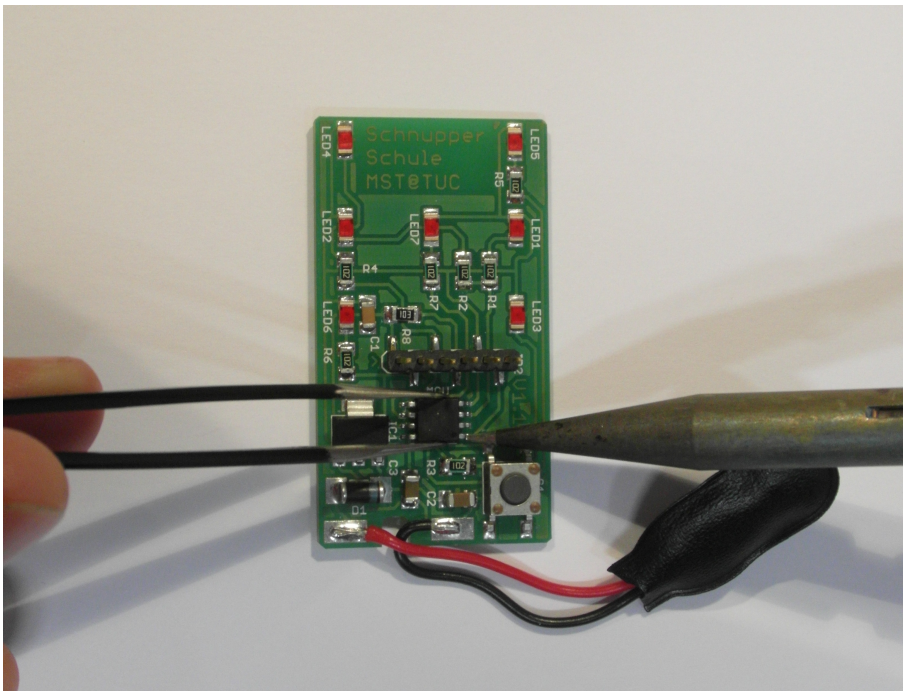
# Schnupperschule

LED–Würfel
TU-Chemnitz, Professur für Mess- und Sensortechnik

## Übersicht

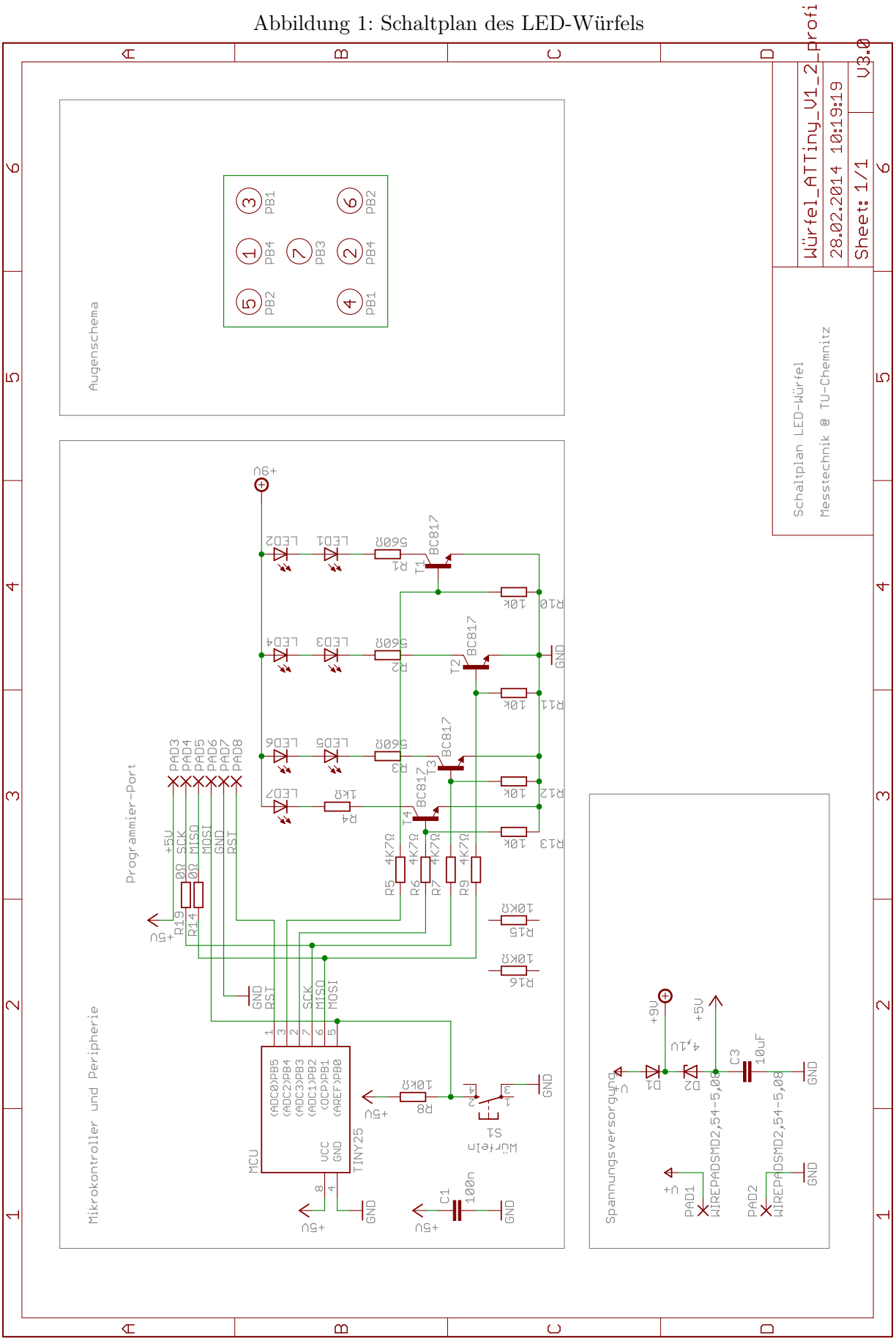





## Nützliche Webseiten

- 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



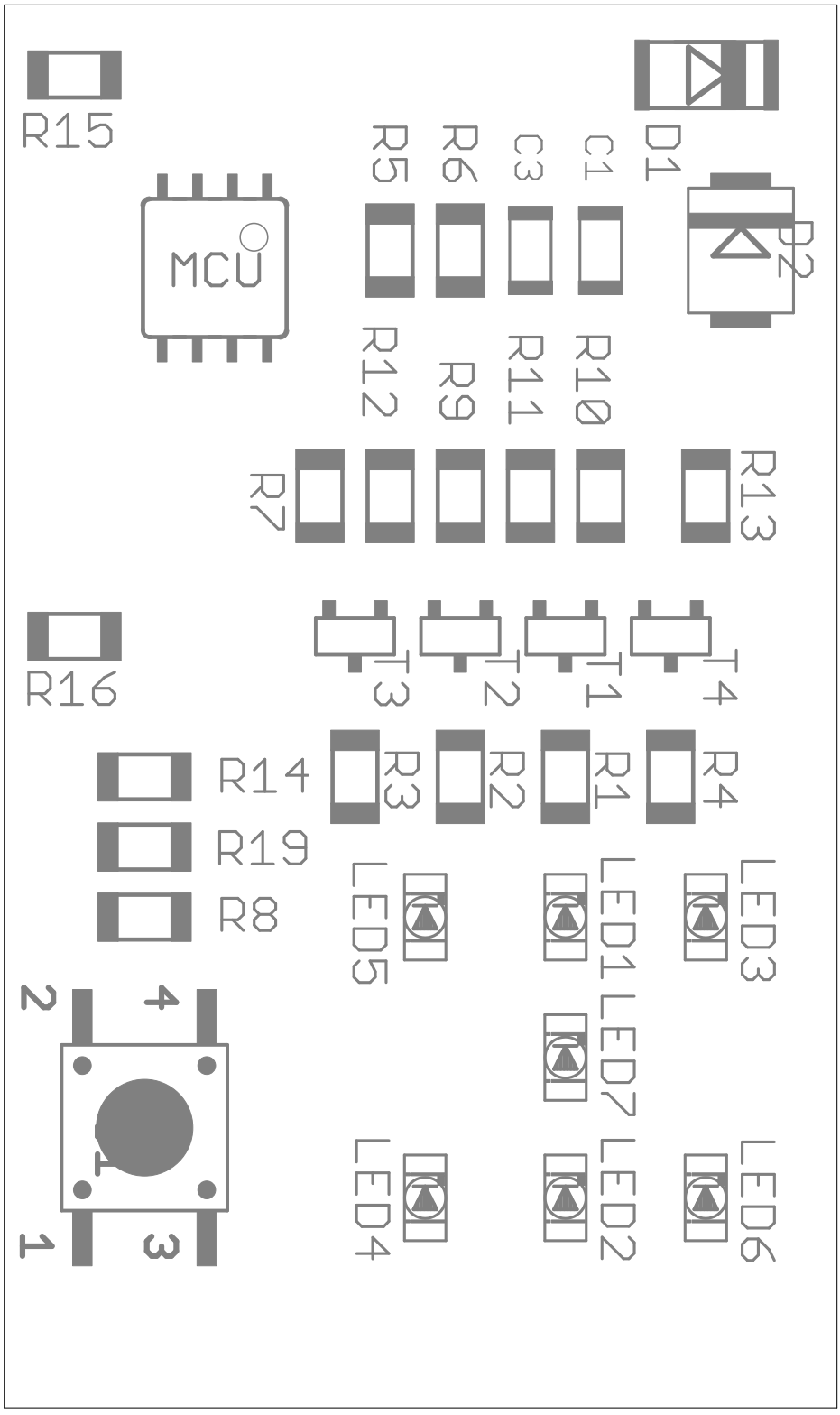



Abbildung 2: Bestückungsplan der Oberseite

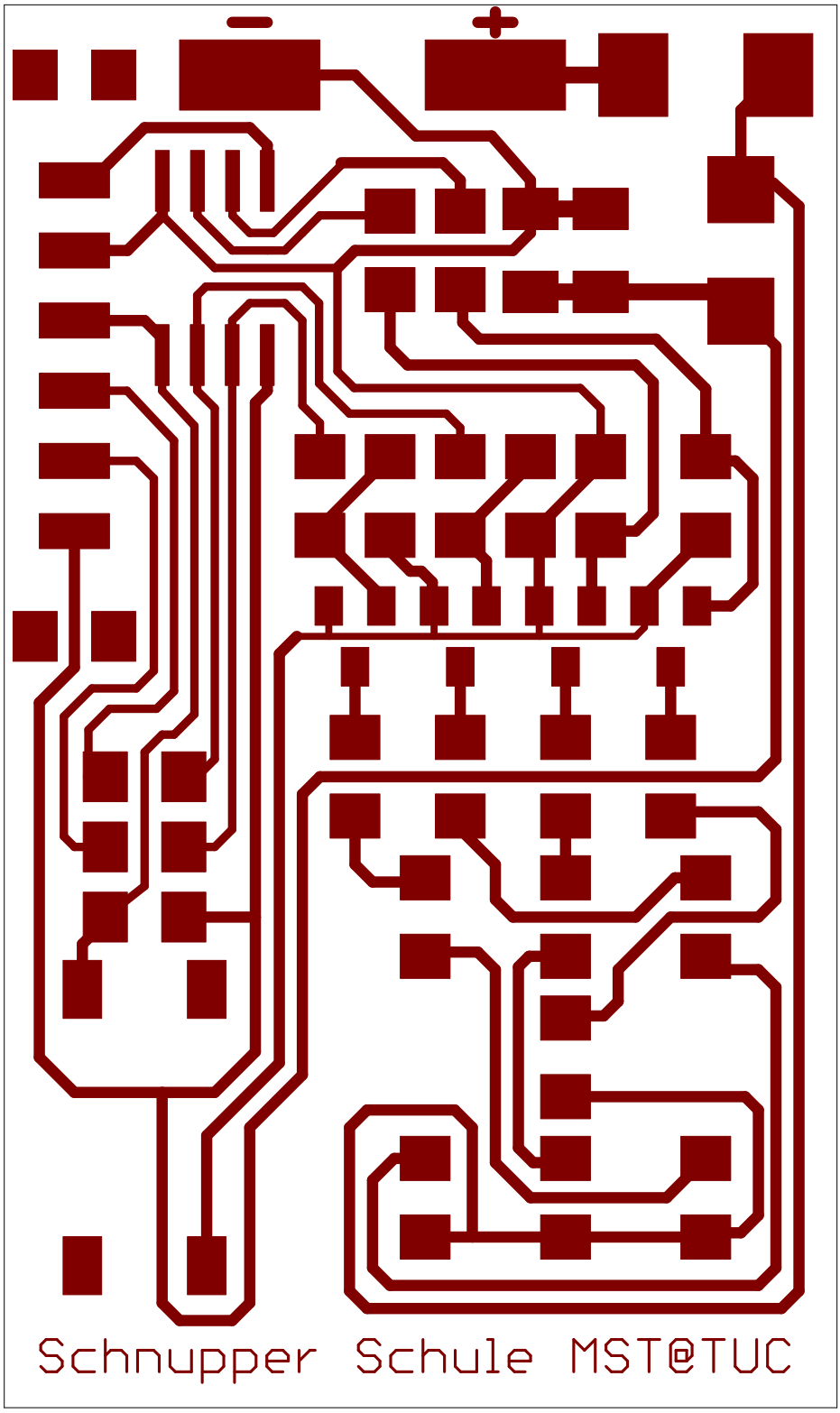



Abbildung 3: Belichtungsmaske und Fräskontur der Leiterplatte

Listing 1: Quelltext des Würfels

```
/*************************************************************
 *                        led_wuerfel.c
 *                        =============
 *            Uwe Berger; 2009,Thomas Günther 2012
 *
 * ...ein elektronischer 1-aus-6 Wuerfel mit einem >= ATtiny25.
 *
 * Nach ca. 18s (ausprobiert, nicht expliziet programmiert) ohne
 * Tastenbetaetigung geht der ATtiny in PowerDown-Mode (Stromver-
