); // 40% duty
11    }
12    else if (freq == 300)
13    {
14        TACCR0 = 3333 - 1; // 300Hz
15        TACCR1 = (unsigned int)(3333 * duty); // 60% duty
16    }
17 }
18
19 int main(void)
20 {
21     WDTCTL = WDTPW | WDTHOLD; // Stop watchdog
22     PM5CTL0 &= ~LOCKLPM5; // Enable GPIO
23 }
```



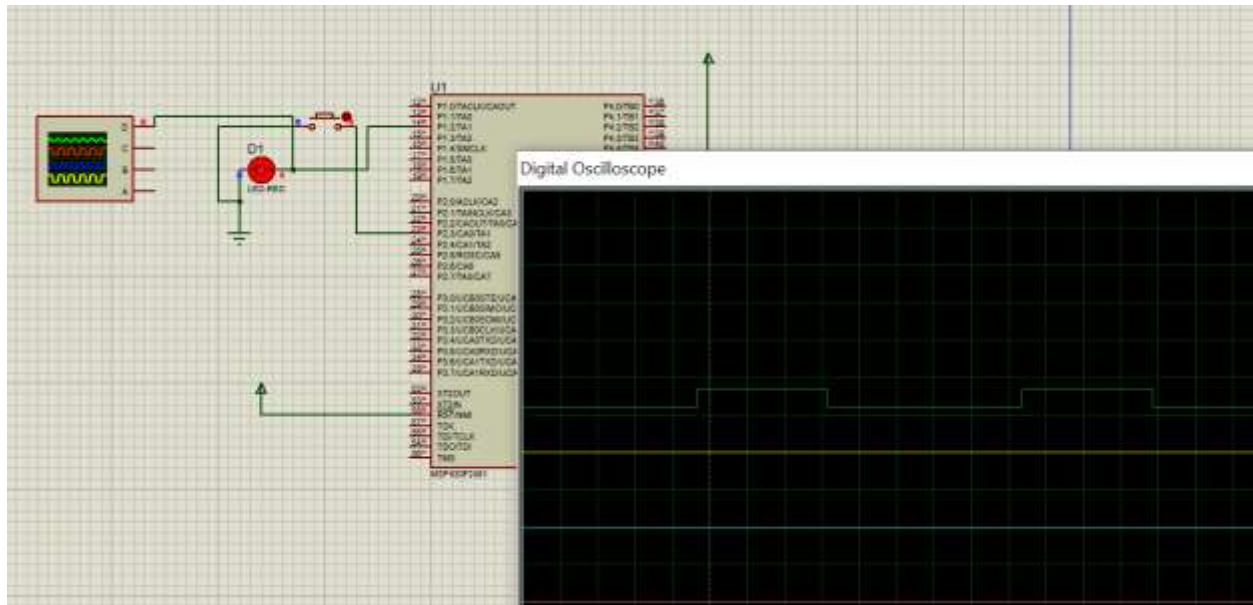
```
24 // Configure P1.2 as TAO.1 output
25 P1DIR |= BIT2;
26 P1SEL0 |= BIT2;
27
28 // Configure P2.3 as input with pull-up resistor
29 P2DIR &= ~BIT3;
30 P2REN |= BIT3;
31 P2OUT |= BIT3;
32
33 // Enable interrupt on P2.3 (falling edge)
34 P2IES |= BIT3;
35 P2IFG &= ~BIT3;
36 P2IE |= BIT3;
37
38 // Timer AB setup
39 TAOCTL1 = OUTMOD_7; // Reset/set mode
40 TAOCTL = TASSEL_SCLK | MC_UP | TACLK;
41
42 // Start with 100Hz, 40%
43 setPWM(100, 0.4);
44
45 __bis_SR_register(GIE); // Global interrupt enable
46
47 while(1)
```



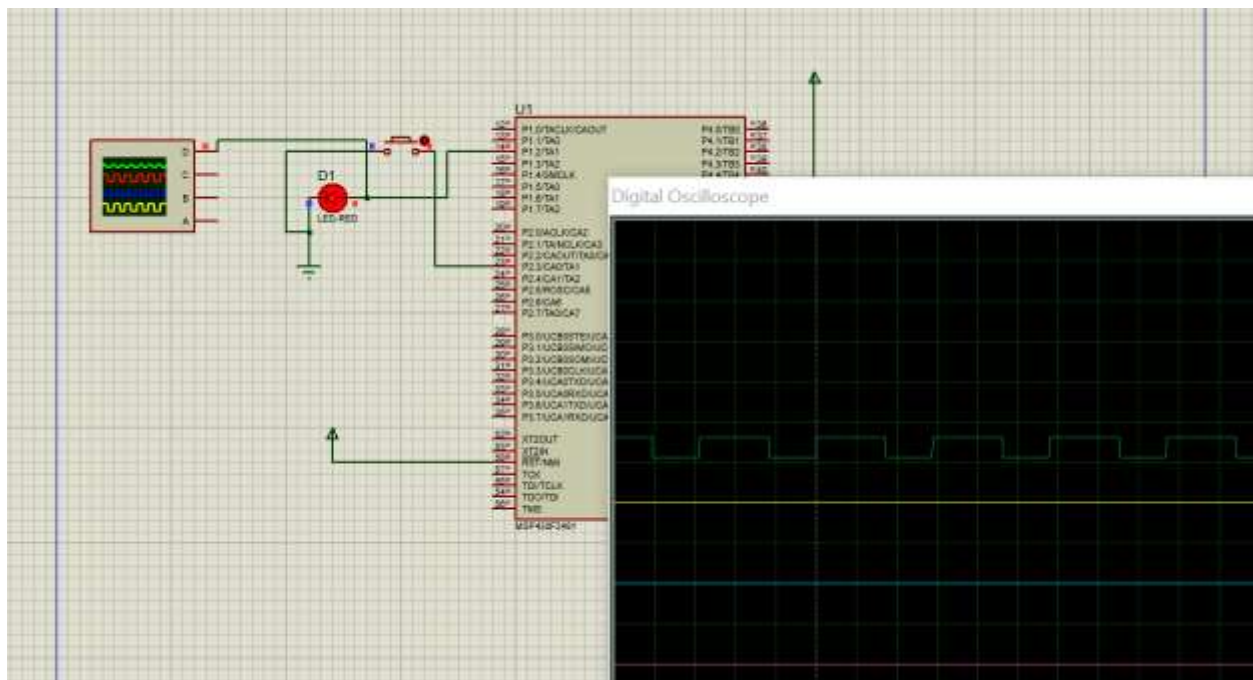
```
48 //
49 {
50     __no_operation(); // For debugger
51 }
52
53 // Port 2 interrupt service routine
54 #pragma vector=PORT2_VECTOR
55 __interrupt void Port_2(void)
56 {
57     state ^= 1; // Toggle state
58
59     if (state)
60         setPWM(300, 0.6); // 300Hz, 60%
61     else
62         setPWM(100, 0.4); // 100Hz, 40%
63
64     P2IFG &= ~BIT3; // Clear flag
65 }
66
```

OUTPUT:

When button not pressed.



When button pressed.



Task 04:

This task consists of two parts, A. Generate a signal x of 2KHz with 75% duty cycle on P1.2. Similarly, generate another signal y of 1KHz with 25% duty cycle on P1.3. As soon a user presses a button on P2.1, x frequency drops by 100Hz and y increases by 100Hz. If x crosses y, an LED at P2.2 is turned ON. Use low power mode when nothing is happening. Additionally, use interrupts and not polling in your program. a. Use Timer interrupt for delay creation

CODE:

