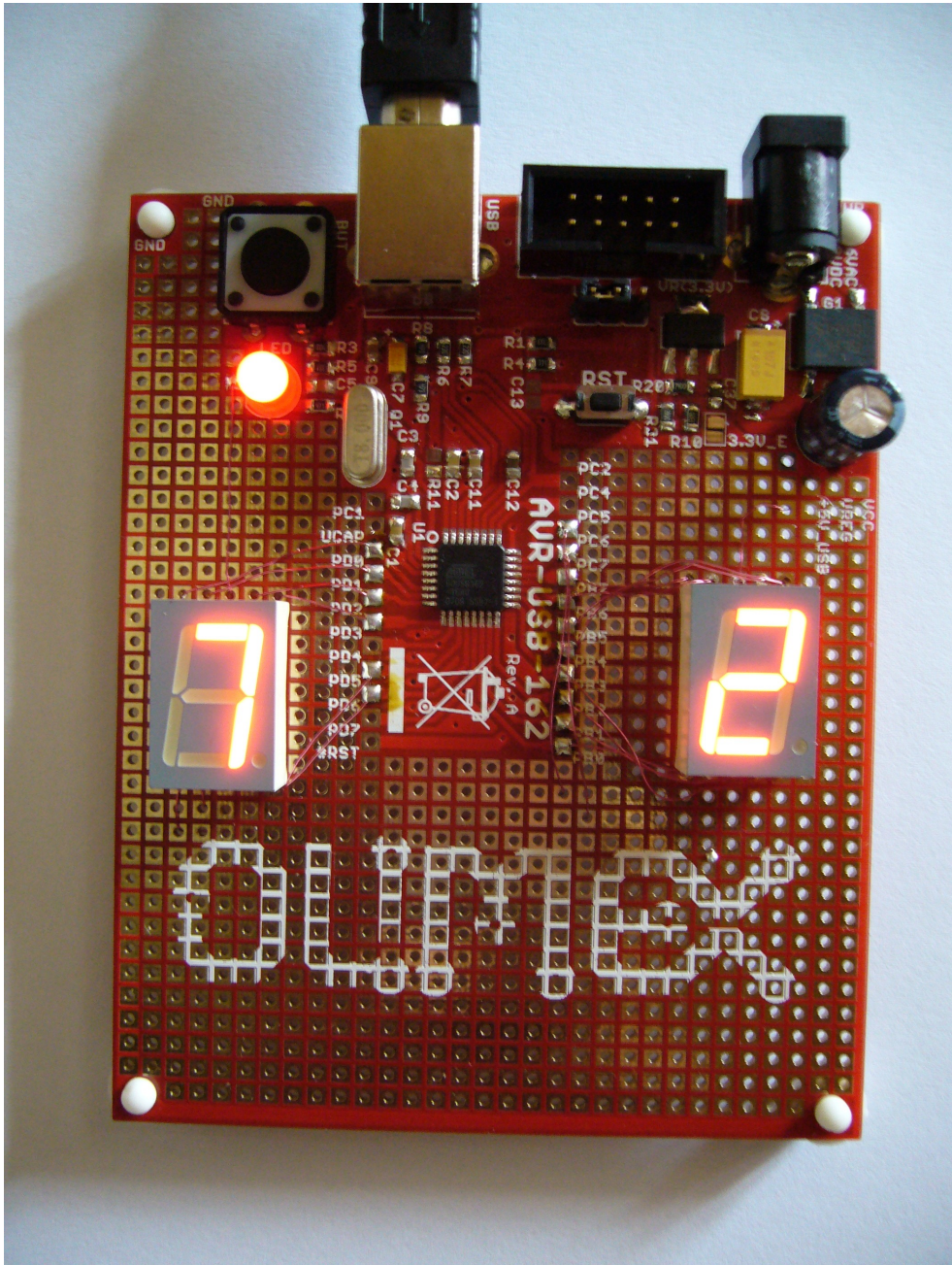


Projektarbeit Mikrocontroller

„Würfelspiel“



Christian Riedel – Juli 2010

Hardware

- OLIMEX AVR-USB-162 Entwicklerboard
- ATmega AT90USB162 (auf Entwicklerboard)
- zwei 7-Segment Anzeigen (auf Entwicklerboard)
- eine LED (auf Entwicklerboard)
- Taster (auf Entwicklerboard)
- USB-Kabel

Software

- AVR Studio 4.18
- Flip 3.4.1

Spielablauf

Das Spiel startet mit dem „Reset“-Taster (oben rechts). Mit diesem kann auch während oder am Ende eines Spiels ein neues Spiel gestartet werden.

