Podstawy techniki mikroprocesorowej 2

Ćwiczenie 5 – Regulator PI z regulowanymi wartościami Kp i Ti

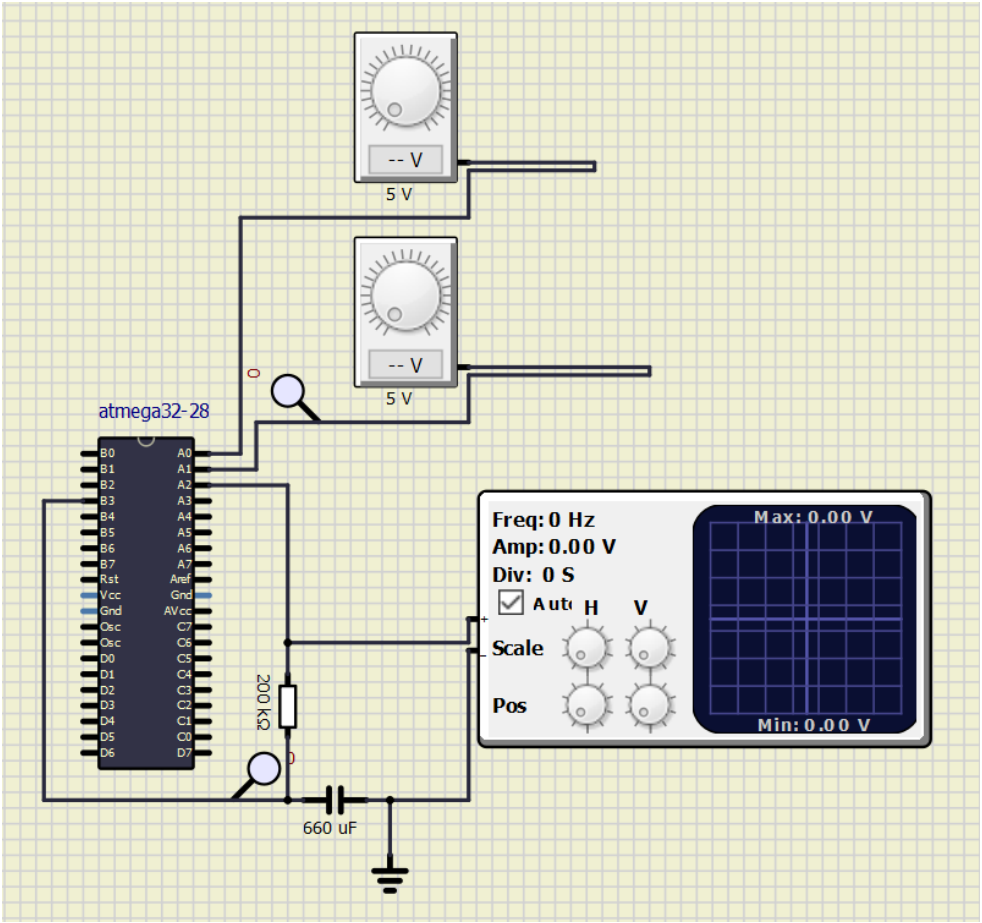
Łukasz Chwistek

Nr. albumu: 243662

1. Wstęp

Celem ćwiczenia było stworzenie regulatora typu PI z wyjściem PWM oraz regulowanymi nastawami Ti i Kp.

2. Schemat układu



Działanie regulatora jest realizowanie w oparciu o mikrokontroler Atmega32. Dla równoległego regulatora typu PI wynik członu proporcjonalnego i całkującego są sumowane. Czasem próbkowania jest okres sygnału PWM ustawionego z preskalerem clk/64, a wraz z przepełnieniem licznika timera generującego sygnał PWM następuje pobranie wartości z pinu A2. Sterowanym obiektem jest układ całkujący RC, do którego podpięty został oscyloskop. Sterowalne źródła napięcia pozwalają na konfigurację nastaw Kd i Ti regulatora. Górne ograniczenie wartości wyjściowej zostało ustawione na 2000.

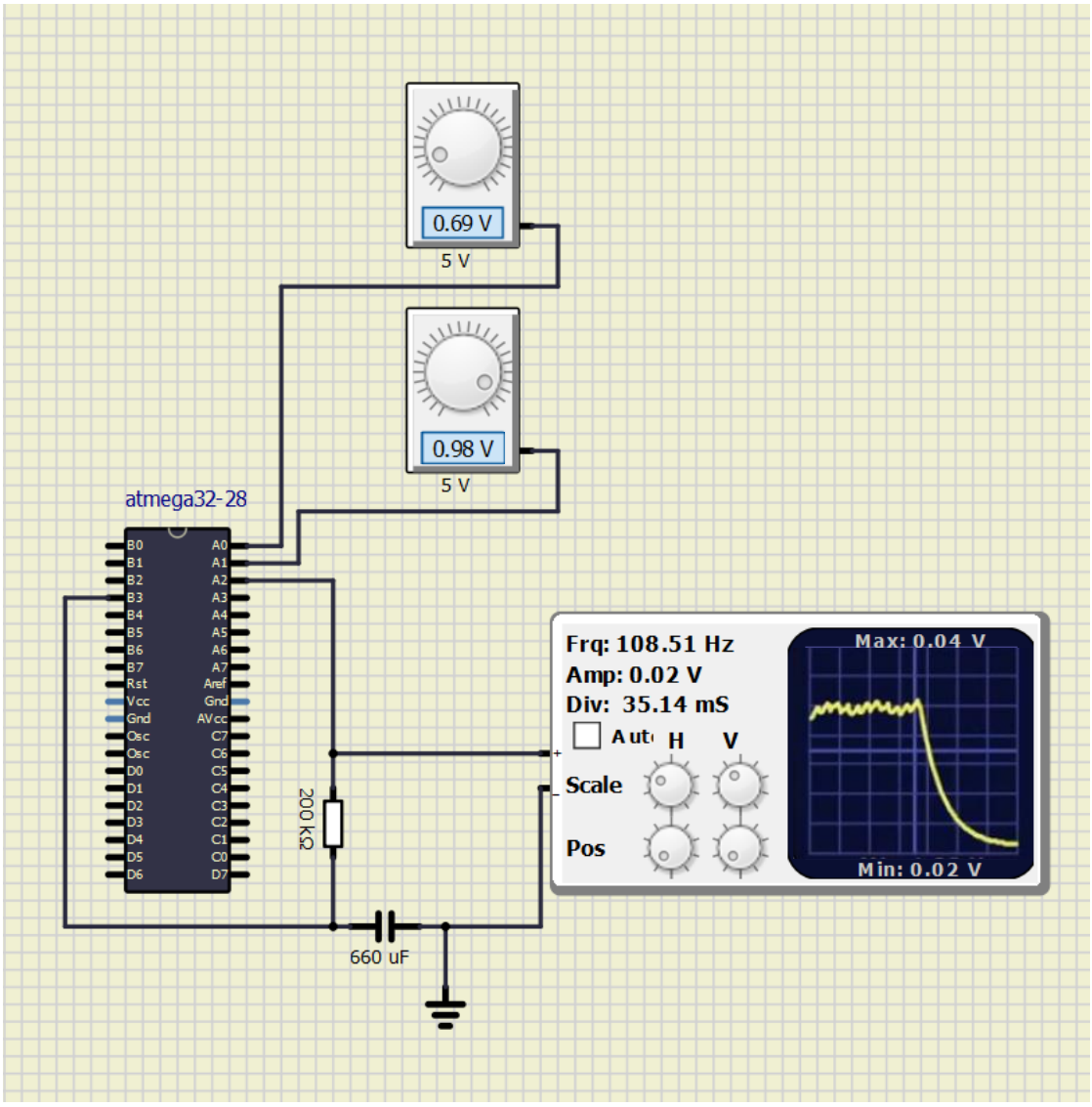
3. Kod programu

```c
/*
 *  Cwiczenie 5
 *  regulator PI z wyjsciem PWM i regulowanymi nastawami
 *
 *  Created on: 18 luty 2021
 *      Author: Łukasz Chwistek
 */



#define __AVR_ATmega32__
#define F_CPU 8000000UL

#include <avr/io.h>
#include <stdlib.h>
#include <string.h>
#include <util/delay.h>
#include "GLOBAL.h"


uint16_t ograniczenie = 2000;
float Kp = 1.0f;
float Ti = 1.0f;
volatile int16_t ustawienie;
volatile int16_t uchyb;


uint16_t ADC_start(int pin)
{
    ADMUX = pin;
    ADMUX |= (1<<REFS0);                                //napiecie referencyjne AVcc 5V
    ADMUX &= ~(1<<ADLAR);                               //10bit, przesunięcie w lewo
    ADCSRA |= (1<<ADPS0) | (1<< ADPS1) | (1<< ADPS2) | (1<<ADEN);  //prescaler 128,  umożliwienie konwersji
    ADCSRA|= (1<<ADSC);                                 //start conversion
    while (ADCSRA & (1<<ADSC));                         //wykonanie pomiaru
    _delay_ms(100);

    int16_t analog = (int16_t)ADCL;
    analog += (int16_t)(ADCH<<8);                      //ADCL must be read first, then ADCH
    return analog;
}
/*Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and
a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost.
```

```
    return analog;
}
/*Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and
a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost.
When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled. */

void PWM_init()
{
    /*ustawienie PWM z nieodwróconym wyjściem, prescaler 8 */
    TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS01) | (1<<CS00);   //rejestr sterujący licznikiem, fastPWM, prescaler 8
