Podstawy techniki mikroprocesorowej 2

Ćwiczenie 4 – Regulator dwustawny z histerezą

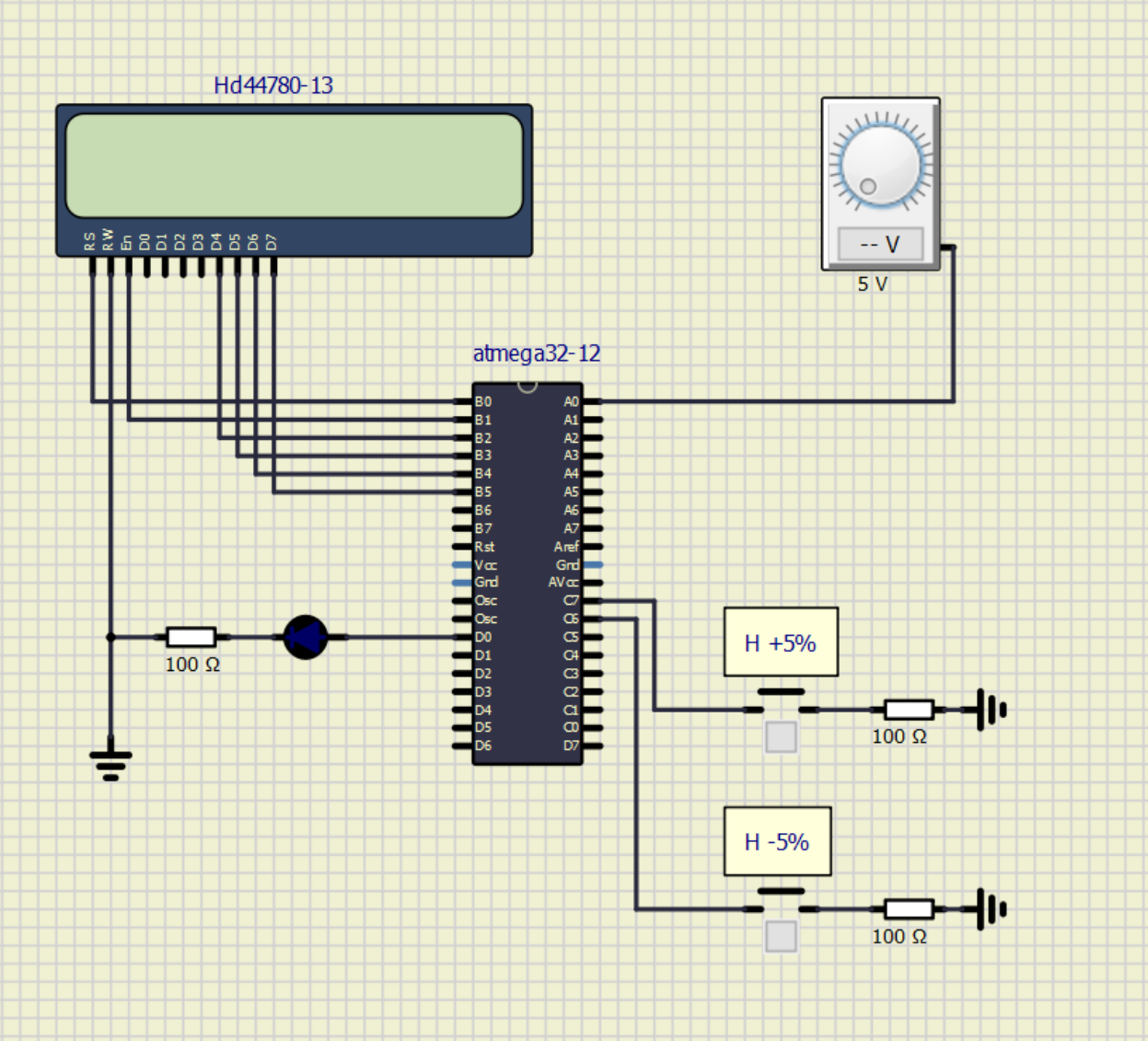
Łukasz Chwistek

Nr. albumu: 243662

1. Wstęp

Stworzono regulator dwustawny z histerezą wyświetlający aktualny stan wyjścia z obiektu na wyświetlaczu. Za pomocą dwóch przycisków można zmieniać histerezę o 5%. Dioda LED zapala się po osiągnięciu wartości 50% + histereza maksymalnej wartości, a gaśnie dla wartości niższych niż 50% - histereza maks. wartości.