Zu Beginn des Spiels werden die Zahlen 1 – 6 im Abstand von 50ms und 10ms auf den beiden 7-Segment-Anzeigen angezeigt, wobei je eine Anzeige für einen Würfel steht.

Drückt der Spieler nun den Taster (oben links) zeigen die beiden Anzeigen die Würfelwerte die der Spieler erreicht hat für 5s an, diese Werte werden addiert.

Nach 5s beginnt der Ablauf von neuem und der Spieler kann erneut würfeln, die Würfe werden alle addiert.

Wenn der Spieler 6 mal gewürfelt hat wird ihm das Ergebnis auf beiden Anzeigen dargestellt und die rote LED (oben links) leuchtet.

Portbelegung

rechte 7-S-A:

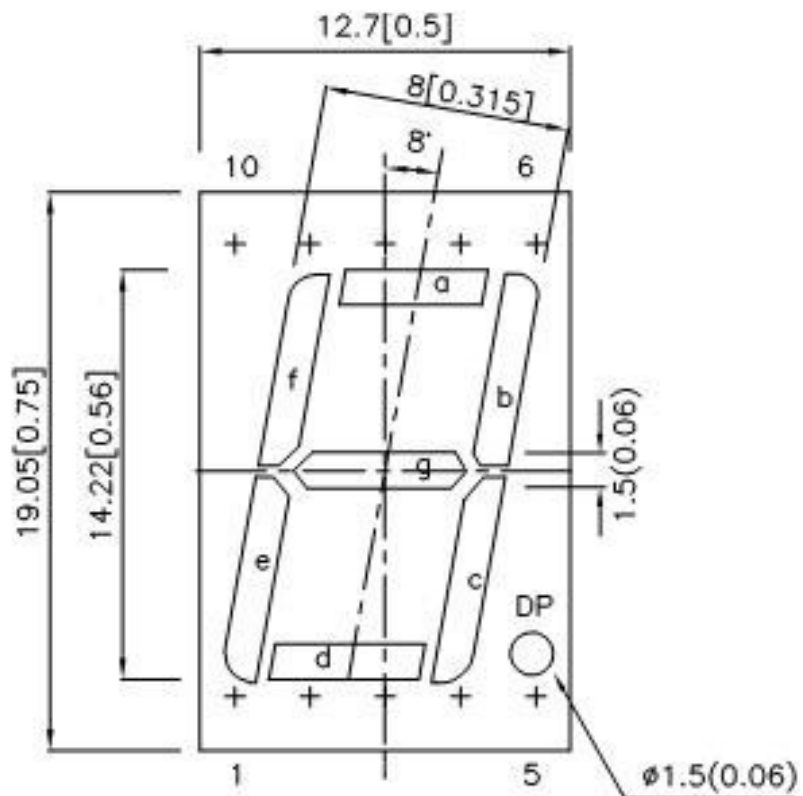
PIN a PORT B5
PIN b PORT B4
PIN c PORT B2
PIN d PORT B1
PIN e PORT B0
PIN f PORT B6
PIN g PORT B7

linke 7-S-A:

