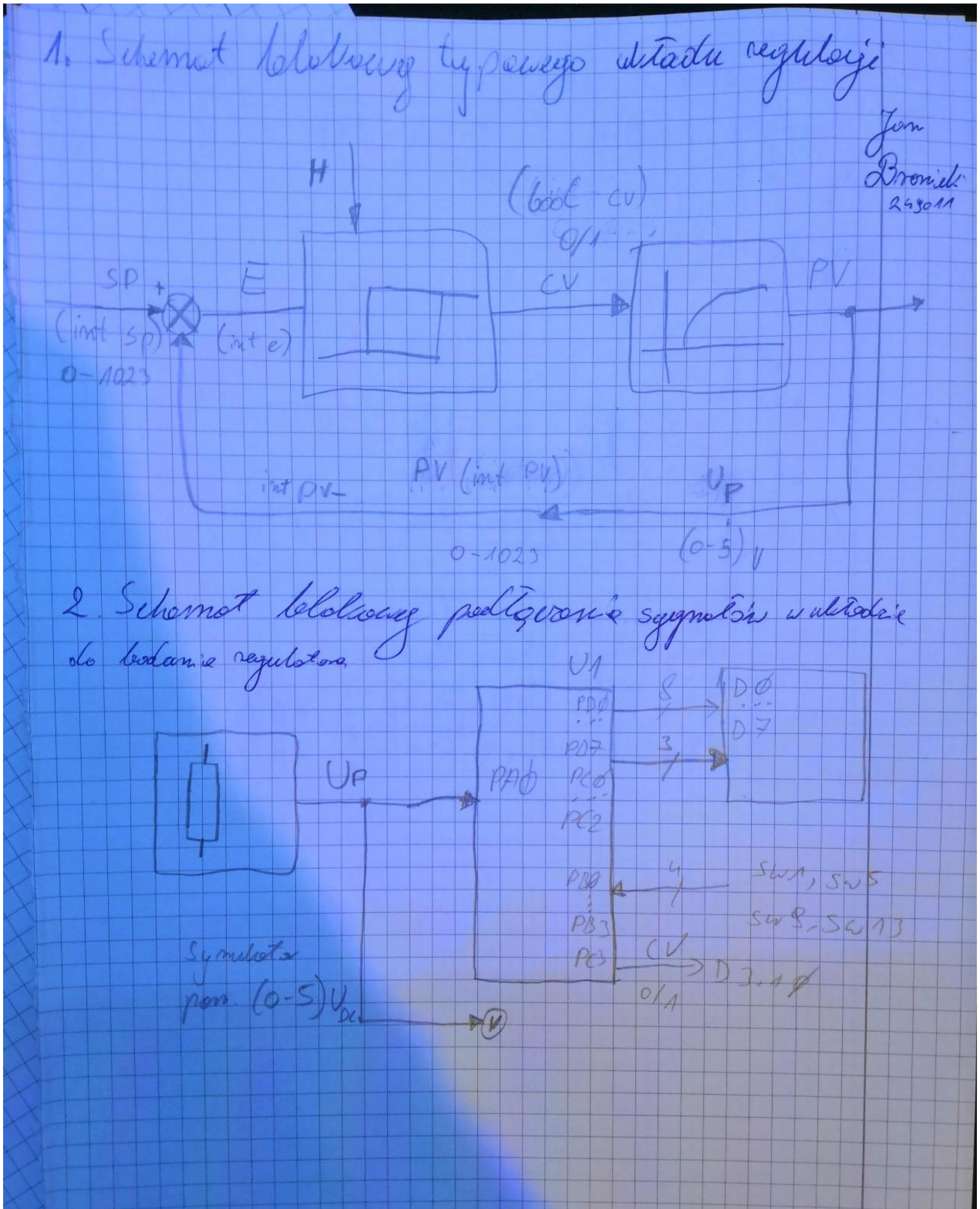
Sprawozdanie Cw1

Jan Bronicki 249011

Borys Staszczak 248958

Temat: Badanie regulatora dwustawnego

1. **Zadanie do wykonania**
   Opracować układ pomiarowy, zmontować układ do badania regulatora, opracować algorytm sterowania w układzie regulacji dwustawnej i przetestować regulator w warunkach laboratoryjnych.
2. **Założenia projektowe**