2. Schemat układu



3. Kod programu

3.1. biblioteka zarządzania wyświetlaczem LCD

```
11  //-------------------------------------------------------------------------------------------------
12
13  #include <avr/io.h>
14  #include <util/delay.h>
15
16  //-------------------------------------------------------------------------------------------------
17  //
18  // Konfiguracja sygna��w steruj�cych wy�wietlaczem.
19  // Mo�na zmieni� stosownie do potrzeb.
20  //
21  //-------------------------------------------------------------------------------------------------
22  #define LCD_RS_DIR      DDRB
23  #define LCD_RS_PORT     PORTB
24  #define LCD_RS          (1 << PB0)
25
26  #define LCD_E_DIR       DDRB
27  #define LCD_E_PORT      PORTB
28  #define LCD_E           (1 << PB1)
29
30  #define LCD_DB4_DIR     DDRB
31  #define LCD_DB4_PORT    PORTB
32  #define LCD_DB4         (1 << PB2)
33
34  #define LCD_DB5_DIR     DDRB
35  #define LCD_DB5_PORT    PORTB
36  #define LCD_DB5         (1 << PB3)
37
38  #define LCD_DB6_DIR     DDRB
39  #define LCD_DB6_PORT    PORTB
40  #define LCD_DB6         (1 << PB4)
41
42  #define LCD_DB7_DIR     DDRB
43  #define LCD_DB7_PORT    PORTB
44  #define LCD_DB7         (1 << PB5)
45
46  //-------------------------------------------------------------------------------------------------
47  //
48  // Instrukcje kontrolera Hitachi HD44780
49  //
50  //-------------------------------------------------------------------------------------------------
51
52  #define HD44780_CLEAR                   0x01
```

```
//------------------------------------------------------------------------------------------------

#define HD44780_CLEAR                   0x01

#define HD44780_HOME                    0x02

#define HD44780_ENTRY_MODE              0x04
    #define HD44780_EM_SHIFT_CURSOR     0
    #define HD44780_EM_SHIFT_DISPLAY    1
    #define HD44780_EM_DECREMENT        0
    #define HD44780_EM_INCREMENT        2

#define HD44780_DISPLAY_ONOFF           0x08
    #define HD44780_DISPLAY_OFF         0
    #define HD44780_DISPLAY_ON          4
    #define HD44780_CURSOR_OFF          0
    #define HD44780_CURSOR_ON           2
    #define HD44780_CURSOR_NOBLINK      0
    #define HD44780_CURSOR_BLINK        1

#define HD44780_DISPLAY_CURSOR_SHIFT    0x10
    #define HD44780_SHIFT_CURSOR        0
    #define HD44780_SHIFT_DISPLAY       8
    #define HD44780_SHIFT_LEFT          0
    #define HD44780_SHIFT_RIGHT         4

#define HD44780_FUNCTION_SET            0x20
    #define HD44780_FONT5x7             0
    #define HD44780_FONT5x10            4
    #define HD44780_ONE_LINE            0
    #define HD44780_TWO_LINE            8
    #define HD44780_4_BIT               0
    #define HD44780_8_BIT               16

#define HD44780_CGRAM_SET               0x40

#define HD44780_DDRAM_SET               0x80

//------------------------------------------------------------------------------------------------
//
// Deklaracje funkcji
//
//------------------------------------------------------------------------------------------------

void LCD_WriteCommand(unsigned char);
void LCD_WriteData(unsigned char);
void LCD_WriteText(char *);
void LCD_GoTo(unsigned char, unsigned char);
void LCD_Clear(void);
void LCD_Home(void);
void LCD_Initialize(void);

//------------------------------------------------------------------------------------------------
//
// Koniec pliku HD44780.h
//
//------------------------------------------------------------------------------------------------
```

3.2. Wykonany program – timer, odczytywanie sygnału analogowego

```c
/*
 *  Cwiczenie 4
 *  regulator dwustawny z histerezą oraz wyswietlanie stanu na LCD
 *
 *  Created on: 14 luty 2021
 *      Author: Lukasz 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"
#include "HD44780.h"

static void ADC_init()
{
    ADMUX = (1<<REFS0);                                         //napiecie referencyjne AVcc 5V
    ADCSRA = (1<<ADPS0) | (1<< ADPS1) | (1<< ADPS2) | (1<<ADEN); //prescaler 128,  enable conversion
}

static uint16_t ADC_10bit()
{
    ADCSRA|= (1<<ADSC);                                 //start conversion
    while (ADCSRA & (1<<ADSC));                         //wykonanie pomiaru
    return ADC;
}

void write (int16_t in, int16_t out, float histeresis)  //wypisywanie na wyswietlaczu
{
    float Vmin=0.0f;
    float Vmax=5.0f;
    char input[16], output[16], hist[16];
    float value=((float)in/1024.0f)*(Vmax-Vmin)+Vmin;
    dtostrf(value,0,3,input);
    dtostrf(out,0,3,output);
    dtostrf(histeresis,0,3,hist);
    LCD_Clear();
    LCD_WriteText(input);
    LCD_WriteText(" V");
```

