Podstawy techniki mikroprocesorowej 2

Ćwiczenie 3 – Generator sinusoidalny na PWM

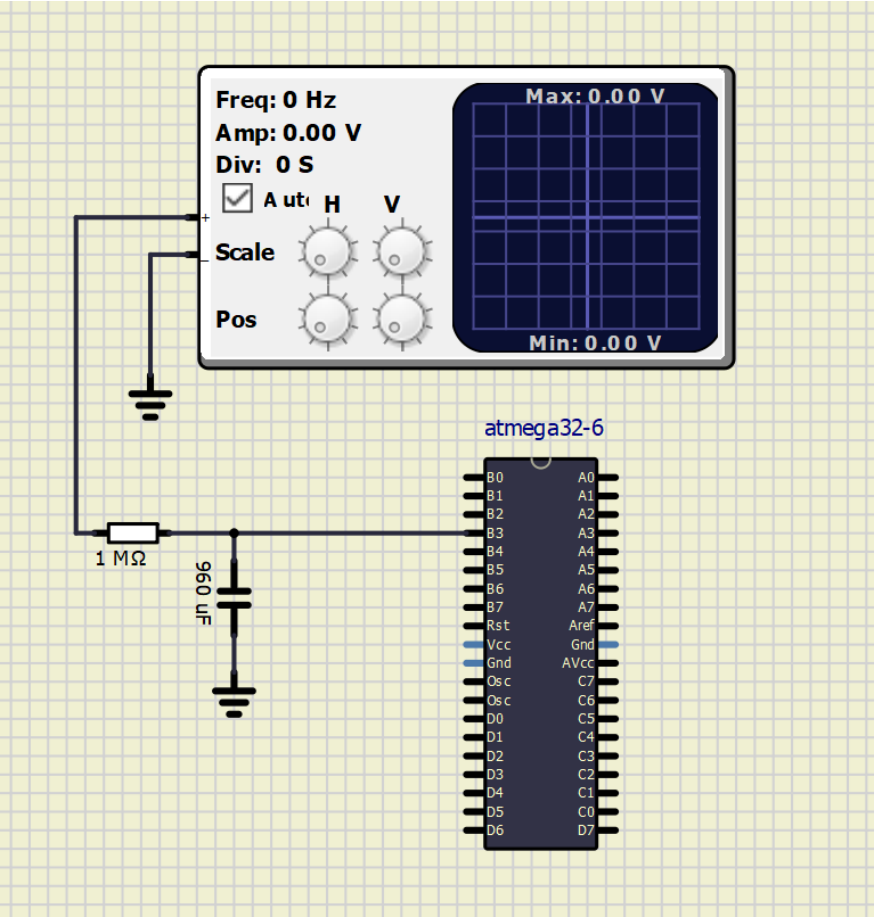
Łukasz Chwistek

Nr. albumu: 243662

1. Wstęp

Ćwiczenie polegało na stworzeniu przetwornika C/A z filtrem RC, a program miał generować sygnał sinusoidalny bazowany na sygnale PWM generowanym przez mikroprocesor, który można obejrzeć na oscyloskopie.

2. Schemat układu



3. Wykonanie zadania

Wykorzystując w projekcie sprzętowy PWM oparty o Timer0 możliwe jest wygenerowanie sygnału sinusoidalnego. Wypełnieniem sterujemy zmieniając wartość rejestru OCR0, a gdy się on przepełni, to od tego momentu przechwytujemy sygnał, wykorzystując przerwanie TIMER0_OVF_vect.

$$f_{PWM} = \frac{F\_CPU}{preskaler \cdot rozdzielczośćpwm}$$

Timer0 jest skonfigurowany w pracy Fast PWM, przeskalowany o clk/8 przy częstotliwości F_CPU=8MHz i daje nam to wartość 3 906,25Hz.

