Politechnika Opolska

L A B O R A T O R I U M

PRZEDMIOT:	Technika mikroprocesorowa

KIERUNEK STUDIÓW:	***AUTOMATYKA I ROBOTYKA***	ROK STUDIÓW:	***3***
SPECJALNOŚĆ:	**-**		
SEMESTR:	**6**	ROK AKADEMICKI:	***2019/2020***

Temat ćwiczenia:

Wykorzystanie liczników oraz timerów z użyciem mikrokontrolera MSP430.

Ćwiczenie wykonali:					
Nazwisko:		*Imię:*	*Nazwisko:*		*Imię:*
1.	**Janas**	**Kamil**	**2.**	**Indyk**	**Dominik**
3.	**Dittrich**	**Sebastian**			

Uwagi:	*Data:*	*Ocena za sprawozdanie:*

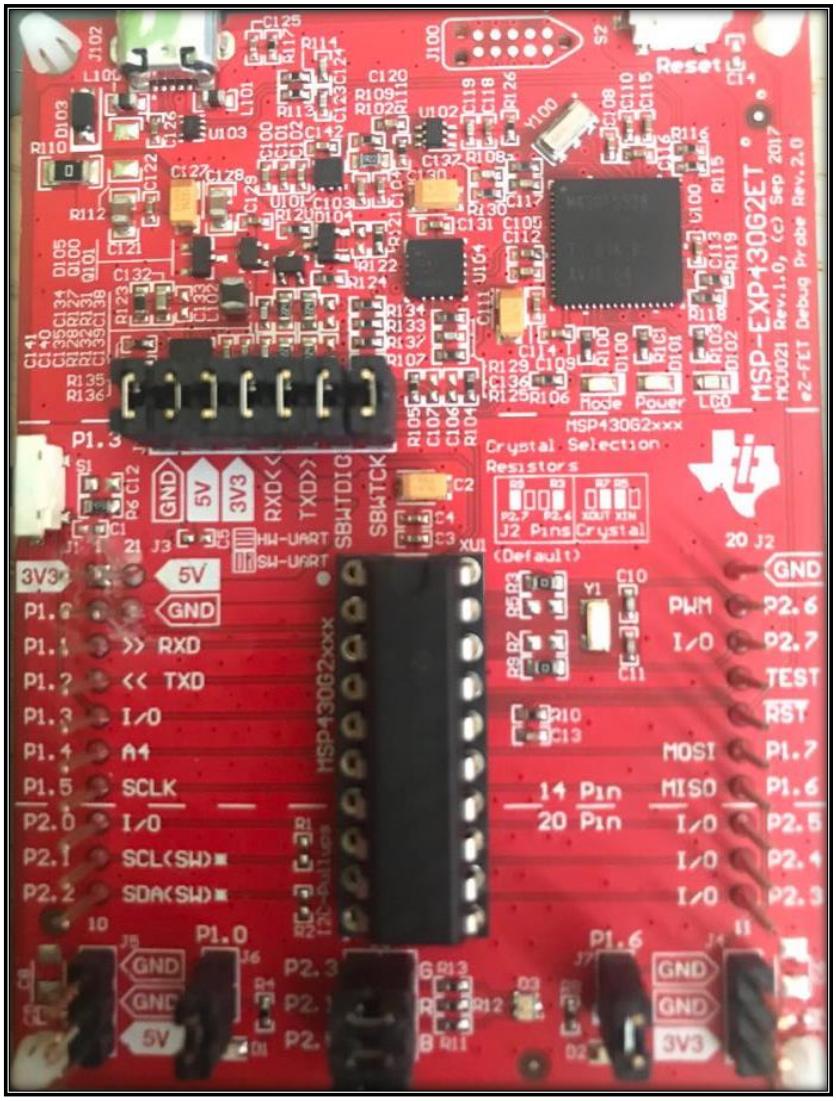
Termin zajęć:					
Data:	07.03.2020	Dzień tygodnia:	SOBOTA	Godzina:	15:40

