HW#8

Mrinmoy Sarkar

ECEN-621

Q1: Code:

```
#include "msp.h"

#define MAX_ADC_VALUE 16384

int period_of_blue_led = 16384;     // fixed period for blue
led

volatile uint16_t JOYresults[2];    // store the ADC value of
the joystick
float duty_cycle = 0.5;             // store the instantaneous
duty cycle of the two led

int main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;         // Stop
WDT

    // Configure GPIO for PWM output
    P2->DIR |= BIT6 | BIT7;                     // red led and
buzzer
    P2->SEL0 |= BIT6 | BIT7;
    P5->DIR |= BIT6;                            // blue led
    P5->SEL0 |= BIT6;

    // configure timer for pwm operation
    TIMER_A0->CCR[0] = 1000 - 1;                // PWM Period

    TIMER_A0->CCTL[3] = TIMER_A_CCTLN_OUTMOD_7; // CCR3
reset/set
    TIMER_A0->CCR[3] = 750;                     // CCR3 PWM
duty cycle

    TIMER_A0->CCTL[4] = TIMER_A_CCTLN_OUTMOD_7; // CCR4
reset/set
    TIMER_A0->CCR[4] = 500;                     // CCR4 PWM
duty cycle

    TIMER_A0->CTL = TIMER_A_CTL_SSEL__SMCLK |   // SMCLK
            TIMER_A_CTL_MC__UP |                // Up mode
            TIMER_A_CTL_CLR;                    // Clear TAR

    TIMER_A2->CCR[0] = period_of_blue_led - 1;  // PWM Period
```

```
    TIMER_A2->CCTL[1] = TIMER_A_CCTLN_OUTMOD_7; // CCR1
reset/set
    TIMER_A2->CCR[1] = 750;                     // CCR1 PWM
duty cycle

    TIMER_A2->CTL = TIMER_A_CTL_SSEL__SMCLK |   // SMCLK
            TIMER_A_CTL_MC__UP |                // Up mode
            TIMER_A_CTL_CLR;                    // Clear TAR

    // Configure GPIO for ADC
    P4->SEL1 |= BIT4;   // Enable A/D channel A9
    P4->SEL0 |= BIT4;
    P6->SEL1 |= BIT0;   // Enable A/D channel A15
    P6->SEL0 |= BIT0;

    // Turn on ADC14, extend sampling time to avoid overflow of
results
    ADC14->CTL0 = ADC14_CTL0_ON | ADC14_CTL0_MSC |
ADC14_CTL0_SHT0__192 | ADC14_CTL0_SHP | ADC14_CTL0_CONSEQ_3;

    ADC14->MCTL[0] = ADC14_MCTLN_INCH_9;
// ref+=AVcc, channel = A9
    ADC14->MCTL[1] = ADC14_MCTLN_INCH_15 | ADC14_MCTLN_EOS;
// ref+=AVcc, channel = A15, end seq.

    ADC14->IER0 = ADC14_IER0_IE1;           // ADC interrupt
enable for ADC14->MEM[1]

    SCB->SCR |= SCB_SCR_SLEEPONEXIT_Msk;    // sleep on exit

    // Start conversion-software trigger
    ADC14->CTL0 |= ADC14_CTL0_ENC | ADC14_CTL0_SC;

    NVIC->ISER[0] = 1 << ((ADC14_IRQn) & 31);   // Enable ADC
interrupt in NVIC module

    // Enable global interrupt
    __enable_irq();

    // Enter LPM0
    __sleep();
}

// ADC14 interrupt service routine
void ADC14_IRQHandler(void)
{
```

```
    if (ADC14->IFGR0 & ADC14_IFGR0_IFG1)
    {
        JOYresults[0] = ADC14->MEM[0];   // Move A9 results,
IFG is cleared. vertical joystick
        JOYresults[1] = ADC14->MEM[1];   // Move A15 results,
IFG is cleared. horizontal joystick
        duty_cycle = (float)JOYresults[1] / MAX_ADC_VALUE;  //
calculate the duty cycle

        TIMER_A0->CCR[0] = JOYresults[0] - 1;              //
reset the PWM Period for buzzer
        TIMER_A0->CCR[3] = duty_cycle * JOYresults[0];     //
reset the duty cycle of red led
        TIMER_A0->CCR[4] = 0.5*JOYresults[0];              //
set the buzzer duty cycle to 50%
        TIMER_A2->CCR[1] = duty_cycle * period_of_blue_led; //
reset the duty cycle of blue led
        __delay_cycles(2000);                              //
small delay
    }
}
```

Q1: UML diagram: