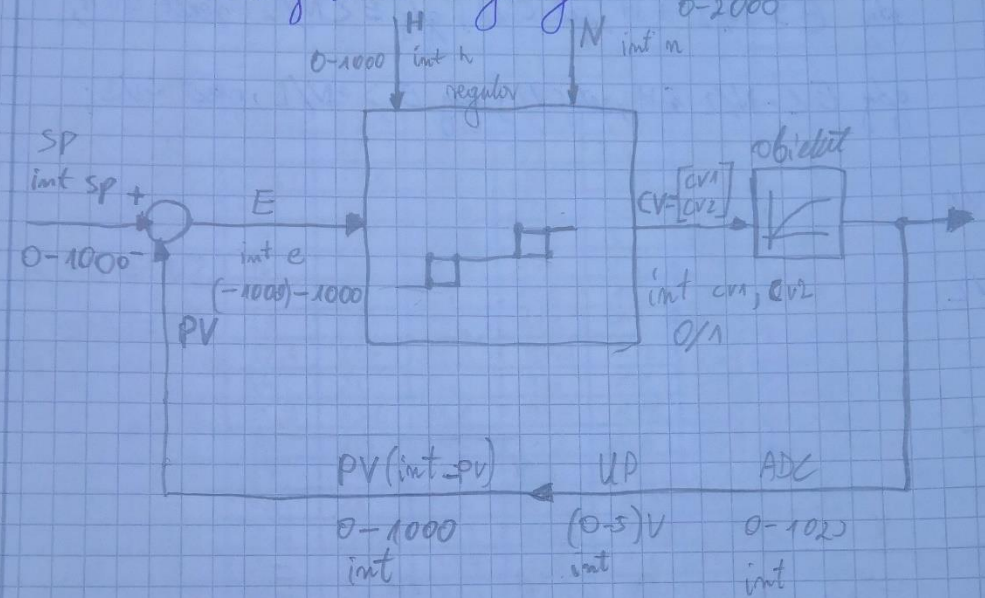
LAB3

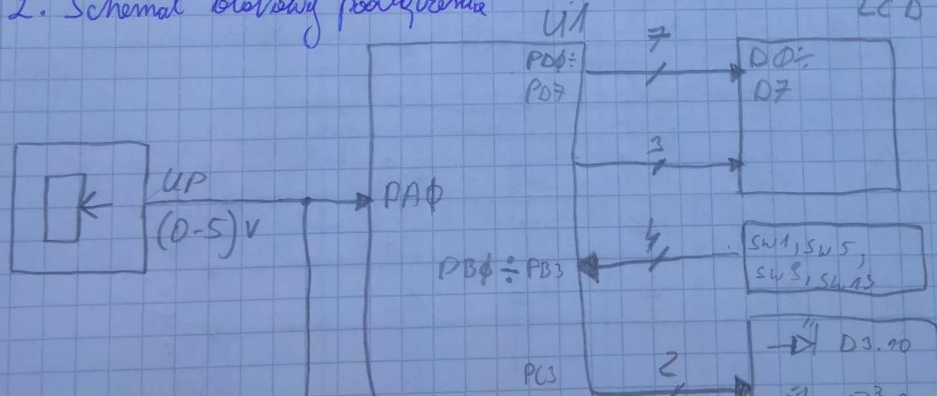
Jan Bronicki 249011

REGULATOR TRÓJSTANOWY

1. Schemat blokowy układu regulacji



2. Schemat blokowy podłączenia



3. Regulator trójstawny

$E = SP - PV$

gdy $E > N/2 + H$, set CV1; gdy $E < N/2$, reset CV1;

