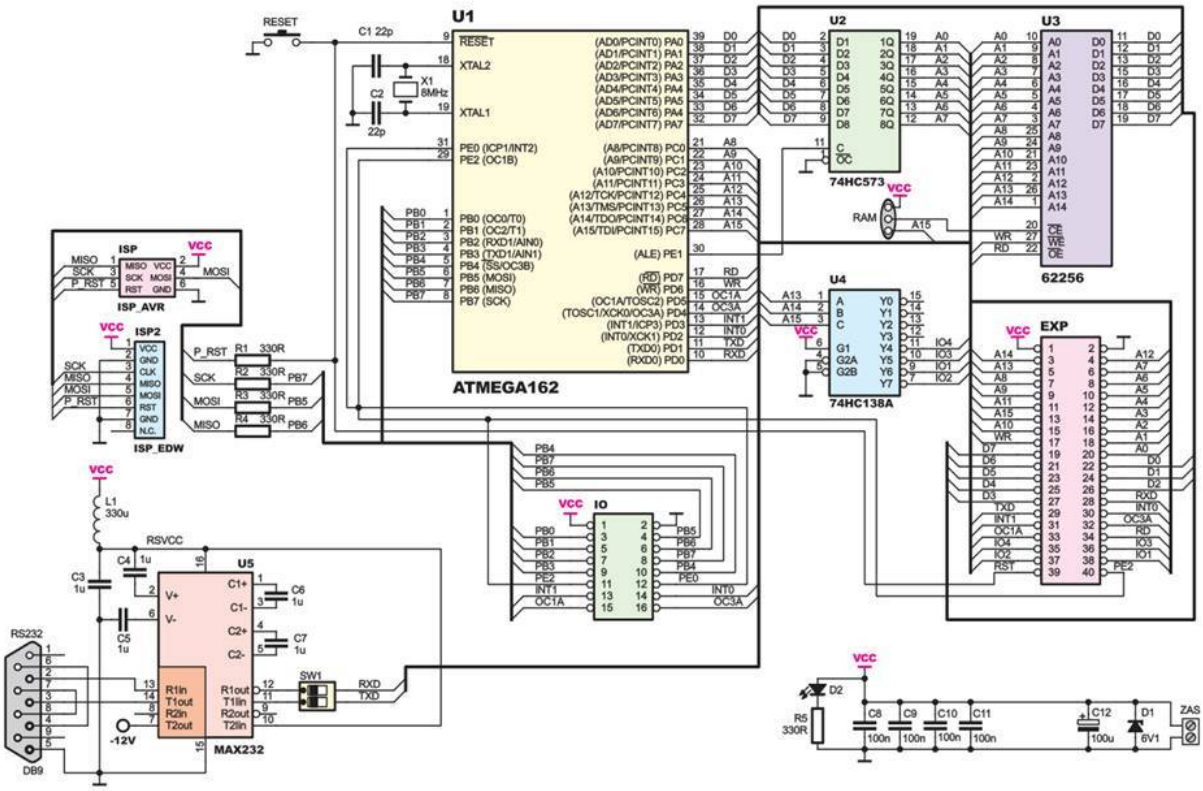
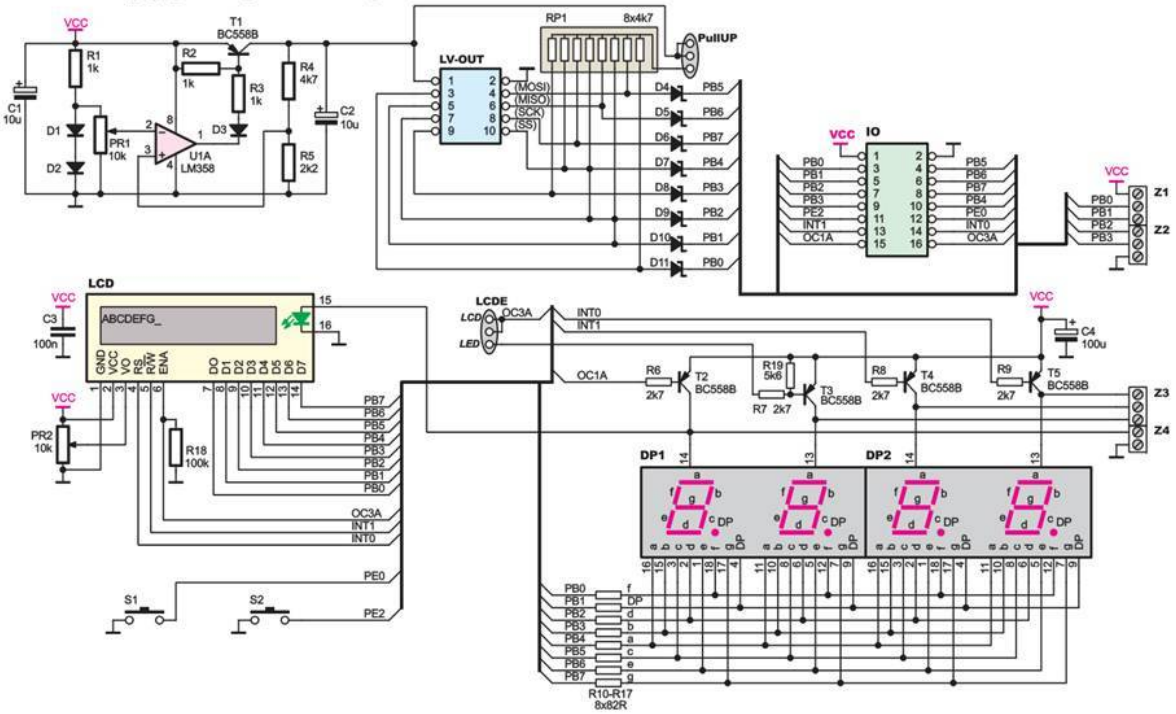
C: Programming an 8 segment AVR LED display

