Marek Lubieniecki

1. Goal of the project

The goal of the project is to design a proof-of-concept version of a control system for an automated safety valve.

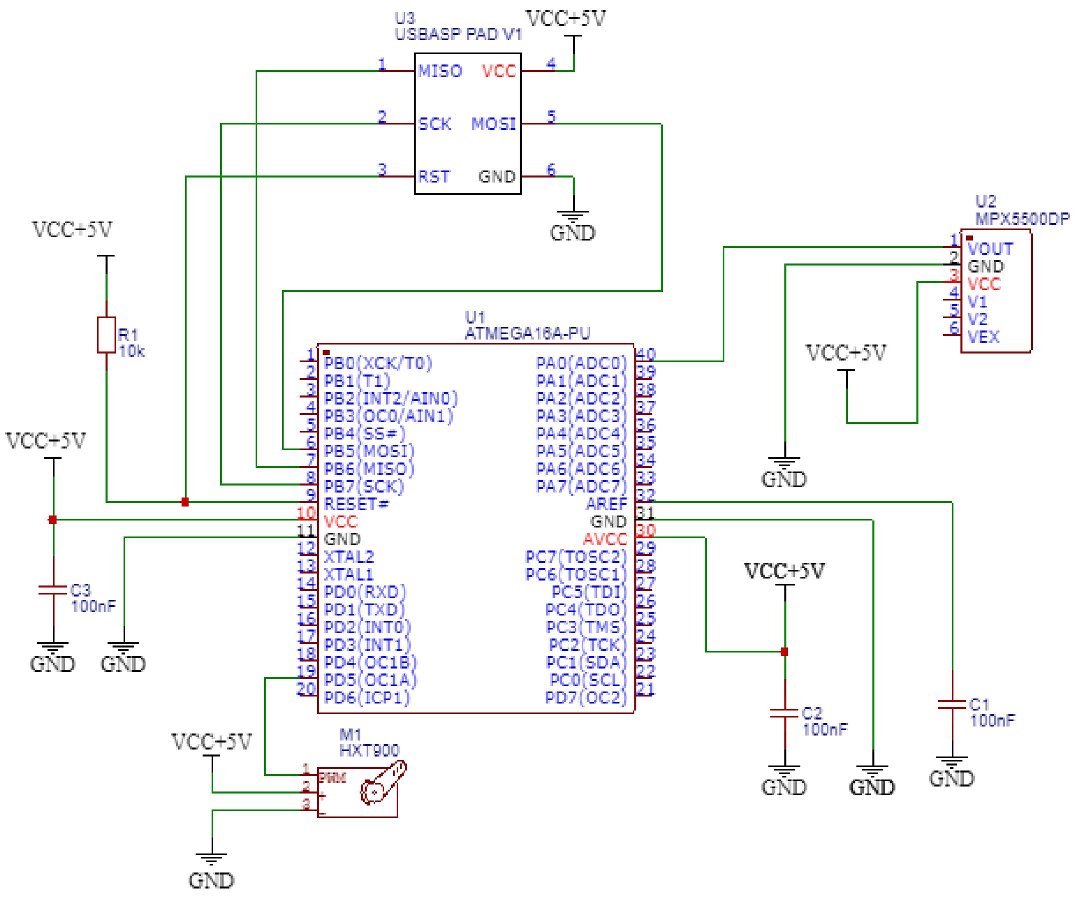
2. How it works

The system consists of four main elements: a valve, a microcontroller, a pressure sensor and a servomechanism. The pressure sensor measures the pressure in the installation and when the pressure is above a certain threshold then the microcontroller sends a signal to the servo to rotate the valve and open it. To measure the pressure a MPX5500DP sensor is used, which outputs an analog voltage signal that is connected to the internal ADC of the Atmega 16A. After voltage measurement the appropriate PWM signal is sent to a Hextronik HXT900 servo that controls the valve. In the first version an internal clock of the Atmega 16A microcontroller is used.

