

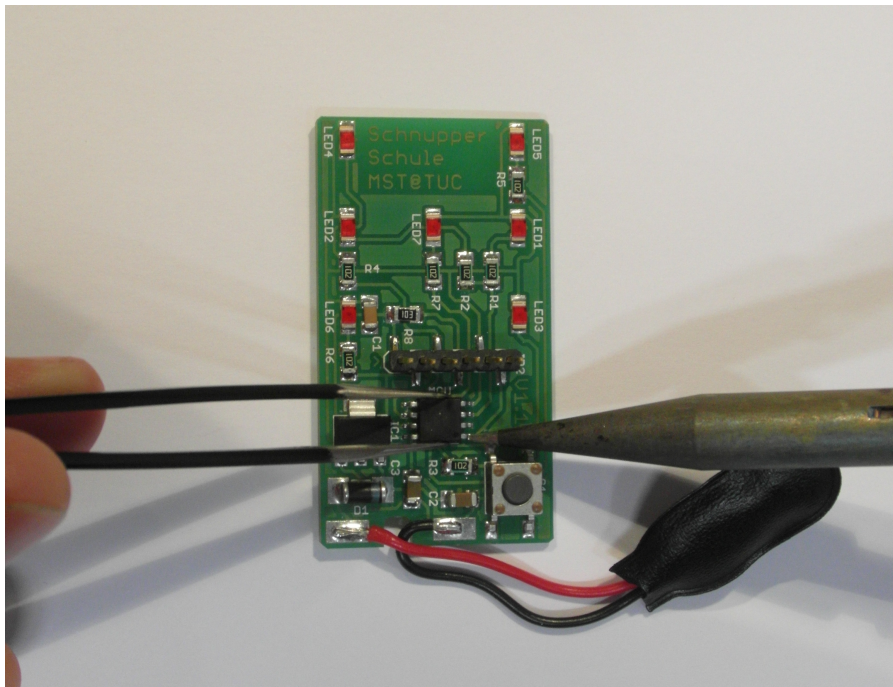
Schnupperschule

LED-Würfel

TU-Chemnitz, Professur für Mess- und Sensortechnik

Übersicht

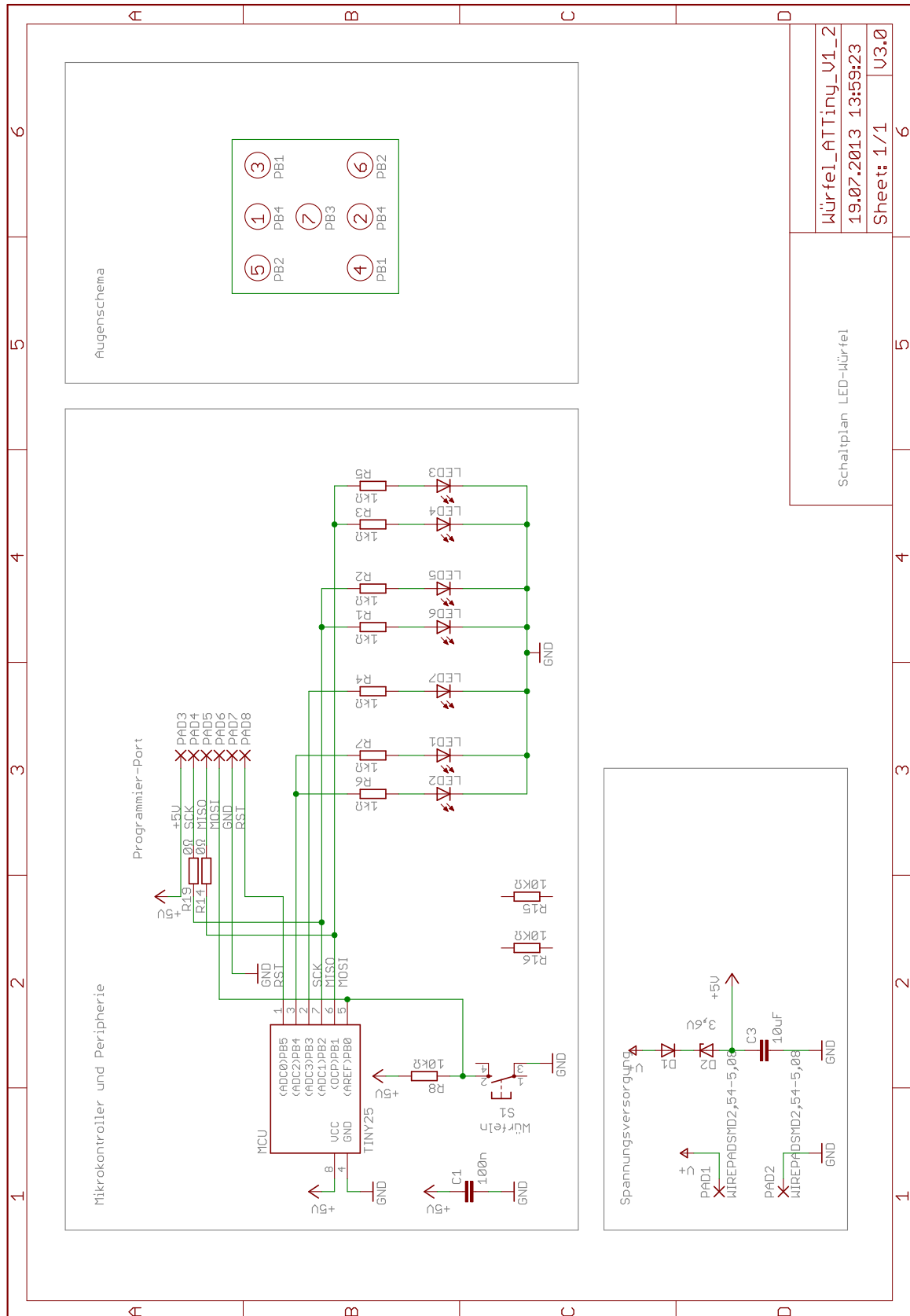
- Schaltplan (Abbildung 1 auf der nächsten Seite)
- Bestückungsplan (Abbildung 2 auf Seite 3)
- Belichtungsmaske (Abbildung 3 auf Seite 4)
- Programmquelltext (Code-Listing 1 auf Seite 5)