This project uses an AVT 3505 module, the diagram of which is below

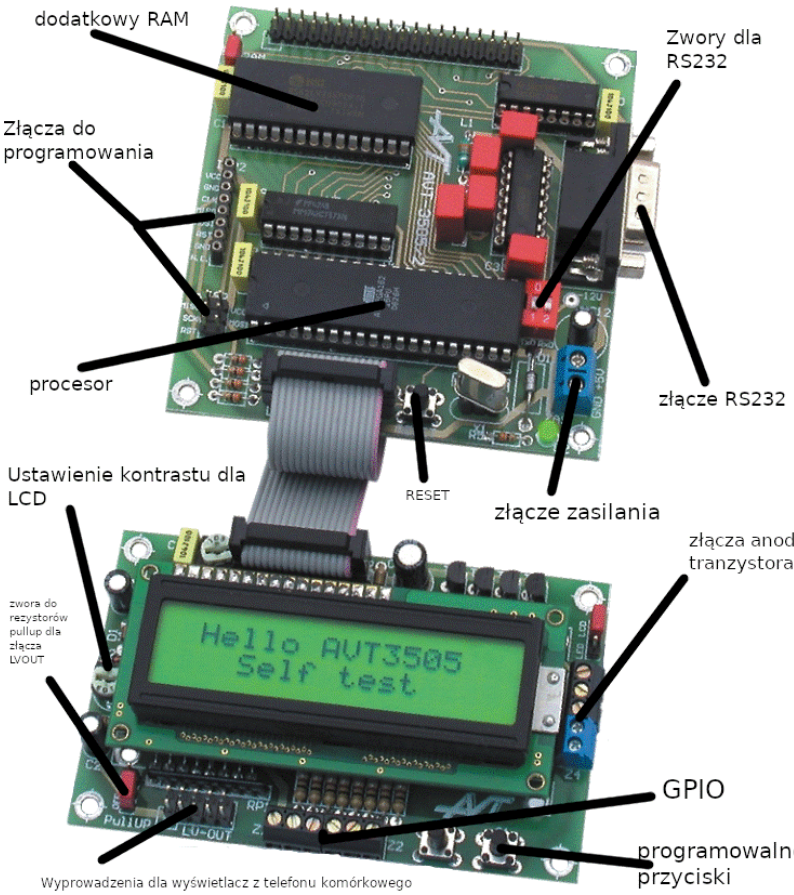


Rys. 1 Schemat ideowy płytki głównej

Rys. 2 Schemat ideowy płytki wykonawczej



The board consists of two parts: the part with the microprocessor (Fig.1) and the part with the actuators (Fig.2)



Moduł mikroprocesora:
- procesor atmega 162
- dwa rodzaje złącz programujących
- pamięć 32kb statyczna
- wbudowane złącze RS232
- zasilanie 5V
- układ zewnętrznej pamięci oraz układ do RS232