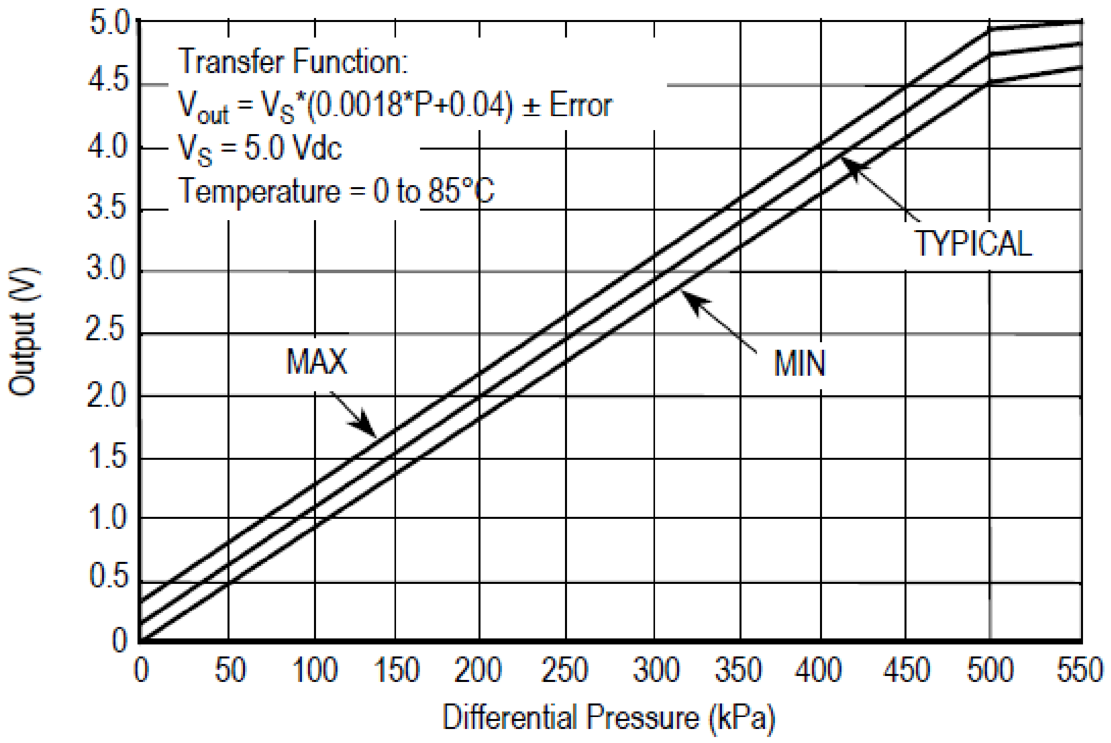
3. Circuit diagram

Elements:

- Atmega16A
- Hextronik HXT900 servo
- MPX5500DP pressure sensor
- USBASP programmer



Calculation of ADC value for valve triggering:



- Chosen pressure 300 kPa
- Voltage:

$$V_{out} = 5V \cdot (0.0018 \cdot 300 + 0.04) = 2.9\ V$$

- ADC value:

