Podstawy techniki mikroprocesorowej 2

Ćwiczenie 2 – PWM

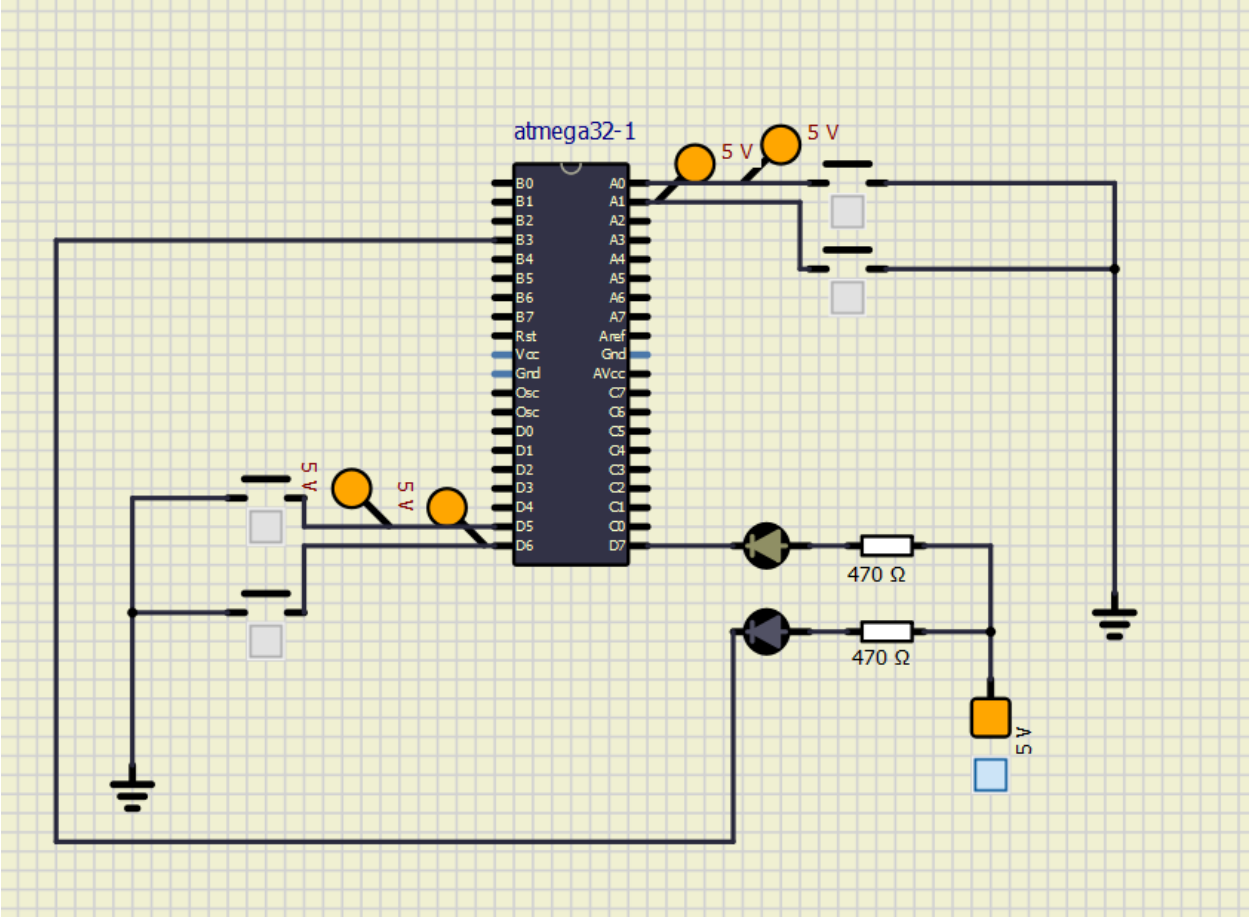
Łukasz Chwistek

Nr. albumu: 243662

1. Wstęp

Ćwiczenie polegało na symulacji układu sterującego jasnością diody, przy pomocy sprzętowego sygnału PWM mikrokontrolera Atmega32.

