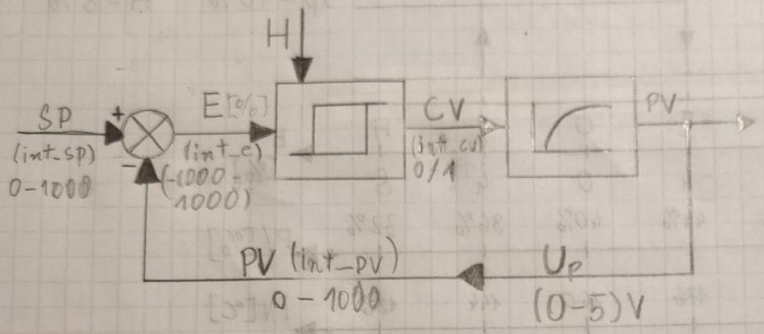
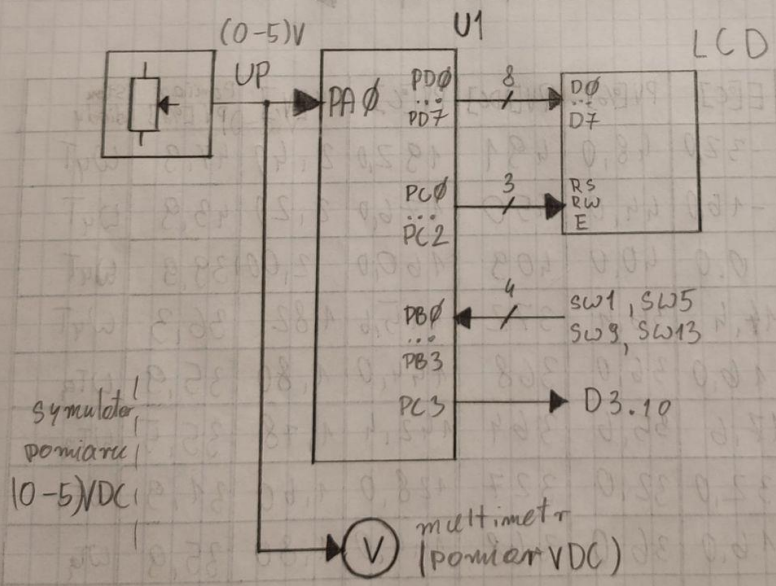
Borys Staszczak 248958