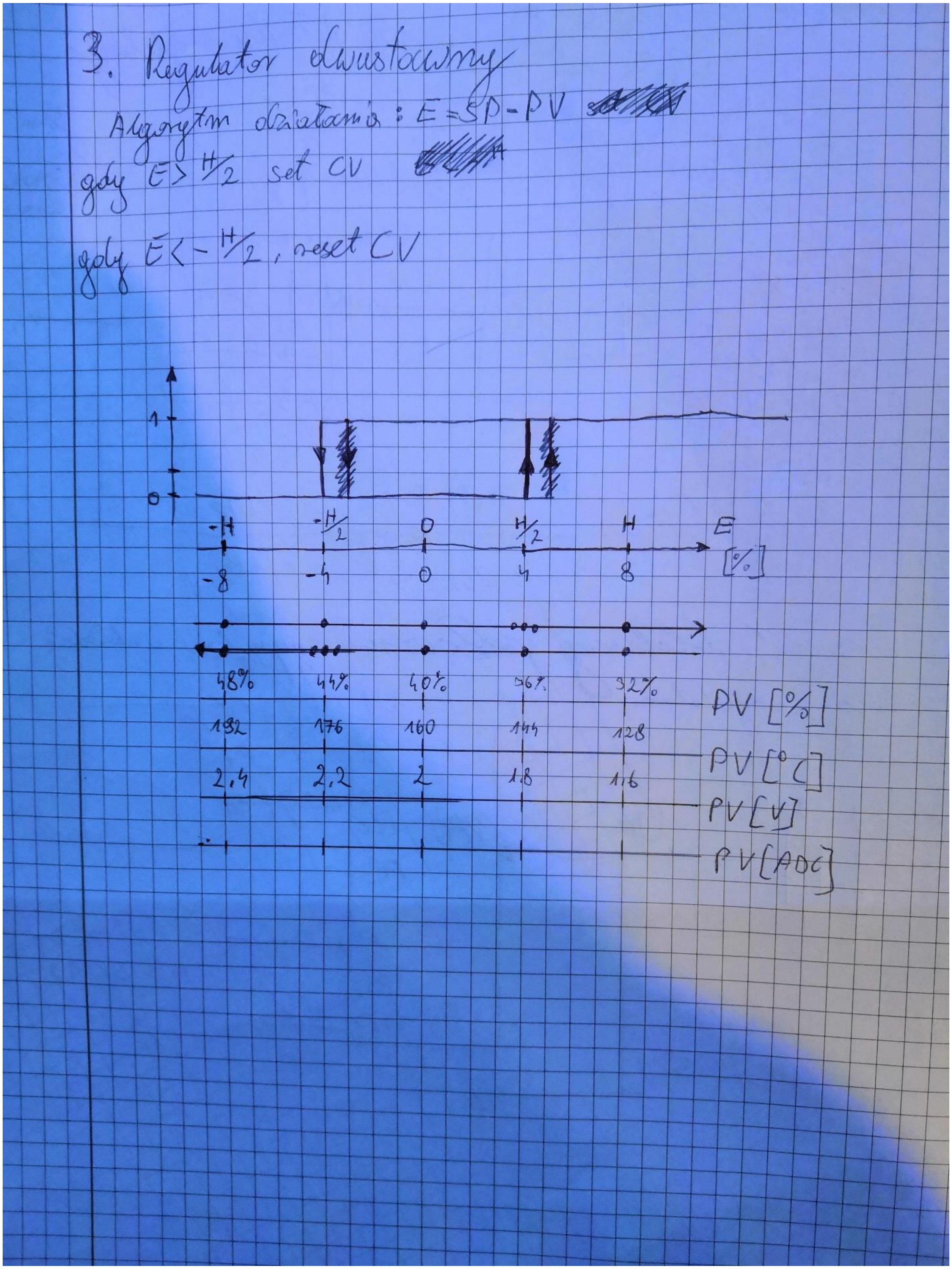


1. Schemat blokowy typowego układu regulacji

Jan Bronicki 249011

2. Schemat blokowy podłączenia sygnałów w układzie do badania regulatora

3. Regulator dwustawny



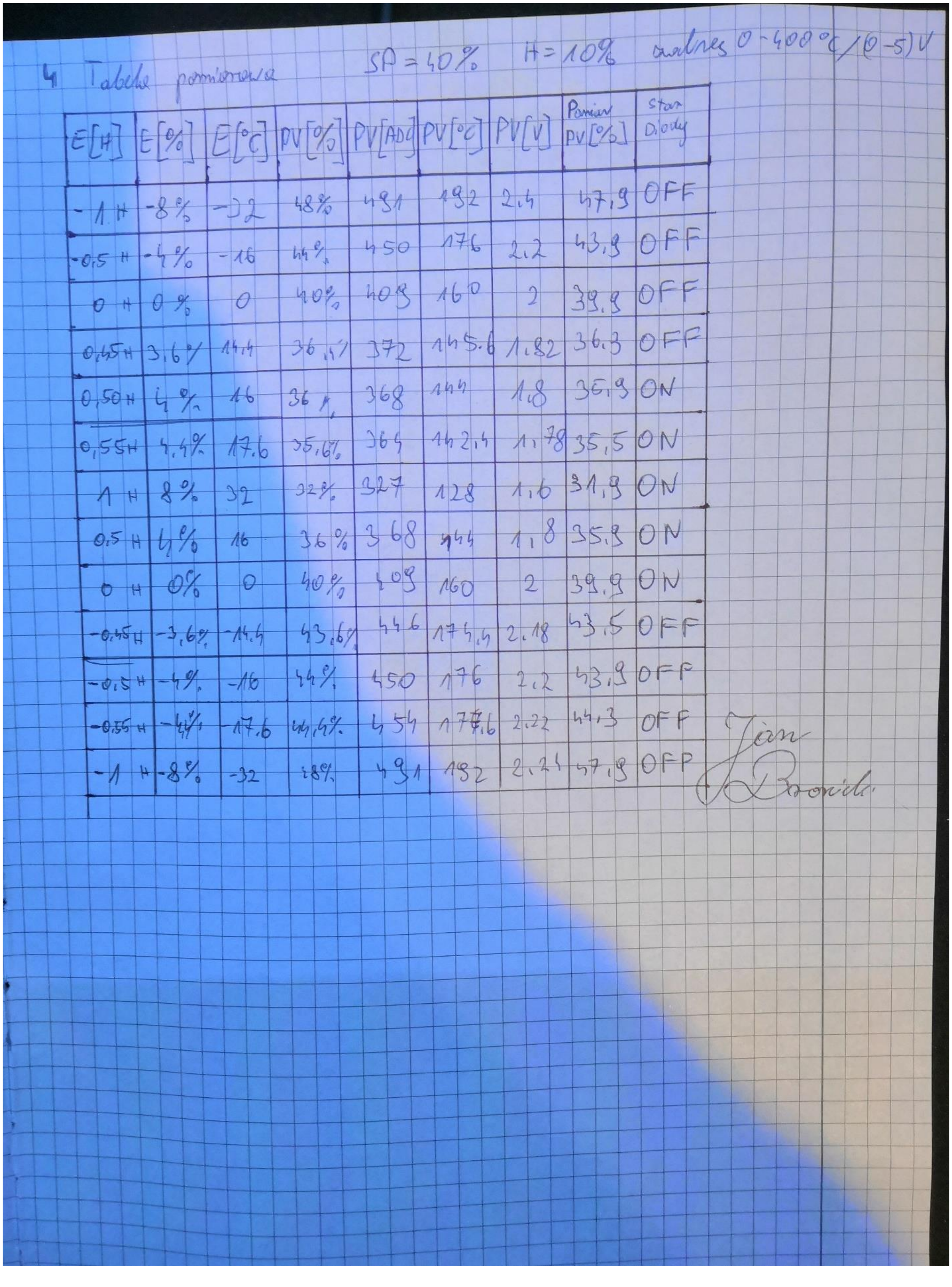
3. Regulator dwustawny

Algorytm działania: $E = SP - PV$

gdy $E > H/2$ set CV

gdy $E < -H/2$, reset CV

4. Tabela pomiarowa (każda grupa oblicza dane do tabeli dla „własnych” danych)



4 Tabela pomiarowa SP = 40% H = 10% zakres 0 - 400 °C / (0 -5) V

E[H]	E[%]	E[°C]	PV[%]	PV[ADC]	PV[°C]	PV[V]	Pomiar PV[%]	Stan Diody
-1 H	-8%	-32	48%	491	192	2,4	47,9	OFF
-0,5 H	-4%	-16	44%	450	176	2,2	43,9	OFF
0 H	0%	0	40%	409	160	2	39,9	OFF
0,45 H	3,6%	14,4	36,4%	372	145.6	1,82	36,3	OFF
0,50 H	4%	16	36%	368	144	1,8	35,9	ON
0,55 H	4,4%	17.6	35,6%	364	142,4	1,78	35,5	ON
1 H	8%	32	32%	327	128	1,6	31,9	ON
0,5 H	4%	16	36%	368	144	1,8	35,9	ON
0 H	0%	0	40%	409	160	2	39,9	ON
-0,45 H	-3,6%	-14,4	43,6%	446	174,4	2.18	43,5	OFF
-0,5 H	-4%	-16	44%	450	176	2,2	43,9	OFF
-0,55 H	-4,4%	-17.6	44,4%	454	177.6	2,22	44,3	OFF
-1 H	-8%	-32	48%	491	192	2,24	47,9	OFP

Jan B[illegible]

5. Uwagi i wnioski

6. Załącznik nr 1: Kod program

```c
/**************************************/
/*              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>

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);
}

// 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();
```

```c

        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;

            _e = (set_point * 10) - full_process_value;
            int_e = _e / 10;
            dec_e = _e % 10;

            // LED On
            if (_e > (_h / 2))
            {
                PORTC = ~(0x01 << 5);
            }

            // LED Off
            if (_e < -(_h / 2))
            {
                PORTC = (0x01 << 5);
            }

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

            LCD2x16_pos(1, 1);
            sprintf(tmp, "SP=%2d PV=%3d.%1d%% ", set_point, int_process_value, abs(dec_process_value));
            for (i = 0; i < 16; i++)
            {
                LCD2x16_putchar(tmp[i]);
            }

            LCD2x16_pos(2, 1);
            sprintf(tmp, "H=%2d   E=%3d.%1d%%  ", _h, int_e, abs(dec_e));
            for (i = 0; i < 16; i++)
            {
                LCD2x16_putchar(tmp[i]);
            }
            delay_ms(500);
        }

        return 0;
    }
```