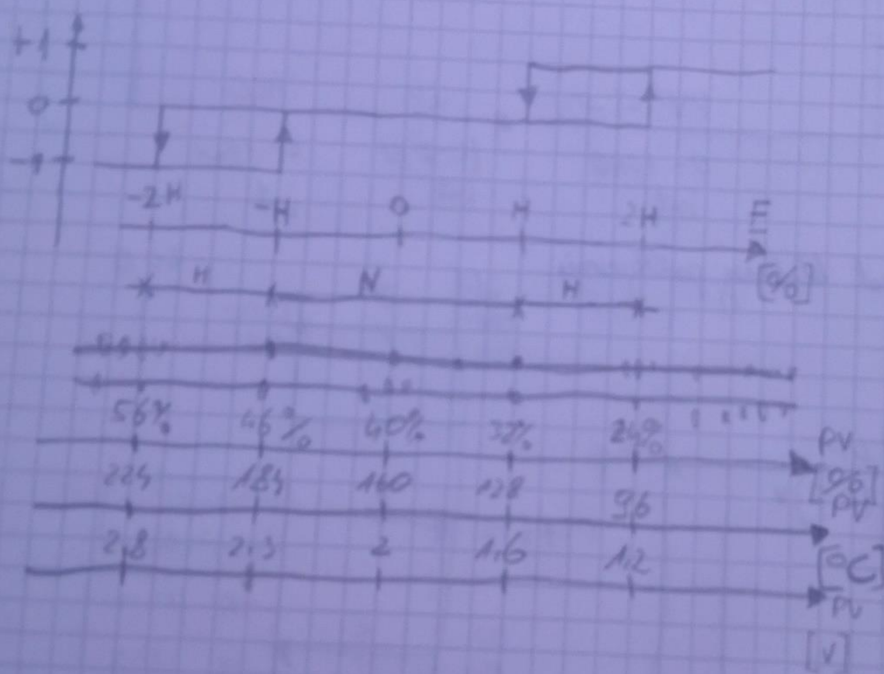
gdy $E < -N/2 - H$, set CV2; gdy $E > -N/2$, reset CV2;