Temat: Badanie regulatora dwustawnego

1. Schemat blokowy typowego układu regulacji



2. Schemat blokowy podłączenia sygnałów w układzie do badania regulatora

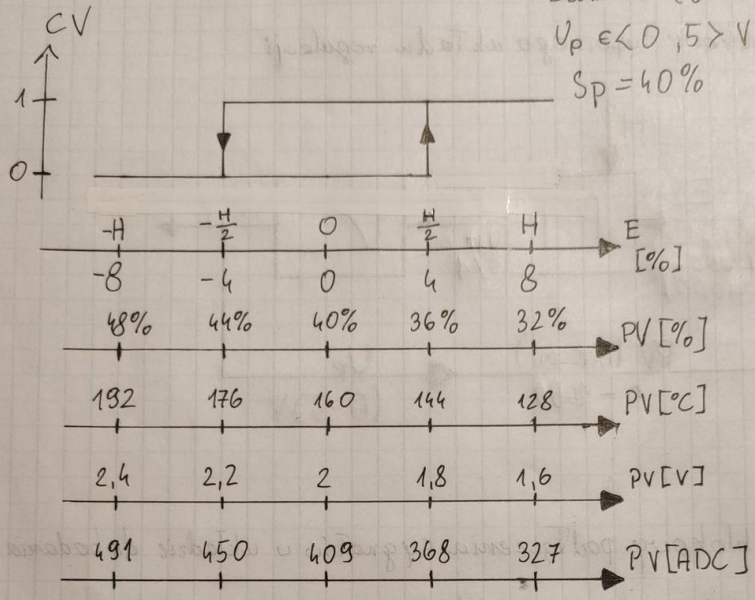


3. Charakterystyka regulatora dwustawnego

Zakres: (0 – 400)°C

$U_p \in \langle 0,5 \rangle V$

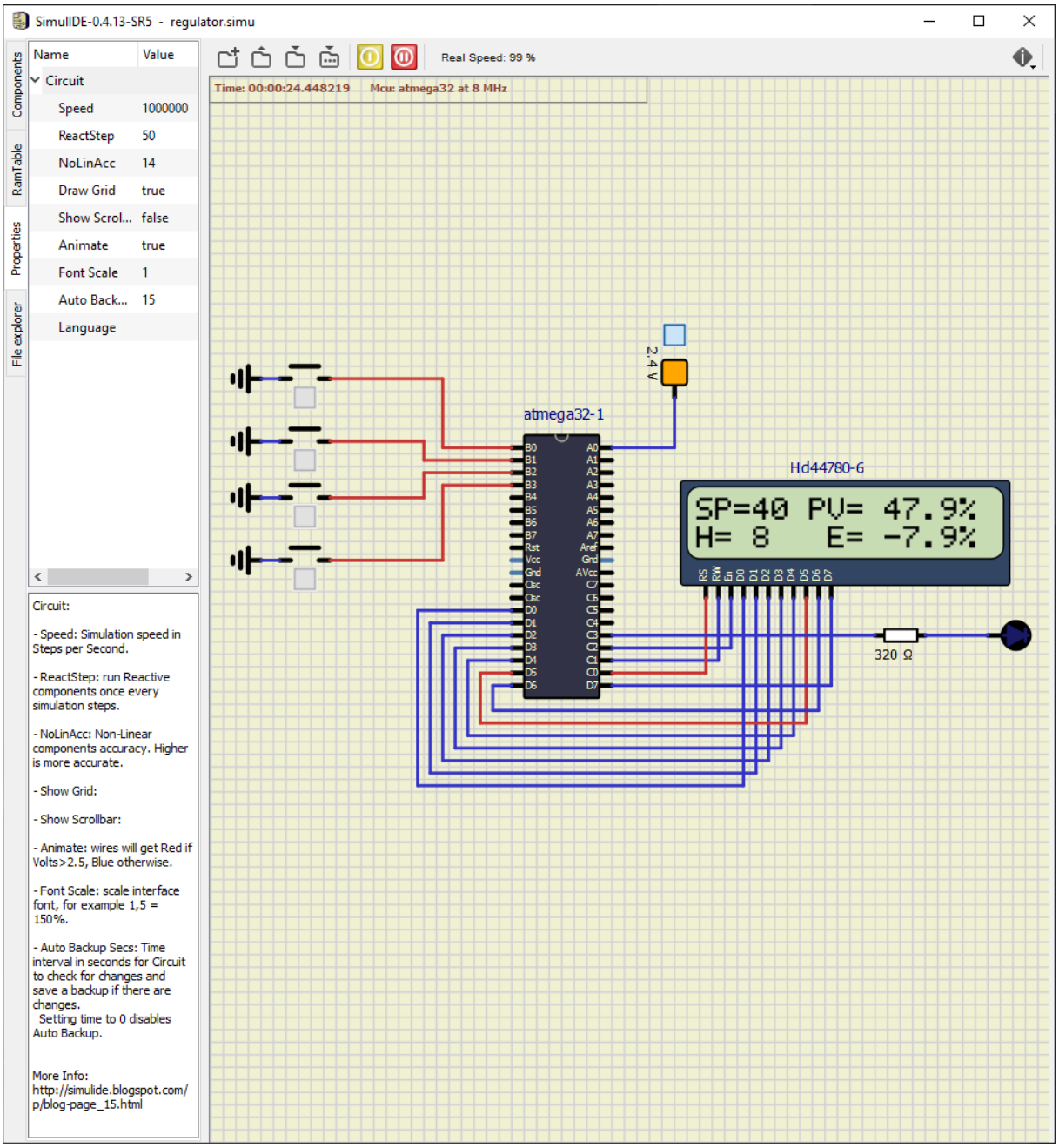
$Sp = 40\%$ $H = 8\%$



4.