```
lab09 > C main.c > setPWM_Y
1  #include <msp430.h>
2
3  volatile unsigned int freq_x = 2000; // x = 2 kHz
4  volatile unsigned int freq_y = 1000; // y = 1 kHz
5
6  // Update PWM for x (P1.2 using TA0.1)
7  void setPWM_X(unsigned int freq, float duty) {
8      unsigned int period = 1000000 / freq; // SMCLK = 1 MHz
9      TA0CCR0 = period - 1;
10     TA0CCR1 = (unsigned int)(period * duty);
11 }
12
13 // Update PWM for y (P1.3 using TA1.1)
14 void setPWM_Y(unsigned int freq, float duty) {
15     unsigned int period = 1000000 / freq;
16     TA1CCR0 = period - 1;
17     TA1CCR1 = (unsigned int)(period * duty);
18 }
19
20 // Update LED if x < y
21 void updateLED(void) {
22     if (freq_x < freq_y)
23         P2OUT |= BIT2; // Turn ON LED
24     else
```

```
lab09 > C main.c > setPWM_Y
25     P2OUT &= ~BIT2; // Turn OFF LED
26 }
27
28 int main(void)
29 {
30     WDTCTL = WDTPW | WDTHOLD; // stop watchdog
31     PM5CTL0 &= ~LOCKLPM5;
32
33     // --- PWM OUTPUTS on P1.2 and P1.3 ---
34     P1DIR |= BIT2 | BIT3;
35     P1SEL0 |= BIT2 | BIT3;
36
37     // --- LED OUTPUT on P2.2 ---
38     P2DIR |= BIT2;
39     P2OUT &= ~BIT2;
40
41     // --- BUTTON INPUT P2.1 w/ pull-up + interrupt ---
42     P2DIR &= ~BIT1;
43     P2REN |= BIT1;
44     P2OUT |= BIT1;
45     P2IES |= BIT1; // falling edge
46     P2IFG &= ~BIT1;
47     P2IE |= BIT1;
48 }
```