Moduł wykonaczy:
- Możliwość podłączenia wyświetlacza LCD, LED
- wyprowadzenia kliku wyjść GPIO z buforem tranzystorowym
- 2 programowalne przełączniki

The program uses multiplexing with transistors T2, T3, T4, T5 (fig.2), this allows to use 4 times less GPIO interfaces for displaying segments on the display. A function has also been written in the program (line 24 in main.c) which converts the
value of a number with a dot to the segment numbers of the display which simplifies the programming, at the end there are some example pins
and a function that automatically prints 4 digital numbers on the display.

main.c

```
#include <avr\io.h>
#include <util\delay.h>
// Definicje wyprowadzen

#define F_CPU 8000000UL

#define LED_A 4
#define LED_B 3
#define LED_C 5
#define LED_D 2
#define LED_E 6
#define LED_F 0
#define LED_G 7
#define LED_DP 1
```

```
#define LEDPORT PORTB
#define LEDDDR DDRB
#define COM1 5
#define COM2 4
#define COM3 3
#define COM4 2
#define COMPORT PORTD
#define COMDDR DDRD

int displayNumber(int num, int com)
{
    switch(com)
    {
        case 1:
        COMDDR = 1<<COM1 | 0<<COM2 | 0<<COM3 | 0<<COM4;
        break;
        case 2:
        COMDDR = 0<<COM1 | 1<<COM2 | 0<<COM3 | 0<<COM4;
        break;
        case 3:
        COMDDR = 0<<COM1 | 0<<COM2 | 1<<COM3 | 0<<COM4;
        break;
        case 4:
        COMDDR = 0<<COM1 | 0<<COM2 | 0<<COM3 | 1<<COM4;
        break;
    }
    switch(num)
    {
        case 0:
        LEDPORT = ~(1<<LED_A | 1<<LED_B | 1<<LED_C | 1<<LED_D |
1<<LED_E | 1<<LED_F );
        break;
        case 1:
        LEDPORT = ~(1<<LED_B | 1<<LED_C);
        break;
        case 2:
        LEDPORT = ~(1<<LED_A | 1<<LED_B | 1<<LED_G | 1<<LED_E |
1<<LED_D);
        break;
        case 3:
        LEDPORT = ~(1<<LED_A | 1<<LED_B | 1<<LED_G | 1<<LED_C |
1<<LED_D);
        break;
        case 4:
        LEDPORT = ~(1<<LED_F | 1<<LED_B | 1<<LED_G | 1<<LED_C );
        break;
        case 5:
        LEDPORT = ~(1<<LED_A | 1<<LED_F | 1<<LED_G | 1<<LED_C |
1<<LED_D);
        break;
        case 6:
```

```
        LEDPORT = ~(1<<LED_A | 1<<LED_F | 1<<LED_G | 1<<LED_C |
1<<LED_D | 1<<LED_E);
        break;
        case 7:
        LEDPORT = ~(1<<LED_B | 1<<LED_C | 1<<LED_A);
        break;
        case 8:
        LEDPORT = ~(1<<LED_A | 1<<LED_B | 1<<LED_C | 1<<LED_D |
1<<LED_E | 1<<LED_F | 1<<LED_G);
        break;
        case 9:
        LEDPORT = ~(1<<LED_A | 1<<LED_B | 1<<LED_C | 1<<LED_D |
1<<LED_F | 1<<LED_G);
        break;
    }

}

int displayInteger(int integer)
{
    int n1,n10,n100,n1000;
    n1000 = integer / 1000 % 10;
    n100 = integer / 100 % 10;
    n10 = integer / 10 % 10;
    n1 = integer % 10;
    displayNumber(n1000,1);
    _delay_ms(10);
    displayNumber(n100,2);
    _delay_ms(10);
    displayNumber(n10,3);
    _delay_ms(10);
    displayNumber(n1,4);
    _delay_ms(10);
}

int main(void)
{
    LEDDDR = 0xff;

    while(1)
    {
//        odliczanie na kazdym wysiwtlaczu od 0 do 9
//        scrollowanie liczb
        for(int i = 0; i <= 9 ; i ++)
        {
            for(int j = 1; j <= 4 ; j ++)
            {
                displayNumber(i,j);
                _delay_ms(2000);
            }
        }
```

```
//         liczenie od 0 do 9999
         for(int i = 0; i <= 9999 ; i++){
             displayInteger(i);
         }
//         displayInteger(1234);
//    }

    return 0;
}
}
```