$$ADC_{value} = floor\left(\frac{2.9}{5} \cdot 1024\right) = 593$$

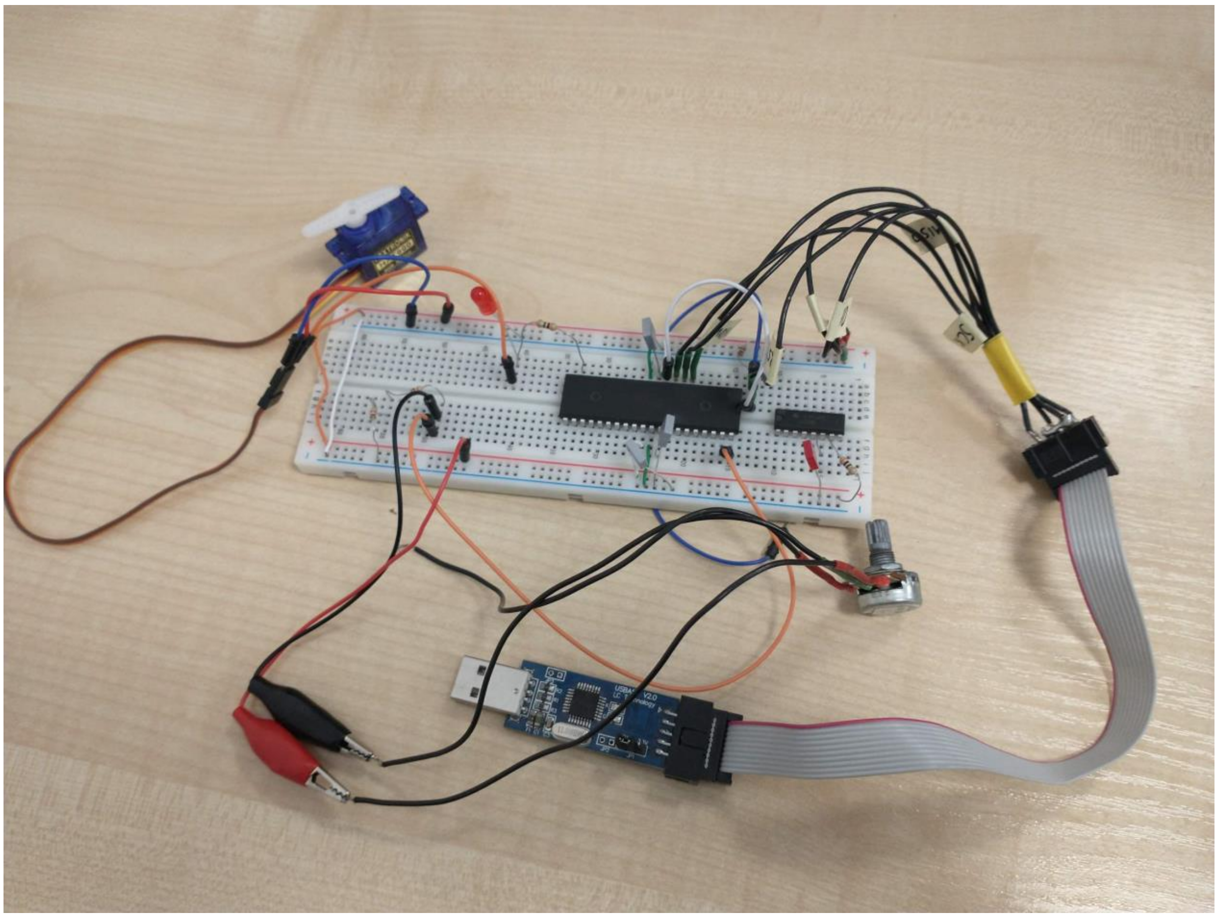
4. Code (written in MicroChip Studio)

```c
/*
 * safety_valve.c
 *
 * Created: 16.05.2022 11:41:52
 * Author : m.lubieniecki
 */
#define F_CPU 8000000UL

#define valve_closed 100 /*PWM pulse width 0.8 ms */
#define valve_opened 250 /*PWM pluse width 2 ms*/

#define reference_voltage 5
#define threshold_adc_value 593 /* 300 kPa calculated for 10bit adc from sensor
datasheet */

#include <avr/io.h>
#include <util/delay.h>

void ADC_Init()
{
	DDRA = 0x0;		/* Make ADC port as input */
```