1. Cel ćwiczenia.

Głównym celem ćwiczenia jest stworzenie algorytmu opierającego się na wykorzystaniu licznika oraz timer'ów, które zawarte są na mikrokontrolerze. Program ma w założeniu wykonywać się następująco:

Naciśnięcie przycisku → Odliczenie 3s (Timer 1) → Zaświeca się dioda zielona → Odliczanie 3s (Timer 1) →Wygaszenie diody zielonej, zaświeca się dioda czerwona (uruchomienie licznika +1) → Powtórz krok aż licznik zliczy 5 → Wyłączenie Timer1 uruchomienie Timer2 (1s) → Dioda czerwona zamruga 2x → Wyłączenie Timera2 oraz reset programu

Do stworzenia tego projektu wykorzystano oprogramowanie CodeComposer, który pozwala na podświetlanie składni oraz implementacje kodu do mikrokontrolera. Układ jest przedstawiony na zdjęciu 1.



Zdj.1. Mikrokontroler MSP430G2553 wykorzystany w ćwiczeniu.

2. Kod skryptu programu.

```
#include "msp430g2553.h"
// DEFINICJA DIOD I PRZELACZNIKA
#define ZIELONA BIT0                    // ZIELONA DIODA P1.0
#define CZERWONA BIT6                   // CZERWONA DIODA P1.6
#define PRZELACZNIK BIT3                // PRZELACZNIK P1.3
// DEFINICJA ZMIENNYCH CALKOWITYCH DO REALIZACJA PROGRAMU
int counter;                            // ZMIENNA CALKOWITA
int x10=0;                              // ZMIENNA CALKOWITA
void main(void){
    WDTCTL = WDTPW + WDTHOLD;           // STOP WATCHDOG TIMER
    // DIODA ZIELONA
    P1DIR |= ZIELONA;                   // USTAWIENIE NA WYJSCIU BITU DIODY
    P1OUT &= ~ZIELONA;                  // WYLACZENIE DIODY ZIELONEJ
    // DIODA CZERWONA
    P1DIR |= CZERWONA;                  // USTAWIENIE NA WYJSCIU BITU DIODY
    P1OUT &= ~CZERWONA;                 // WYLACZENIE DIODY CZERWONEJ
    // PRZELACZNIK
    P1DIR &= ~PRZELACZNIK;              // USTAWIENIE PRZELACZNIKA
    P1REN |= PRZELACZNIK;               // URUCHOMIENIE REZYSTORA P1.3
    P1OUT |= PRZELACZNIK;               // USTAWIENIE PULL-UP NA REZYSTORZE
    // TIMER
    TA0CCTL0 |= CCIE;                   // USTAWIENIE PRZERWANIA PO TIMERZE0
    TA0CTL |= TASSEL_2 + TACLR + ID_3;  // WYKONANIE ZLICZANIA NA TIMERZE SMCLK
    TA1CCTL0 |= CCIE;                   // USTAWIENIE PRZERWANIA PO TIMERZE1
    TA1CTL |= TASSEL_2 + TACLR + ID_3;  // WYKONANIE ZLICZANIA NA TIMERZE SMCLK
    TA0CCR0 = 37000;                    // TIMER0 - 3s
    TA1CCR0 = 12500;                    // TIMER1 - 1s
    // PRZERWANIE
    P1IES &= ~PRZELACZNIK;                      // USTAWIENIE ZBOCZA NARASTAJACEGO PRZY PUSZCZANIU PRZYCISKU
    P1IE |= PRZELACZNIK;                        // PRZERWANIE NA PRZYCISKU

    __bis_SR_register(LPM0_bits + GIE);
}
// USTAWIENIE PRZERWANIA NA WCISNIECIE PRZYCISKU
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){
    counter=0;                              // WARTOSC ZMIENNEJ USTAWIONEJ NA 0
    TACTL |= MC_1;                          // PRZERWANIE TIMERA0
    P1IFG &= ~PRZELACZNIK;                  // NASTEPUJE CZYSZCZENIE FLAGI PRZYCISKU
}
// NASTEPUJE PRZERWANIE PO ODLICZENIU TIMERA0
#pragma vector = TIMER0_A0_VECTOR
__interrupt void CCR0_ISR(void){

    x10++;                                  // INKREMENTACJA ZMIENNEJ X10
    if (x10==10){                           // JESLI ZMIENNA X10 BEDZIE WYNOSILA 10
        if (counter==0){                    // ORAZ ZMIENNA COUNTER 0 TO
            P1OUT |= ZIELONA;               // NASTEPUJE ZAPALENIE DIODY ZIELONEJ
        }
        else if (counter<4){                // GDY ZMIENNA COUNTER MNIEJSZA OD 4
            P1OUT ^= ZIELONA;               // NASTEPUJE ZMIANA STANU NA DIODZIE ZIELONEJ
            P1OUT ^= CZERWONA;              // ORAZ ZMIANA STANU NA DIODZIE CZERWONEJ
        }
        else{                               // W POZOSTALYCH PRZYPADKACH
            counter=0;                      // ZMIENNA COUNTER WYNOSI 1
            TA0CTL &= ~MC_3;                // ZATRZYMA SIE TIMER0 ORAZ
            TA1CTL |= MC_1;                 // URUCHOMI SIE TIMER1
            P1OUT &= ~ZIELONA;              // WYLACZENIE DIODY ZIELONEJ ORAZ
            P1OUT &= ~CZERWONA;             // WYLACZENIE DIODY CZERWONEJ
            x10=0;                          // WYZEROWANIE ZMIENNEJ X10
        }
        x10=0;                              // WYZEROWANIE ZMIENNEJ X10
        counter++;                          // ZMIENNA COUNTER JEST INKREMENTOWANA
    }
}
```

```
// WYKONUJE SIE PRZERWANIE ODLICZANIE TIMERA1
#pragma vector = TIMER1_A0_VECTOR
__interrupt void CCR1_ISR(void){
        while(counter<=4){                  // WYKONANIE 2 MIGNIEC DIODY CZERWONEJ
            P1OUT ^= CZERWONA;              // ZMIANA STANU WYJSCIA DIODY CZERWONEJ
            __delay_cycles(1000000);        // OPOZNIENIE WYKONANIA PETLI - 1s
            counter++;                      // ZMIENNA COUNTER JEST INKREMENTOWANA
        }

        TA1CTL &= ~MC_3;                    // ZATRZYMANIE TIMERA1
}
```