Nützliche Webseiten

- Übersicht zum Angebot und Anmeldung zur Schnupperschule
<http://www.tu-chemnitz.de/etit/schnupperschule/>
- Sehr umfangreiches Tutorial, gut geeignet für Einsteiger
<http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial>
- Interesse geweckt?
<http://reichelt.de/> → Suche nach Arduino
<http://arduino.cc/> → Vollständige Dokumentation

Abbildung 1: Schaltplan des LED-Würfels



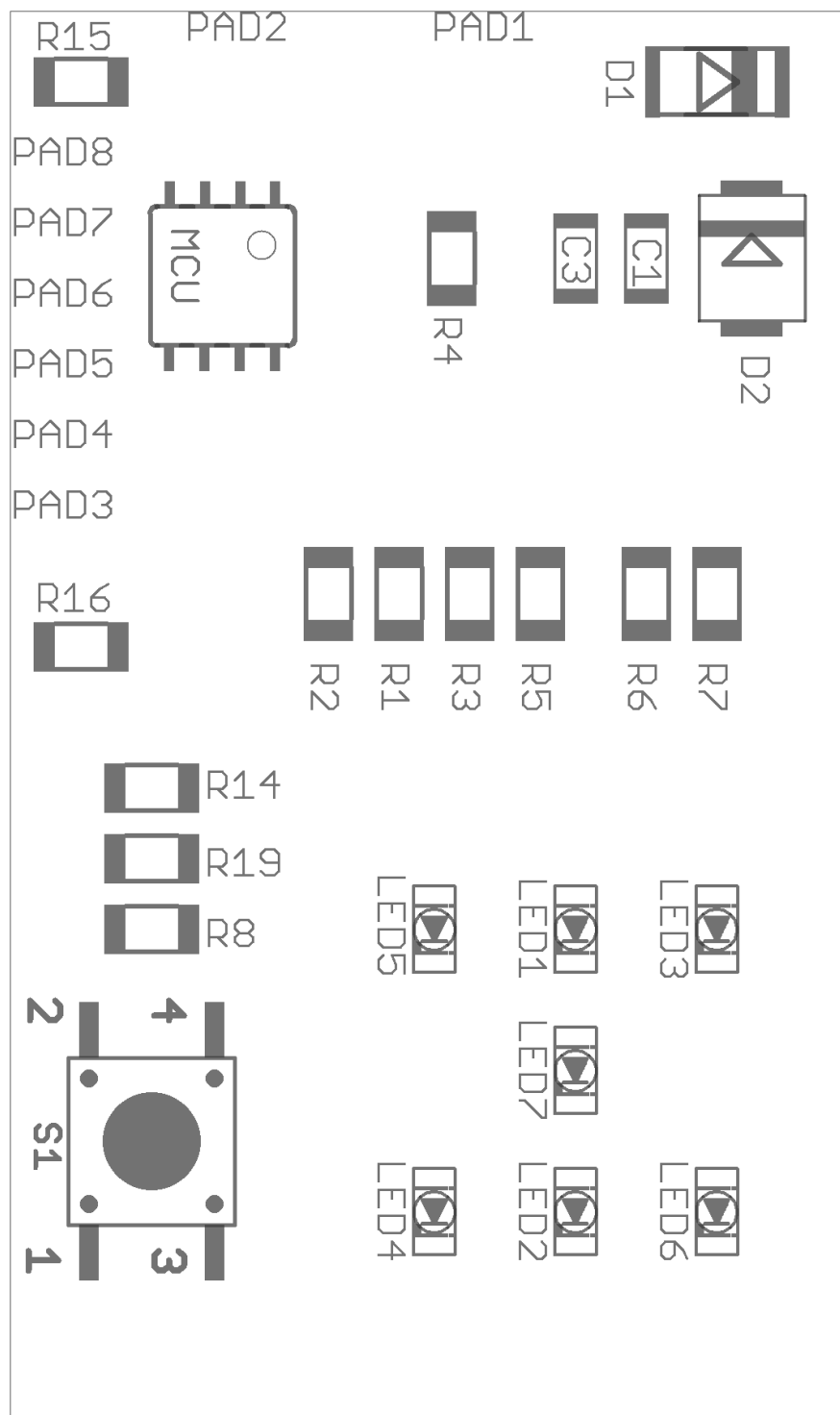


Abbildung 2: Bestückungsplan der Oberseite

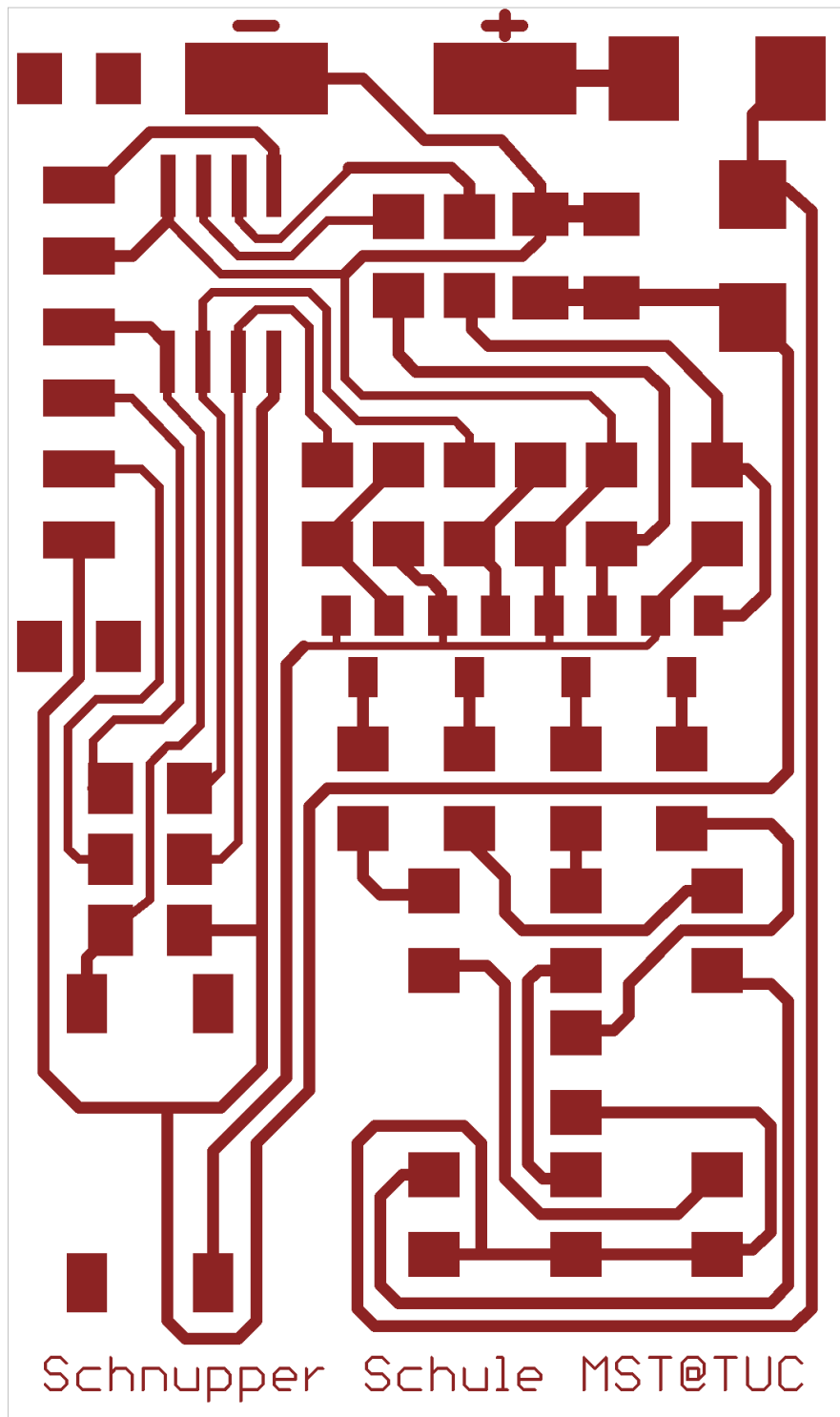


Abbildung 3: Belichtungsmaske und Fräskontur der Leiterplatte

Listing 1: Quelltext des Würfels

```

1  /*****
2  *                               led_wuerfel.c
3  *                               =====
4  *                               Uwe Berger; 2009, Thomas Günther 2012
5  *