CV1

CV2

SP=50%

H=8%

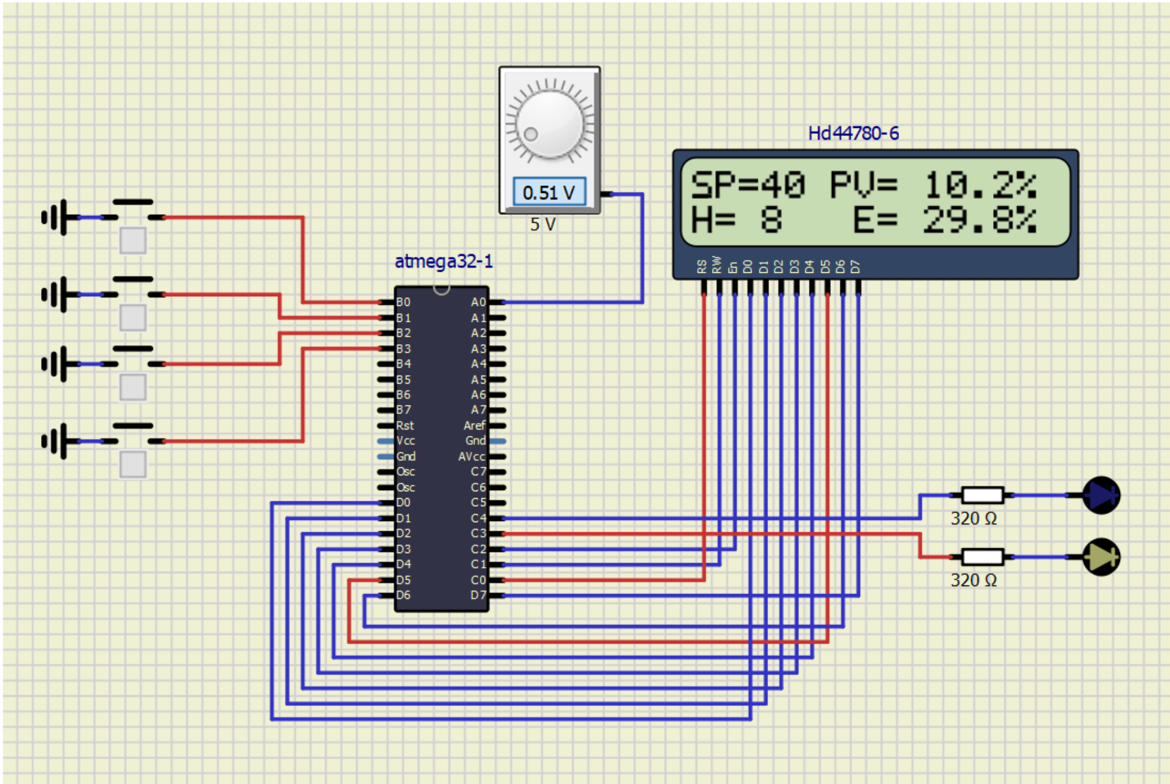


4. Tabela pomiarowa , Badanie regulatora 3-stawnego

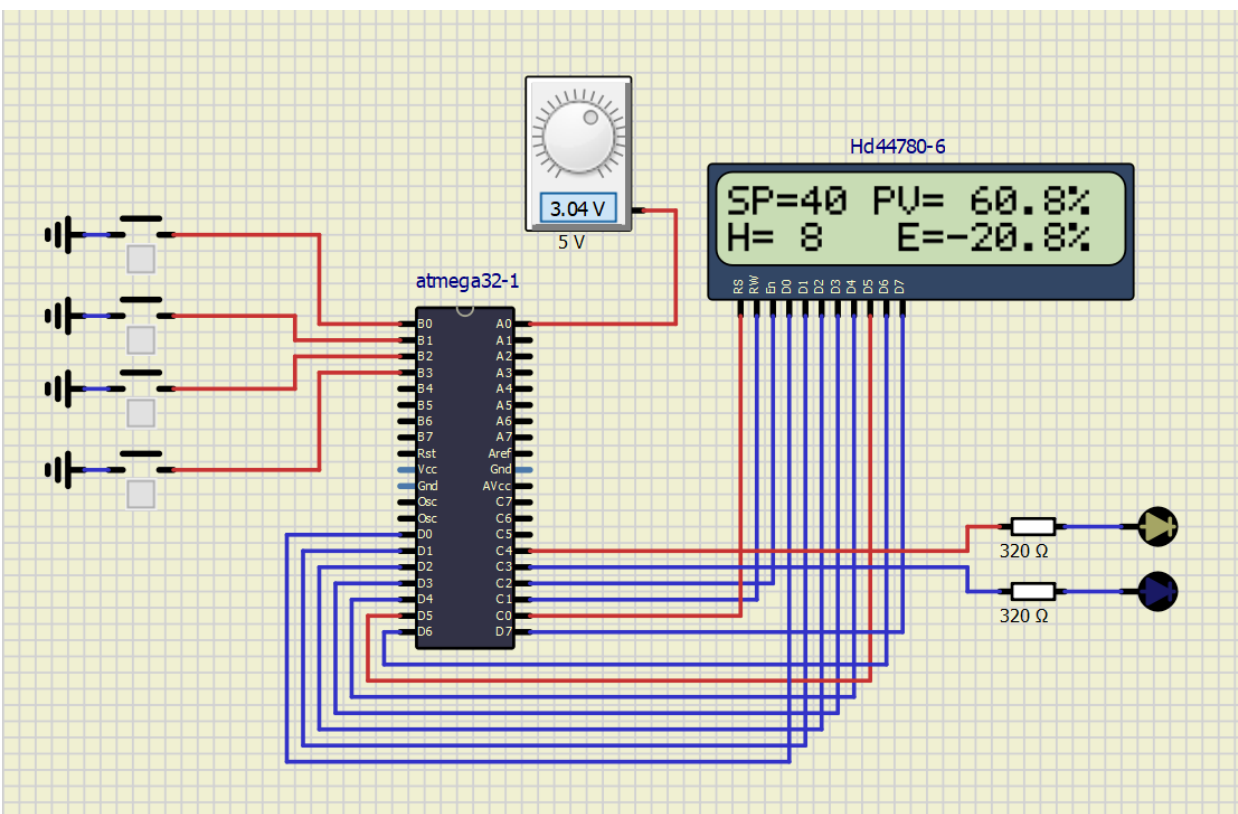
SP = 40 % , H = 8 % , N = 2H , zakres (0-400°C) / (0-5) V

E [H]	E [%]	E [°C]	PV [%]	PV [ADC]	PV [°C]	PV [V]	POMIAR PV [%]	STAN DIOD 3.10	STAN DIODY 3.9
-2.50	-20	-80	60	614	240	3	58,9	V	X
-2,00	-16	-64	56	573	224	2,8	55,9	V	X
-1,50	-12	-48	52	532	208	2,6	51,9	V	X
-1,05	-8,4	-33,6	48,4	495	193,6	2,42	48,3	V	X
-1,00	-8	-32	48	491	192	2,4	47,9	X	X
-0,95	-7,6	-30,4	47,6	487	190,4	2,38	47,5	X	X
-0,50	-4	-16	44	450	176	2,2	43,9	X	X
0,00	0	0	40	409	160	2	39,9	X	X
0,50	4	16	30	368	144	1,8	35,9	X	X
1,00	8	32	32	327	128	1,6	31,9	X	X
1,50	12	48	28	286	112	1,4	27,9	X	X
1,95	15,6	62,4	28,4	250	97,6	1,22	24,3	X	X
2,00	16	64	24	246	96	1,2	23,9	X	V
2,05	16,4	65,6	23,6	241	94,4	1,18	23,5	X	V
2,50	20	80	20	205	80	1,0	19,9	X	V
2,00	16	64	24	246	96	1,2	23,9	X	V
1,50	12	48	28	286	112	1,4	27,9	X	V
1,05	8,4	33,6	31,6	323	126,4	1,58	31,5	X	V
1,00	8	32	32	327	128	1,6	31,9	X	V
0,95	7,6	30,4	32,4	331	129,6	1,62	32,3	X	X
0,50	4	16	36	368	144	1,8	35,9	X	X
0,00	0	0	40	409	160	2	39,9	X	X
-0,50	-4	-16	44	450	176	2,2	43,9	X	X
-1,00	-8	-32	48	491	192	2,4	47,9	X	X
-1,50	-12	-48	52	532	208	2,6	51,9	X	X
-1,95	-15,6	-62,4	55,6	569	222,4	2,78	55,5	X	X
-2,00	-16	-64	56	573	224	2,8	55,9	X	X
-2,05	-16,4	-65,6	56,4	577	225,6	2,82	56,3	V	X
-2,50	-20	-80	60	614	240	3	59,9	V	X