$$f_{sin} = \frac{f_{PWM}}{n}$$

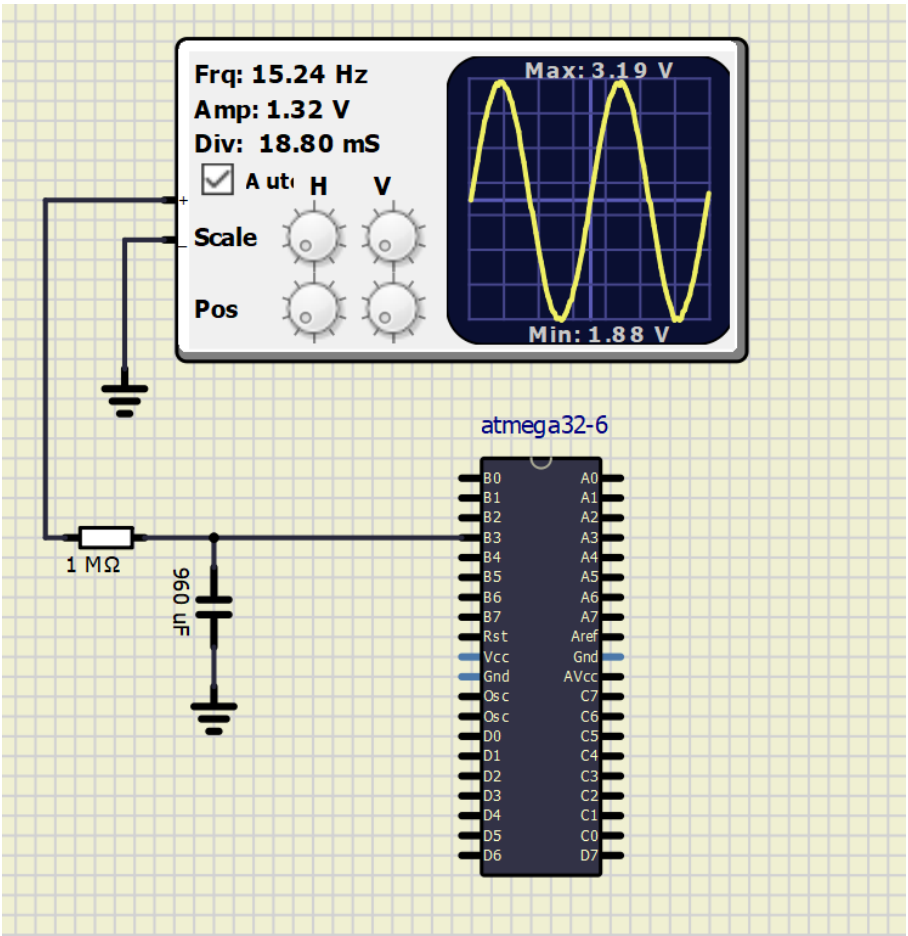
Generowany sygnał ma 256 okresów sygnału PWM, a więc częstotliwość sygnału sinusoidalnego to 15.26Hz.

$$\frac{1}{f_{sin}} = \frac{2\pi}{R_0 C_0} \Rightarrow C_0 = \frac{2\pi f_{sin}}{R_0}$$

Dopiero po przefiltrowaniu przez filtr RC otrzymamy sygnał sinusoidalny. Iloczyn wartości RC musi być większy od wartości R_0 i C_0 z powyższego równania. Dla R_0 = 1MΩ wartość pojemności to C_0=96uF. Według założeń wartość 10 razy większa jest wystarczająca, więc zbudowano filtr wykorzystując rezystor 1MΩ oraz kondensator 960uF.

4. Wyniki

Otrzymany sygnał jest zbliżony do sinusoidy, co pozwala stwierdzić, że obliczenia zostały wykonane poprawnie a program spełnia założenia.