6  * ...ein elektronischer 1-aus-6 Wuerfel mit einem >= ATtiny25.
7  *
8  * Nach ca. 18s (ausprobiert, nicht explizit programmiert) ohne
9  * Tastenbetaetigung geht der ATtiny in PowerDown-Mode (Stromver-
10 * brauch dann ca. 0,00026mA). Das Aufwachen erfolgt wiederum via
11 * Taste.
12 *
13 * Die Anordnung der 7 LEDs und deren Verschaltung an den Pins des
14 * ATtiny ist dem Quelltext zu entnehmen. Entsprechend dimensionier-
15 * te Vorwiderstaende sollten nicht vergessen werden.
16 *
17 * Der geschlossene Taster legt GND an den entsprechenden Pin, es
18 * sollte ein PullUp-Widerstand von 10kOhm vorgesehen werden.
19 *
20 */
21
22 #include <avr/interrupt.h>
23 #include <avr/io.h>
24
25 #ifndef F_CPU
26 #define F_CPU 1000000UL
27 #endif
28
29 #define BOOL(x)          (!(! (x)))
30
31 #define BIT_MASK(x)      ((0x01)<<(x))
32 #define SET_BIT(x,y)     ((x) |=  BIT_MASK(y))
33 #define CLEAR_BIT(x,y)   ((x) &= ~BIT_MASK(y))
34 #define TOGGLE_BIT(x,y)  ((x) ^=  BIT_MASK(y))
35
36 #define TEST_BIT(x,y)     (BOOL((x) & BIT_MASK((y))))
37
38 #include <stdlib.h>
39 #include <avr/sleep.h>
40 #include <util/delay.h>
41
42 /* Anordnung der LEDs
43 * 5   1   3
44 *     7
45 * 4   2   6
46 */
47
48 // LED   1  2  3  4  5  6  7
49 // PBx   4  4  1  1  2  2  3
50 // Zahl
51 // 1                x
52 // 2      x  x
53 // 3                x  x                x
54 // 4                x  x  x  x
55 // 5                x  x  x  x  x
56 // 6      x  x  x  x  x  x
57
58 #define LED_DDR      DDRB
59 #define LED_PORT      PORTB
60 #define MASK_LED12    BIT_MASK(PB4)
61 #define MASK_LED34    BIT_MASK(PB1)
62 #define MASK_LED56    BIT_MASK(PB2)