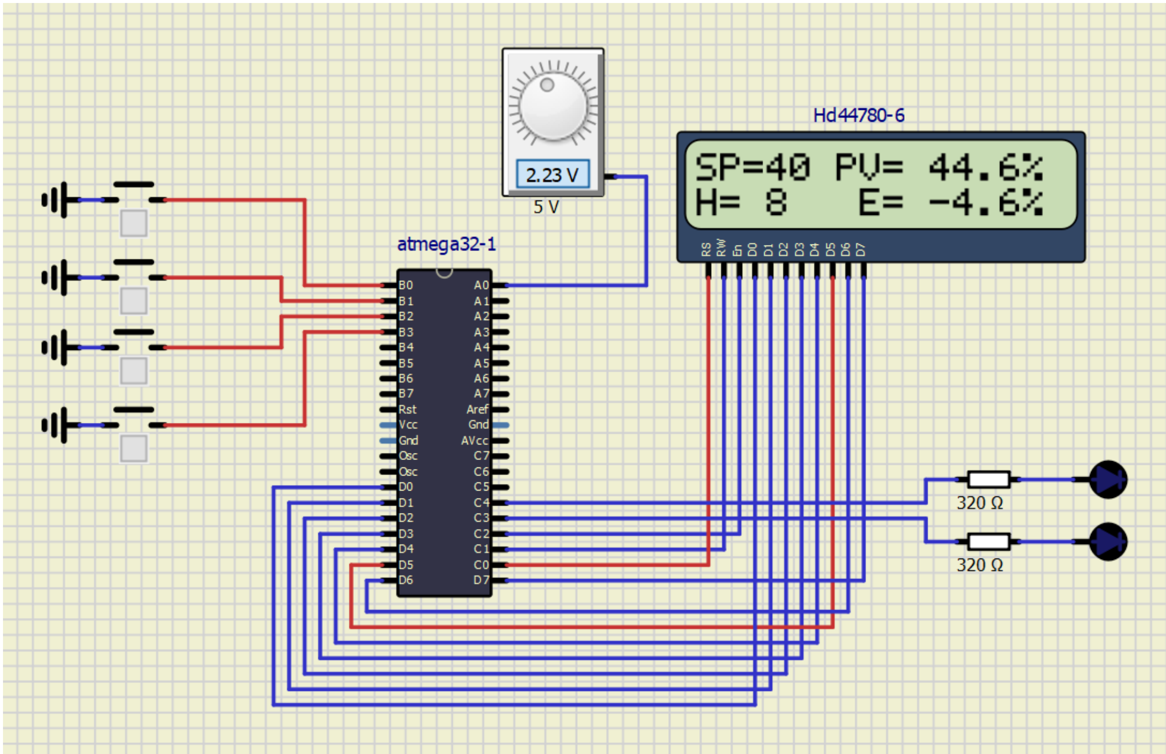
SYMULACJA:



0.51 V
5 V
Hd44780-6
SP=40 PV= 10.2%
H= 8 E= 29.8%
atmega32-1
320 Ω
320 Ω



3.04 V
5 V
Hd44780-6
SP=40 PV= 60.8%
H= 8 E=-20.8%
atmega32-1
320 Ω
320 Ω



KOD:

```
/*************************************/
/*              ARE 2008            */
/*       e-mail: biuro@are.net.pl    */
/*       www    : are.net.pl         */
/*************************************/

// Jan Bronicki 249011
// Borys Staszczak 248958

#define __AVR_ATmega32__
#define F_CPU 8000000UL

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