2. Schemat układu



3. Kod programu

Kod programu konfiguruje Timer0 oraz Timer2 na tryb Fast PWM i ustawia odpowiednie porty. W pętli programu kontrolowany jest stan przycisków, które sterują wypełnieniem sygnału PWM, zmieniając intensywność świecenia diody do nich przypisanej. Przycisk podłączony do portu D5 zwiększa jasność świecenia diody podłączone do pinu B3.

Ustawiając rejestr TCCR0 konfigurujemy sygnał PWM, gdzie:

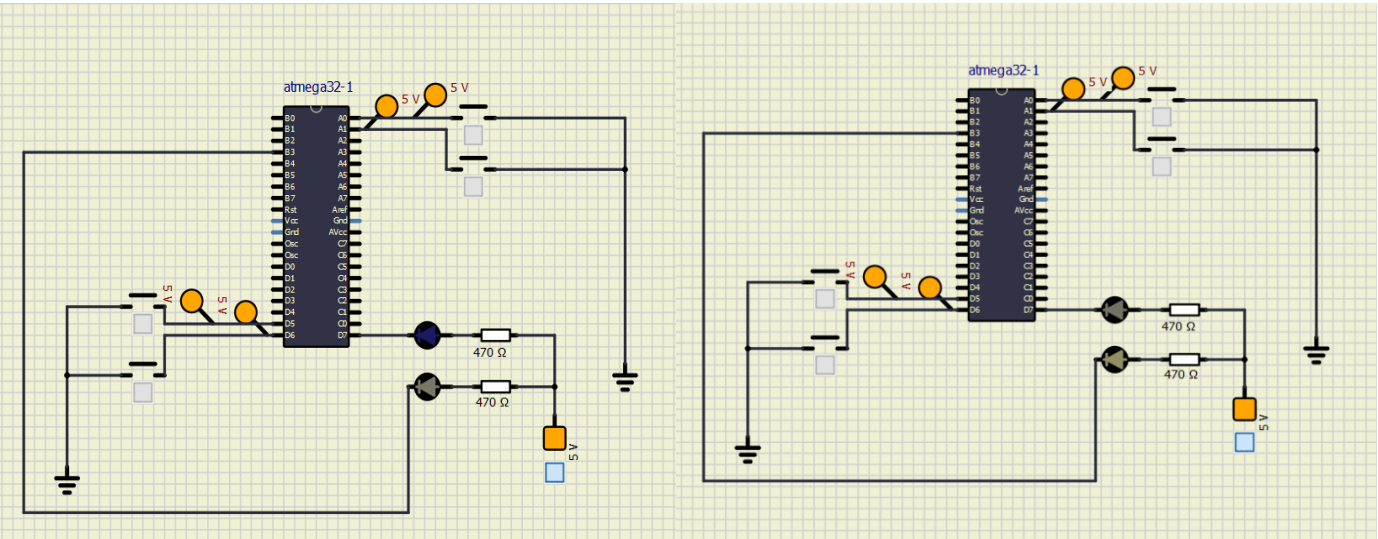
- bity WGMxx ustawiają tryb timerów
- bity CSXX ustawiają sygnał zegara
- bity COMxx konfigurują tryb wyjściowy PWM
- rejestr OCRx pozwala na sterowanie wypełnieniem sygnału PWM