```

```
63 #define MASK_LED7    BIT_MASK(PB3)
   #define MASK_ALL_LEDS    (MASK_LED12|MASK_LED34|MASK_LED56|MASK_LED7);
65 // entsprechendes LED-Muster
uint8_t led_tab[6]={
67     MASK_LED7,
        MASK_LED12,
69     MASK_LED7|MASK_LED34,
        MASK_LED34|MASK_LED56,
71     MASK_LED34|MASK_LED56|MASK_LED7,
        MASK_LED12|MASK_LED34|MASK_LED56
73 };

75 // entsprechendes LED-Muster
uint8_t anim_tab[6]={
77     MASK_LED12,
        MASK_LED34,
79     MASK_LED56
    };

81 // Taster
83 #define TASTER_DDR    DDRB
   #define TASTER_PORT  PORTB
85 #define TASTER_PIN    PB0
   #define TASTER_INT    PCINT0
87
   #define SCHLAF_WERT 50000    // ca. 3s
89
volatile uint8_t taste = 0;
91 uint8_t zahl = 0;
uint32_t schlafen = 0;
93
//*****
95 ISR(PCINT0_vect)
{
97     // Taste losgelassen (Low --> High)...
    if (TEST_BIT(PINB, TASTER_PIN)) {
99         taste = 1;
    }
101 }

103 //*****
// damit Gleichverteilung der Zufallszahlen ueber den ganzen Bereich
105 // gewaehrleistet ist, muss ein wenig mehr gemacht werden, als rand()
// aufzurufen...; Quelle: http://www.mikrocontroller.net/topic/80481
107 #define MAX_WERT 6
uint8_t my_rand()
109 {
    uint16_t x;
111     while((x = rand()) >= RAND_MAX - (RAND_MAX % MAX_WERT));
    return x % MAX_WERT;
113 }

115 //*****
// seed-Wert fuer srand() generieren; Quelle:
117 // http://www.roboternetz.de/wissen/index.php/Zufallszahlen\_mit\_avr-gcc
unsigned int get_seed()
119 {
    uint16_t seed = 0;
121     uint16_t *p = (uint16_t*) (RAMEND+1);
    extern uint16_t __heap_start;
123     while (p >= &__heap_start + 1)
        seed ^= * (--p);
125     return seed;
}
```

```
127 }
129 //*****
130 //Hauptprogramm
131 int main (void)
132 {
133     // Ausgaenge setzen
134     LED_DDR |= (MASK_LED12|MASK_LED34|MASK_LED56|MASK_LED7);
135     // PCINT fuer Taster initialisieren
136     CLEAR_BIT(TASTER_PORT, TASTER_PIN); // Taster-Pin als Eingang
137     SET_BIT(PCMSK, TASTER_INT);          // Taster-Pin maskieren
138     SET_BIT(GIMSK, PCIE);                // Port Change Interrupt frei
139     sei();                               // Interrupts einschalten
140
141     srand(get_seed());                   // Zufallsgenerator initialisieren
142     while(1) {
143
144         schlafen++;
145
146         // Taste gedrueckt, LEDs animieren
147         if (!(TEST_BIT(PINB, TASTER_PIN)) &&
148             !(schlafen % 50)) {
149             zahl++;
150             schlafen = 0;
151             if (zahl > 5) zahl = 0;
152             LED_PORT = led_tab[zahl];    // LEDs sind high-aktiv
153         }
154
155         // Taste losgelassen, also wuerfeln...
156         if (taste) {
157             taste = 0;
158             zahl = my_rand();
159             schlafen = 0;
160             LED_PORT = led_tab[zahl];    // LEDs sind high-aktiv
161         }
162
163         // Schlafwert erreicht, dann Power Down...
164         if (schlafen > SCHLAF_WERT) {
165             LED_PORT &= ~MASK_ALL_LEDS; // LEDs aus (high-aktiv)
166             schlafen = 0;                // Schlafzaehler zuruecksetzen!
167             set_sleep_mode(SLEEP_MODE_PWR_DOWN);
168             sleep_mode();
169         }
170     }
171 }
```