 * brauch dann ca. 0,00026mA). Das Aufwachen erfolgt wiederum via
 * Taste.
 *
 * Die Anordnung der 7 LEDs und deren Verschaltung an den Pins des
 * ATtiny ist dem Quelltext zu entnehmen. Entsprechend dimensionier-
 * te Vorwiderstaende sollten nicht vergessen werden.
 *
 * Der geschlossene Taster legt GND an den entsprechenden Pin, es
 * sollte ein PullUp-Widerstand von 10kOhm vorgesehen werden.
 *
*/

#include <avr/interrupt.h>
#include <avr/io.h>

#ifndef F_CPU
#define F_CPU 1000000UL
#endif

#define BOOL(x)          (!(!(x)))

#define BIT_MASK(x)      ((0x01)<<(x))
#define SET_BIT(x,y)     ((x) |=  BIT_MASK(y))
#define CLEAR_BIT(x,y)   ((x) &= ~BIT_MASK(y))
#define TOGGLE_BIT(x,y)  ((x) ^=  BIT_MASK(y))

#define TEST_BIT(x,y)    (BOOL((x) & BIT_MASK((y))))

#include <stdlib.h>
#include <avr/sleep.h>
#include <util/delay.h>

/* Anordnung der LEDs
*  5   1   3
*      7
*  4   2   6
*/

// LED   1  2 3 4 5 6 7
// PBx   4  4   1   1   2   2   3
// Zahl
// 1                            x
// 2     x  x
// 3            x   x           x
// 4            x   x   x   x
// 5            x   x   x   x   x
// 6     x  x   x   x   x   x

#define LED_DDR        DDRB
#define LED_PORT       PORTB
#define MASK_LED12     BIT_MASK(PB4)
#define MASK_LED34     BIT_MASK(PB1)
#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;
```

```
}


//*****************************************************************
//Hauptprogramm
int main (void)
{
  // Ausgaenge setzen
  LED_DDR |= (MASK_LED12|MASK_LED34|MASK_LED56|MASK_LED7);
  // PCINT fuer Taster initialisieren
  CLEAR_BIT(TASTER_PORT, TASTER_PIN); // Taster-Pin als Eingang
  SET_BIT(PCMSK, TASTER_INT);       // Taster-Pin maskieren
  SET_BIT(GIMSK, PCIE);          // Port Change Interrupt frei
  sei();                 // Interrupts einschalten

  srand(get_seed());           // Zufallsgenerator initialisieren
  while(1) {

    schlafen++;

    // Taste gedrueckt, LEDs animieren
    if (!(TEST_BIT(PINB, TASTER_PIN)) &&
      !(schlafen % 50)) {
        zahl++;
        schlafen = 0;
        if (zahl > 5) zahl = 0;
        LED_PORT = led_tab[zahl];     // LEDs sind high-aktiv
    }

    // Taste losgelassen, also wuerfeln...
    if (taste) {
      taste = 0;
      zahl = my_rand();
      schlafen = 0;
      LED_PORT = led_tab[zahl];        // LEDs sind high-aktiv
    }

    // Schlafwert erreicht, dann Power Down...
    if (schlafen > SCHLAF_WERT) {
      LED_PORT &= ~MASK_ALL_LEDS;       // LEDs aus (high-aktiv)
      schlafen = 0;              // Schlafzaehler zuruecksetzen!
      set_sleep_mode(SLEEP_MODE_PWR_DOWN);
      sleep_mode();
    }
  }
}
```