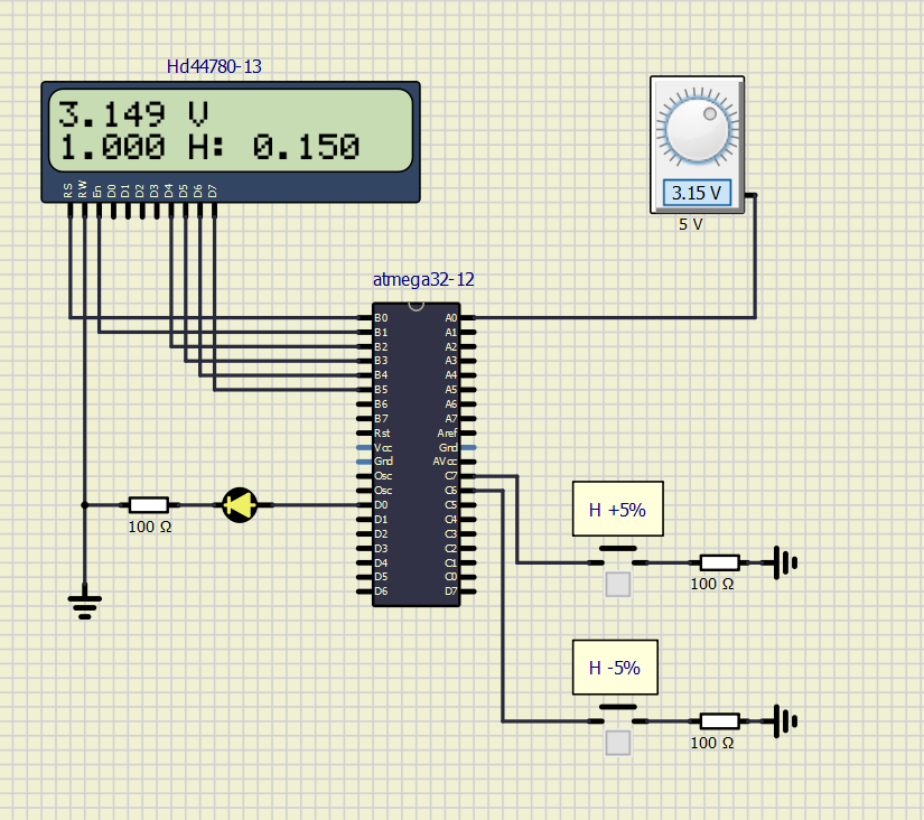
```
    LCD_GoTo(0,1);
    LCD_WriteText(output);
    LCD_WriteText(" H: ");
    LCD_WriteText(hist);
}

int main()
{
    DDRD|=(1<<DDD0);
    PORTC |= (1 << PC6) | (1 << PC7);
    int16_t in;
    int16_t out;
    ADC_init();
    LCD_Initialize();
    const float setpoint = 0.5f;
    volatile float histeresis = 0.1f;
    float on = setpoint + histeresis;
    float off = setpoint - histeresis;
    while(1)
    {
        if(!(PINC & (1<<PIN7)))
        {
            histeresis+=0.05f;
        }
        if (!(PINC & (1<<PIN6)))
        {
            if(histeresis>0)
            {
                histeresis-=0.05f;
            }
        }
        on = setpoint + histeresis;
        off = setpoint - histeresis;

        int16_t outon = on*1024;
        int16_t outoff = off*1024;

        in=ADC_10bit();
        if(in > outon)
        {
            out=1;
            PORTD |= (1 << PD0);
        }
        else if (in < outoff)
        {
            out=0;
            PORTD &= ~(1 << PD0);
        }
        write(in,out,histeresis);
        _delay_ms(1000);
    }
}
```

4. Wyniki



5. Wnioski

Dioda LED zapala się i gaśnie przy odpowiednio wysokich i niskich wartościach napięcia. Aktualne stany układu są poprawnie wyświetlane na wyświetlaczu LCD.