PIN a PORT D2
PIN b PORT D3
PIN c PORT D6
PIN d PORT C7
PIN e PORT C6
PIN f PORT D1
PIN g PORT D0

LED:
PORT D4

Taster:
PORT D7



Quellcode

```
#include <avr/io.h>
#define F_CPU 1000000
#include <util/delay.h>
#include <stdlib.h>

void zahl_links(int);
void zahl_rechts(int);

int main(void)
{
    int i=1, j=0, k, ges=0, z_zahl1=0, z_zahl2=0;

    DDRD &= ~(1 << DDD7);

    DDRB = 0xff;

    DDRC |= ((1 << DDC6) | (1 << DDC7));

    DDRD |= ((1 << DDD0) | (1 << DDD1) | (1 << DDD2) | (1 << DDD3));
    DDRD |= ((1 << DDD4) | (1 << DDD5) | (1 << DDD6));

    PORTD |= (1 << PORTD7);

    PORTB &= ~((1 << PORTB0) | (1 << PORTB1) | (1 << PORTB2));
    PORTB &= ~((1 << PORTB3) | (1 << PORTB5) | (1 << PORTB6) | (1 << PORTB7));

    PORTC &= ~((1 << PORTC6) | (1 << PORTC7));
    PORTD &= ~((1 << PORTD0) | (1 << PORTD1) | (1 << PORTD2));
    PORTD &= ~((1 << PORTD3) | (1 << PORTD4) | (1 << PORTD5) | (1 << PORTD6));

    while(1)
    {
        if(!(PIND & (1 << PORTD7)))
        {
            _delay_ms(50);

            srand(ges*512+i);
            z_zahl1=(rand()%6+1);

            srand(ges*1024+i);
            z_zahl2=(rand()%6+1);

            ges+=(z_zahl1+z_zahl2);

            zahl_links(z_zahl1);
            zahl_rechts(z_zahl2);

            _delay_ms(5000);

            j++;
        }
    }
}
```

```

        while(j==6)
        {
            PORTD |= (1 << PORTD4);

            zahl_links(ges/10);
            zahl_rechts(ges%10);
        }
    }
else
{
    _delay_ms(50);
    zahl_links(i);
    _delay_ms(10);
    zahl_rechts(i);

    if(i==6)
    {
        i=1;
    }
    else
    {
        i++;
    }
}
}

return 0;
}

```

```

void zahl_links(int dec)
{
    PORTC &= ~((1 << PORTC6) | (1 << PORTC7));
    PORTD &= ~((1 << PORTD0) | (1 << PORTD1) | (1 << PORTD2));
    PORTD &= ~((1 << PORTD3) | (1 << PORTD5) | (1 << PORTD6));

    switch(dec)
    {
        case 0: PORTC |= ((1 << PORTC6) | (1 << PORTC7));
                PORTD |= ((1 << PORTD1) | (1 << PORTD2));
                PORTD |= ((1 << PORTD3) | (1 << PORTD6));
                break;
        case 1: PORTD |= ((1 << PORTD6) | (1 << PORTD3));
                break;
        case 2: PORTD |= ((1 << PORTD2) | (1 << PORTD3) | (1 << PORTD0));
                PORTC |= ((1 << PORTC6) | (1 << PORTC7));
                break;
        case 3: PORTD |= ((1 << PORTD2) | (1 << PORTD3));
                PORTD |= (1 << PORTD0) | (1 << PORTD6));
                PORTC |= ((1 << PORTC7));
                break;
        case 4: PORTD |= ((1 << PORTD1) | (1 << PORTD0));
                PORTD |= ((1 << PORTD3) | (1 << PORTD6));
                break;
    }
}

```

```

case 5: PORTD |= ((1 << PORTD2) | (1 << PORTD1));
        PORTD |= ((1 << PORTD0) | (1 << PORTD6));
        PORTC |= ((1 << PORTC7));
        break;
case 6: PORTD |= ((1 << PORTD2) | (1 << PORTD1));
        PORTD |= ((1 << PORTD0) | (1 << PORTD6));
        PORTC |= ((1 << PORTC6) | (1 << PORTC7));
        break;
case 7: PORTD |= ((1 << PORTD6) | (1 << PORTD3) | (1 << PORTD2));
        break;
case 8: PORTC |= ((1 << PORTC6) | (1 << PORTC7));
        PORTD |= ((1 << PORTD0) | (1 << PORTD1) | (1 << PORTD2));
        PORTD |= ((1 << PORTD3) | (1 << PORTD6));
        break;
case 9: PORTD |= ((1 << PORTD2) | (1 << PORTD1) | (1 << PORTD0));
        PORTD |= ((1 << PORTD6) | (1 << PORTD3));
        PORTC |= ((1 << PORTC7));
        break;
    }
}

```

```

void zahl_rechts(int dec)
{
    PORTB &= ~(1 << PORTB0) | (1 << PORTB1) | (1 << PORTB2) | (1 << PORTB3));
    PORTB &= ~(1 << PORTB5) | (1 << PORTB6) | (1 << PORTB7));

    switch(dec)
    {
        case 0: PORTB = 0x77;
                break;
        case 1: PORTB = 0x14;
                break;
        case 2: PORTB = 0xb3;
                break;
        case 3: PORTB = 0xb6;
                break;
        case 4: PORTB = 0xd4;
                break;
        case 5: PORTB = 0xe6;
                break;
        case 6: PORTB = 0xe7;
                break;
        case 7: PORTB = 0x34;
                break;
        case 8: PORTB = 0xf7;
                break;
        case 9: PORTB = 0xf6;
                break;
    }
}

```