    OCR0 = 128;                                                             //wypelnienie sygnalu PWM, PB3

    TIMSK = (1<<TOIE0);                                                     //włączenie przerwania przepełnienia na timerze0
}

float skalowanie_regulatora(int16_t wartosc)
{
    return ((float)wartosc)/1024;
}

int16_t sumowanie(int16_t wartosc)
{
    static int16_t suma = 0;
    if(abs(suma += wartosc) > ograniczenie)
    {
        suma = ograniczenie;
    }
    return suma;
}

int16_t nastawy_regulatora(int16_t wartosc)
{
    int16_t uchyb = ustawienie -  wartosc;
    int16_t wynik = (int16_t) (Kp*uchyb + Ti*sumowanie(uchyb));
    if (wynik < 255)
        {
            if (wynik > 0)
                return wynik;
            else
                return 0;
        }
    else
        return 255;
```

```
        return 255;
}

ISR(TIMER0_OVF_vect)
{
    OCR0 = nastawy_regulatora(ADC_start(PINA2));
}


int main(void)
{
    DDRB |= (1<<DDB3);                                          //PB3 ustawiony jako wyjscie, OC0 jako wyjsciowy wektor
    PWM_init();

    sei();

    while(1)
    {
        Kp = skalowanie_regulatora(ADC_start(PINA0));
        Ti = skalowanie_regulatora(ADC_start(PINA1));

        ustawienie = 23;
        _delay_ms(200);
        ustawienie = 98;
        _delay_ms(250);
    }
}
```

4. Wyniki symulacji



5. Wnioski

Układ regulatora PI z możliwością wartości Kp i Ti oraz wyjściem PWM został wykonany zgodnie z założeniami. Za pomocą regulatorów analogowych można zmieniać wartość Kp i Ti, a wartości zostały ograniczone górnie do 2000. W projekcie wykorzystano przerwania timera, sygnał PWM oraz przetworniki analogowo cyfrowy na pinach mikrokontrolera.