```

lab09 > C main.c > setPWM_Y
49 // --- Timer A0 (x: P1.2) ---
50 TA0CCTL1 = OUTMOD_7;
51 TA0CTL = TASSEL_2 | MC_1 | TACLK; // SMCLK, Up
52
53 // --- Timer A1 (y: P1.3) ---
54 TA1CCTL1 = OUTMOD_7;
55 TA1CTL = TASSEL_2 | MC_1 | TACLK;
56
57 // Initial PWM setup
58 setPWM_X(freq_x, 0.75);
59 setPWM_Y(freq_y, 0.25);
60 updateLED();
61
62 __bis_SR_register(GIE | LPM0_bits); // Enable interrupts, go LPM0
63 while (1); // stay in LPM0
64 }
65
66 // P2.1 button interrupt with simple debounce
67 #pragma vector = PORT2_VECTOR
68 __interrupt void Port_2_ISR(void)
69 {
70     P2IE &= ~BIT1; // disable button IRQ
71     P2IFG &= ~BIT1; // clear pending flag
72

```

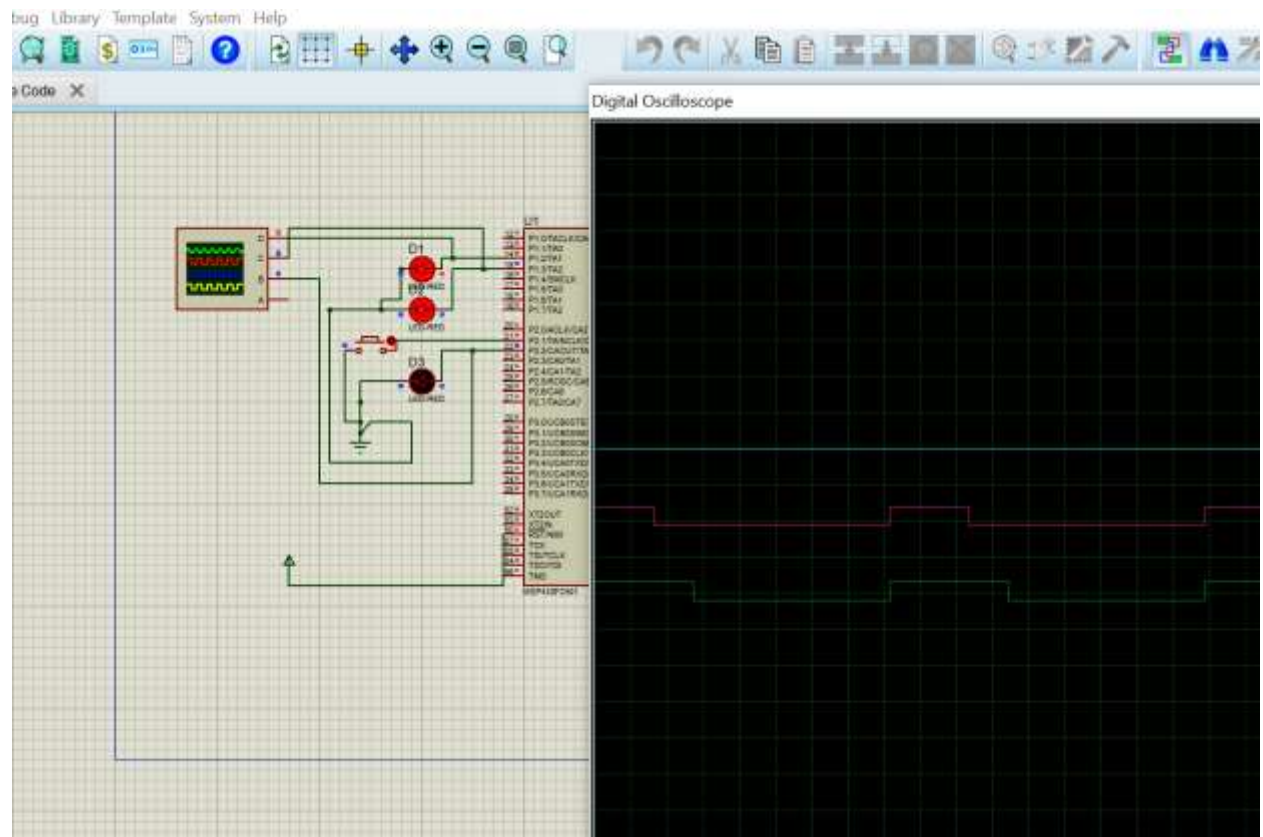
```

72
73     __delay_cycles(10000); // ~10 ms debounce
74
75     // Adjust frequencies:
76     if (freq_x > 200) freq_x -= 100;
77     if (freq_y < 10000) freq_y += 100;
78
79     setPWM_X(freq_x, 0.75);
80     setPWM_Y(freq_y, 0.25);
81     updateLED();
82
83     P2IE |= BIT1; // re-enable button IRQ
84 }
85

```

Output:

Before crossing of x and y



After crossing of x and y