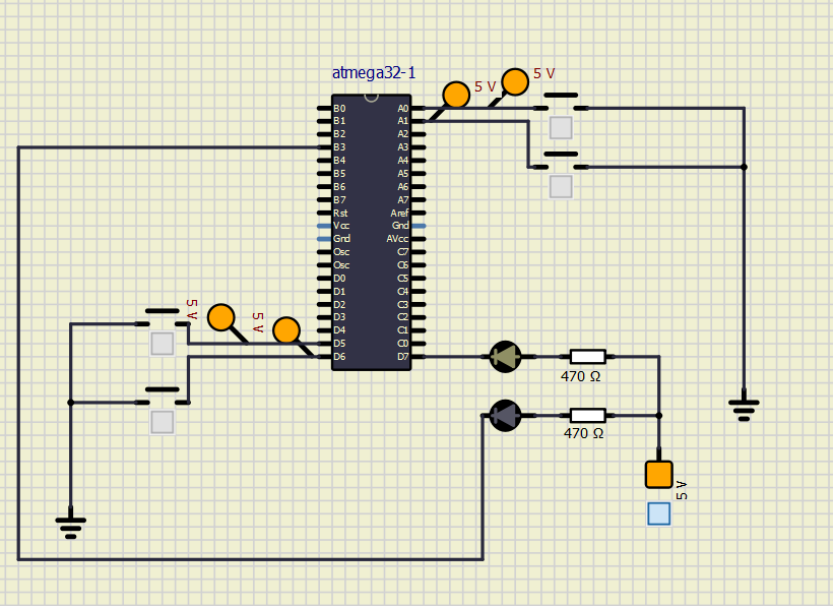
```c
/*
 * Cwiczenie 2
 *
 *  Created on: 21 gru 2020
 *      Author: Lukasz Chwistek
 */
#define __AVR_ATmega32__
#define F_CPU 8000000UL             //definiujemy F_CPU na 8MHz

#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <string.h>

/* T = F_CPU/(freq*2*N) -1 */

void PWM_init()
{
    /*ustawienie PWM z nieodwróconym wyjściem, bez prescalera*/
    TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS00);
    OCR0 = 127;                     //licznik wstepnie ustawiony na polowe wypelnienia sygnalu PWM
    DDRB|=(1<<DDB3);                //PB3 ustawiony jako wyjscie, OC0 jako wyjscie
}


void PWM_init1()
{
    /*ustawienie PWM z nieodwróconym wyjściem, bez prescalera*/
    TCCR2 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS00);
    OCR2 = 127;
    DDRD|=(1<<DDD7);                //PD7 ustawiony jako wyjscie, OC2 jako wyjście
}

int main(void){
                                    //pozostale bity sa domyslnie wejsciami
    PORTA |= (1<<PA0) | (1<<PA1);   //porty PA0 i PA1 ustawione jako stan wysoki
    PORTD |= (1<<PD5) | (1<<PD6);   //porty PD5 i PD6 ustawione jako stan wysoki

    PWM_init();
    PWM_init1();


    while (1)
    {
        if (!(PIND & (1<<PIN5)))    //jezeli na pinie 5 portu D wystapi 0 to
        {
            if (OCR0<250)
            {
                OCR0+=5;            //zwiekszenie intensywnosci swiatla, zwiekszenie wypełnienia pierwszego PWM
            }
            _delay_ms(25);
        }
        if (!(PIND & (1<<PIN6)))
```

```
53          if (!(PIND & (1<<PIN6)))
54          {
55              if(OCR0>5)
56              {
57                  OCR0-=5;            //zmniejszenie intensywnosci swiatla
58              }
59              _delay_ms(25);
60          }
61
62          if (!(PINA & (1<<PIN0)))    //jezeli na pinie 0 portu A wystapi 0 to
63          {
64              if (OCR2<250)
65              {
66                  OCR2+=5;            //zwiekszenie intensywnosci swiatla, zwiekszenie wypełnienia drugiego PWM
67              }
68              _delay_ms(25);
69          }
70          if (!(PINA & (1<<PIN1)))
71          {
72              if(OCR2>5)
73              {
74                  OCR2-=5;            //zmniejszenie intensywnosci swiatla
75              }
76              _delay_ms(25);
77          }
78      }
79      return 0;
80  }
81
```

4. Wyniki

Po kompilacji programów i wgraniu ich do mikrokontrolera oba programy pracują zgodnie z założeniami. Wciśnięcie przycisku zapala diodę LED, a ponowne wciśnięcie ją gasi.





5. Wnioski

Ćwiczenie pozwoliło zapoznać się z pracą z dokumentacją mikrokontrolera w celu jego konfiguracji do uzyskania sygnału PWM. Program działa zgodnie z założeniami, więc ćwiczenie zostało wykonane poprawnie.