```c
void delay_ms(int ms)
{
    volatile long unsigned int i;
    for (i = 0; i < ms; i++)
        _delay_ms(1);
}

void delay_us(int us)
{
    volatile long unsigned int i;
    for (i = 0; i < us; i++)
        _delay_us(1);
}

#define RS 0
#define RW 1
#define E 2

void LCD2x16_init(void)
{
    PORTC &= ~(1 << RS);
    PORTC &= ~(1 << RW);

    PORTC |= (1 << E);
    PORTD = 0x38; // dwie linie, 5x7 punktow
    PORTC &= ~(1 << E);
    _delay_us(120);

    PORTC |= (1 << E);
    PORTD = 0x0e; // wlacz wyswietlacz, kursor, miganie
    PORTC &= ~(1 << E);
    _delay_us(120);

    PORTC |= (1 << E);
    PORTD = 0x06;
    PORTC &= ~(1 << E);
    _delay_us(120);
```

```
}

void LCD2x16_clear(void)
{
    PORTC &= ~(1 << RS);
    PORTC &= ~(1 << RW);

    PORTC |= (1 << E);
    PORTD = 0x01;
    PORTC &= ~(1 << E);
    delay_ms(120);
}

void LCD2x16_putchar(int data)
{
    PORTC |= (1 << RS);
    PORTC &= ~(1 << RW);

    PORTC |= (1 << E);
    PORTD = data;
    PORTC &= ~(1 << E);
    _delay_us(120);
}

void LCD2x16_pos(int wiersz, int kolumna)
{
    PORTC &= ~(1 << RS);
    PORTC &= ~(1 << RW);

    PORTC |= (1 << E);
    delay_ms(1);
    PORTD = 0x80 + (wiersz - 1) * 0x40 + (kolumna - 1);
    delay_ms(1);
    PORTC &= ~(1 << E);
    _delay_us(120);
}
```

```c
// Set point (in 0.1%)
int _sp = 400;
// Histereza (in 0.1%)
int _h = 80;
// Nieczułość (in 0.1%)
int _n = 160;
// Error value
int _e;
// Integer part of the error
int int_e;
// Decimal value of the error
int dec_e;
// Whole process value (in 0-1023 range)
float process_value;
// Process value with decimal part
int _pv;
// Integer part of process value
int _ipv;
// Decimal part of process value
int _decpv;


int main(void)
{

    char tmp[16];

    int i;

    DDRD = 0xff;
    PORTD = 0x00;
    DDRC = 0xff;
    PORTC = 0x00;
    DDRB = 0x00;
    PORTB = 0xff;

    _delay_ms(500);
```

```
LCD2x16_init();
LCD2x16_clear();

ADMUX = 0x40;
ADCSRA = 0xe0;

while (1)
{
    // Start an ADC conversion by setting ADSC bit (bit 6)
    ADCSRA = ADCSRA | (1 << ADSC);

    // Wait until the ADSC bit has been cleared
    while (ADCSRA & (1 << ADSC))
        ;

    //_n=_h+_h;
    process_value = ADC;
    _pv = (process_value / 1023.0) * 1000;
    _ipv = _pv / 10;
    _decpv = _pv % 10;

    _e = _sp - _pv;
    int_e = _e / 10;
    dec_e = _e % 10;

    // LED CV1 ON
    if (_e > ((_n/2)+_h))
    {
        PORTC = ~(0x01 << 4);
    }

    // LED OFF
    if(_e < _n/2 && _e > -_n/2)
    {
        PORTC=(0x00);
    }
```

```
// LED CV2 ON
if (_e < ((-_n/2)-_h))
{
    PORTC = ~(0x01 << 3);
}

if (!(PINB & (8 << PB0)))
{
    _sp = 50;
}
if (!(PINB & (4 << PB0)))
{
    _sp = 40;
}
if (!(PINB & (2 << PB0)))
{
    _h = 8;
    _n = 16;
}
if (!(PINB & (1 << PB0)))
{
    _h = 10;
    _n = 20;
}


LCD2x16_pos(1, 1);
sprintf(tmp, "SP=%2d PV=%3d.%1d%% ", _sp/10, _ipv, abs(_decpv));
for (i = 0; i < 16; i++)
{
    LCD2x16_putchar(tmp[i]);
}

LCD2x16_pos(2, 1);
sprintf(tmp, "H=%2d   E=%3d.%1d%% ", _h/10, int_e, abs(dec_e));
for (i = 0; i < 16; i++)
```

```
        {
            LCD2x16_putchar(tmp[i]);
        }
        delay_ms(500);
    }

    return 0;
}
```