5. Wnioski

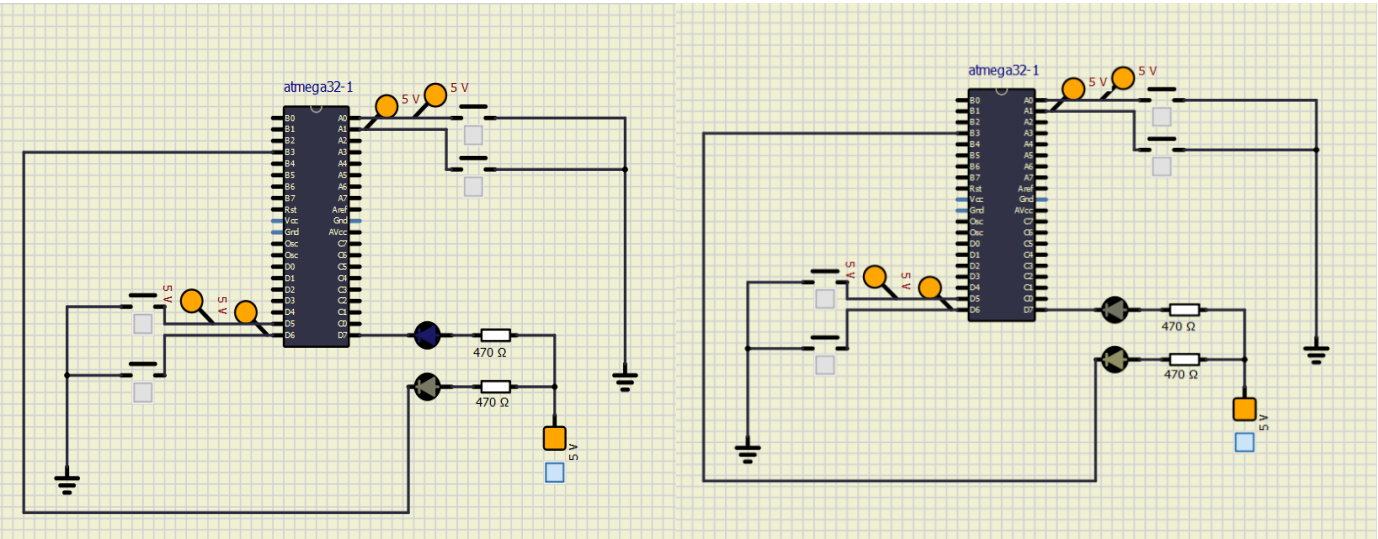
Dzięki odpowiednim obliczeniu częstotliwości sygnału PWM i sygnału sinusoidalnego, w sposób analityczny można obliczyć wartości elementów RC. Wykorzystano przerwania timera, co umożliwiło dynamiczną zmianę wartości przepełnienia sygnału i otrzymano całkiem dokładny sygnał sinusoidalny.

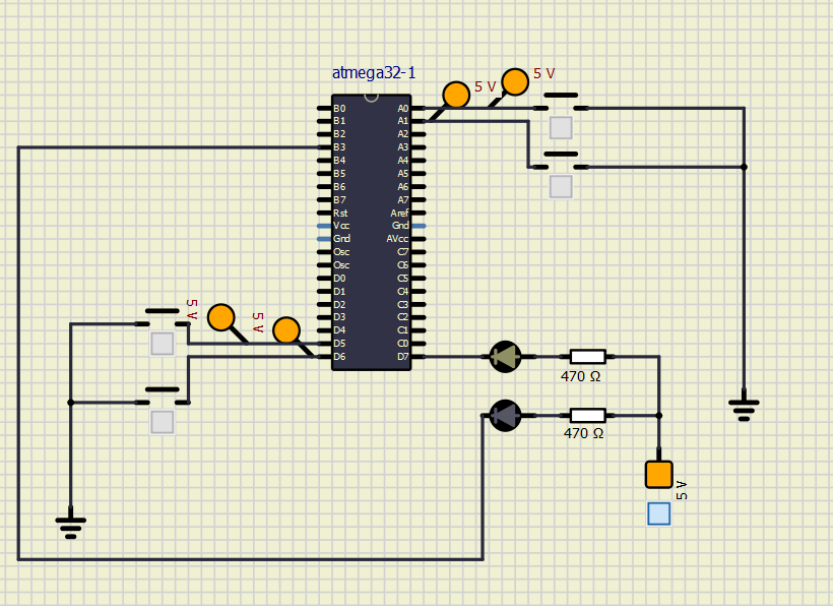
```c
/*
 * Cwiczenie 2
 *
 *  Created on: 21 gru 2020
 *      Author: Lukasz Chwistek
 */
#define __AVR_ATmega32__
#define F_CPU 8000000UL             //definiujemy F_CPU na 8MHz

#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
```

6. 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.





7. 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.