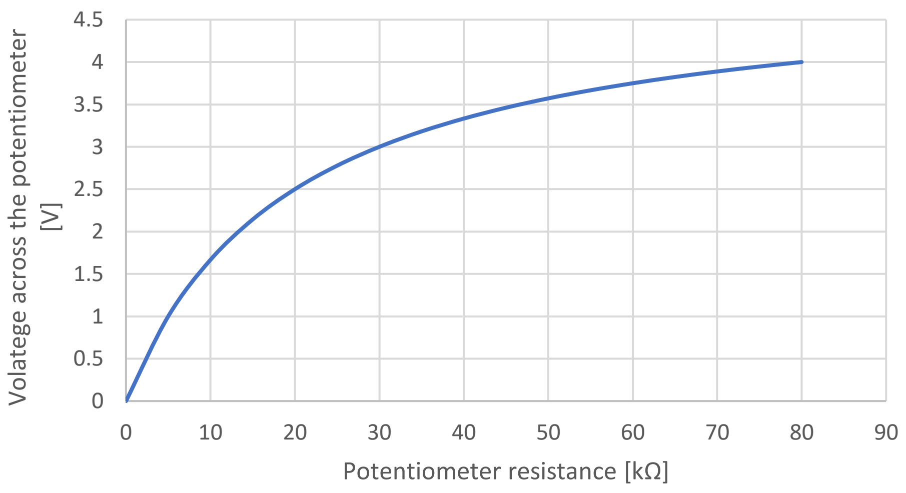
```
       ADCSRA = 0x87;   /* 10000111  Enable ADC, fr/128  */
       ADMUX = 0x40;  /* 01000000 Vref: Avcc, ADC channel: 0 */
}

void PWM_Init()
{
              DDRD |= (1<<PD5);    /* Make OC1A pin as output */
              TCNT1 = 0;               /* Set timer1 count zero */
              ICR1 = 2499;         /* Set TOP count for timer1 in ICR1 register */

              /* Set Fast PWM, TOP in ICR1, Clear OC1A on compare match, clk/64 */
              TCCR1A = (1<<WGM11)|(1<<COM1A1);
              TCCR1B = (1<<WGM12)|(1<<WGM13)|(1<<CS10)|(1<<CS11);
}


int ADC_Read(char channel)
{
       int Ain,AinLow;

       ADMUX=ADMUX|(channel & 0x0f);      /* Set input channel to read */

       ADCSRA |= (1<<ADSC);        /* Start conversion */
       while((ADCSRA&(1<<ADIF))==0);      /* Monitor end of conversion interrupt */

       _delay_us(10);
       AinLow = (int)ADCL;         /* Read lower byte*/
       Ain = (int)ADCH*256;        /* Read higher 2 bits and
                                   Multiply with weight */
       Ain = Ain + AinLow;
       return(Ain);                /* Return digital value*/
}


int main(void)
{
       ADC_Init();
       PWM_Init();
       int adc_value;

       while(1)

       {
              adc_value = ADC_Read(0);    /* Read ADC channel 0 */


              if ((adc_value > threshold_adc_value) && (OCR1A != valve_opened)) /* if
valve should be opened and is not opened then open */
              {
                     OCR1A = valve_opened;       /* Set servo shaft at opened position
*/
                     _delay_ms(1000); /* open for 1 s to avoid unstable operation */
              }
              else if (OCR1A != valve_closed) /* if not already closed then close */
              {
                     OCR1A = valve_closed;       /* Set servo at closed position */
              }


       }

}
```

5. Circuit in reality

The circuit has been assembled on a prototype board.
Video: https://www.youtube.com/watch?v=gGjcYsSJAy4



One change has been made – in this prototype a potentiometer is used to supply voltage to the ADC, as it is easier to test the circuit this way. The servo then reacts when a potentiometer is rotated to a certain position. Its max resistance is 80 kΩ so it has been placed in series with two 10 kΩ resistors to allow simulation od 0 – 4V input:



6. Summary

a) Problems:

- generally understanding the connections of the power supply and the programmer to the microcontroller. The solution was careful study of the microcontroller datasheet and the schematics posted online.

b) Future improvements:

- Add an external power source to the servo (do not use programmer to power everything)
- Verify that the system works with the target sensor
- Add an external crystal to improve stability of the clock
- Design a mount point for the servomechanism on the valve
- Design a custom PCB for the microcontroller and pressure sensor
- Probably change the microcontroller to something smaller and cheaper
- Add an SD card to allow to log the pressure and valve status
- Add more servos to a single controller