E [H]	E [%]	E [°C]	PV [%]	PV [ADC]	PV [°C]	PV [V]	Pomiar PV [%]	Stan diody
-1,00	-8,00	-32,0	48,0	491	192,0	2,40	47,9	$W_4\bar{T}$
-0,50	-4,00	-16,0	44,0	450	176,0	2,20	43,9	$W_4\bar{T}$
0,00	0,00	0,0	40,0	409	160,0	2,00	39,9	$W_4\bar{T}$
0,45	3,60	14,4	36,4	372	145,6	1,82	36,3	$W_4\bar{T}$
0,50	4,00	16,0	36,0	368	144,0	1,80	35,9	WT_4
0,55	4,40	17,6	35,6	364	142,4	1,78	35,5	WT_4
1,00	8,00	32,0	32,0	327	128,0	1,60	31,9	WT_4
0,50	4,00	16,0	36,0	368	144,0	1,80	35,9	WT_4
0,00	0,00	0,0	40,0	409	160,0	2,00	39,9	WT_4
-0,45	-3,60	-14,4	43,6	446	174,4	2,18	43,5	$W_4\bar{T}$
-0,50	-4,00	-16,0	44,0	450	176,0	2,20	43,9	$W_4\bar{T}$
-0,55	-4,40	-17,6	44,4	454	177,6	2,22	44,3	$W_4\bar{T}$
-1,00	-8,00	-32,0	48,0	491	192,0	2,40	47,9	$W_4\bar{T}$

Bonus stałe szczęście 248858


SimulIDE-0.4.13-SR5 - regulator.simu
Components
RamTable
Properties
File explorer
Name
Value
Circuit
Speed 1000000
ReactStep 50
NoLinAcc 14
Draw Grid true
Show Scrol... false
Animate true
Font Scale 1
Auto Back... 15
Language
Circuit:
- Speed: Simulation speed in Steps per Second.
- ReactStep: run Reactive components once every simulation steps.
- NoLinAcc: Non-Linear components accuracy. Higher is more accurate.
- Show Grid:
- Show Scrollbar:
- Animate: wires will get Red if Volts>2.5, Blue otherwise.
- Font Scale: scale interface font, for example 1,5 = 150%.
- Auto Backup Secs: Time interval in seconds for Circuit to check for changes and save a backup if there are changes.
Setting time to 0 disables Auto Backup.
More Info:
http://simulide.blogspot.com/p/blog-page_15.html
Real Speed: 99 %
Time: 00:00:24.448219
Mcu: atmega32 at 8 MHz
5 V
atmega32-1
Hd44780-6
SP=40 PV= 47.9%
H= 8 E= -7.9%
320 Ω

```
// Set point (in %)
int set_point = 40;
// Histereza (in %)
int _h = 8;
// 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 full_process_value;
// Integer part of process value
int int_process_value;
// Decimal part of process value
int dec_process_value;

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))
            ;

        process_value = ADC;

        full_process_value = (process_value / 1023.0) * 1000;
        int_process_value = full_process_value / 10;
        dec_process_value = full_process_value % 10;
```

```
147
148         _e = (set_point * 10) - full_process_value;
149         int_e = _e / 10;
150         dec_e = _e % 10;
151
152
153         // LED On
154         if (_e > (_h / 2))
155         {
156             PORTC = ~(0x01 << 5);
157         }
158
159         // LED Off
160         if (_e < -(_h / 2))
161         {
162             PORTC = (0x01 << 5);
163         }
164
165         if (!(PINB & (8 << PB0)))
166         {
167             set_point = 50;
168         }
169         if (!(PINB & (4 << PB0)))
170         {
171             set_point = 40;
172         }
173         if (!(PINB & (2 << PB0)))
174         {
175             _h = 8;
176         }
177         if (!(PINB & (1 << PB0)))
178         {
179             _h = 10;
180         }
181
182         LCD2x16_pos(1, 1);
183         sprintf(tmp, "SP=%2d PV=%3d.%1d%% ", set_point, int_process_value, abs(dec_process_value));
184         for (i = 0; i < 16; i++)
185         {
186             LCD2x16_putchar(tmp[i]);
187         }
188
189         LCD2x16_pos(2, 1);
190         sprintf(tmp, "H=%2d   E=%3d.%1d%%  ", _h, int_e, abs(dec_e));
191         for (i = 0; i < 16; i++)
192         {
193             LCD2x16_putchar(tmp[i]);
194         }
195         delay_ms(500);
196     }
197
198     return 0;
199 }
```