3. Opis programu.

W początkowych liniach kodu zostały zdefiniowane zmienne w oparciu o porty oraz bity znajdujące się na mikrokontrolerze odpowiedzialne za diodę zieloną, czerwoną oraz przycisk, a także zadeklarowano zmienne całkowite odpowiedzialne za licznik. W sekcji *void main(void)* zostały zadeklarowane bity diod, przycisk oraz ustawienia obu czasówek (timerów). Przy wykorzystaniu *#pragma vector=PORT1_VECTOR* ustawiono przerwanie timera 0 oraz wyzerowanie licznika.
W sekcji *#pragma vector = TIMER0_A0_VECTOR* zostało określone przerwanie po odliczeniu czasu 3s oraz wariant przełączania się diód zliczanych w liczniku z wykorzystaniem instrukcji warunkowej *if*. W ostatniej części programu określono wykonanie przerwania na timera 1 w którym zawarto wykonanie 2 mignięć diody czerwonej w opóźnieniu 1 sekundy po czym następuje zakończenie programu oraz zatrzymanie timera 1. Pozwala to przy ponownym przyciśnieciu przycisku na kolejne wykonywanie programu.

4. Wnioski.

Powyżej przedstawione rozwiązanie zadanego algorytmu pokazuje, że wykorzystując liczniki oraz timery można w dowolny sposób przy użyciu opcji dostępnej na mikrokontrolerze SMCLK wykonywać wiele algorytmów przełączania oraz zmiany stanów wyjściowych na diodach, bez konieczności użycia przycisku do ich zmiany, działając jedynie na czasie, ściśle określając go w kodzie źródłowym programu wgrywanego do płytki MSP430.