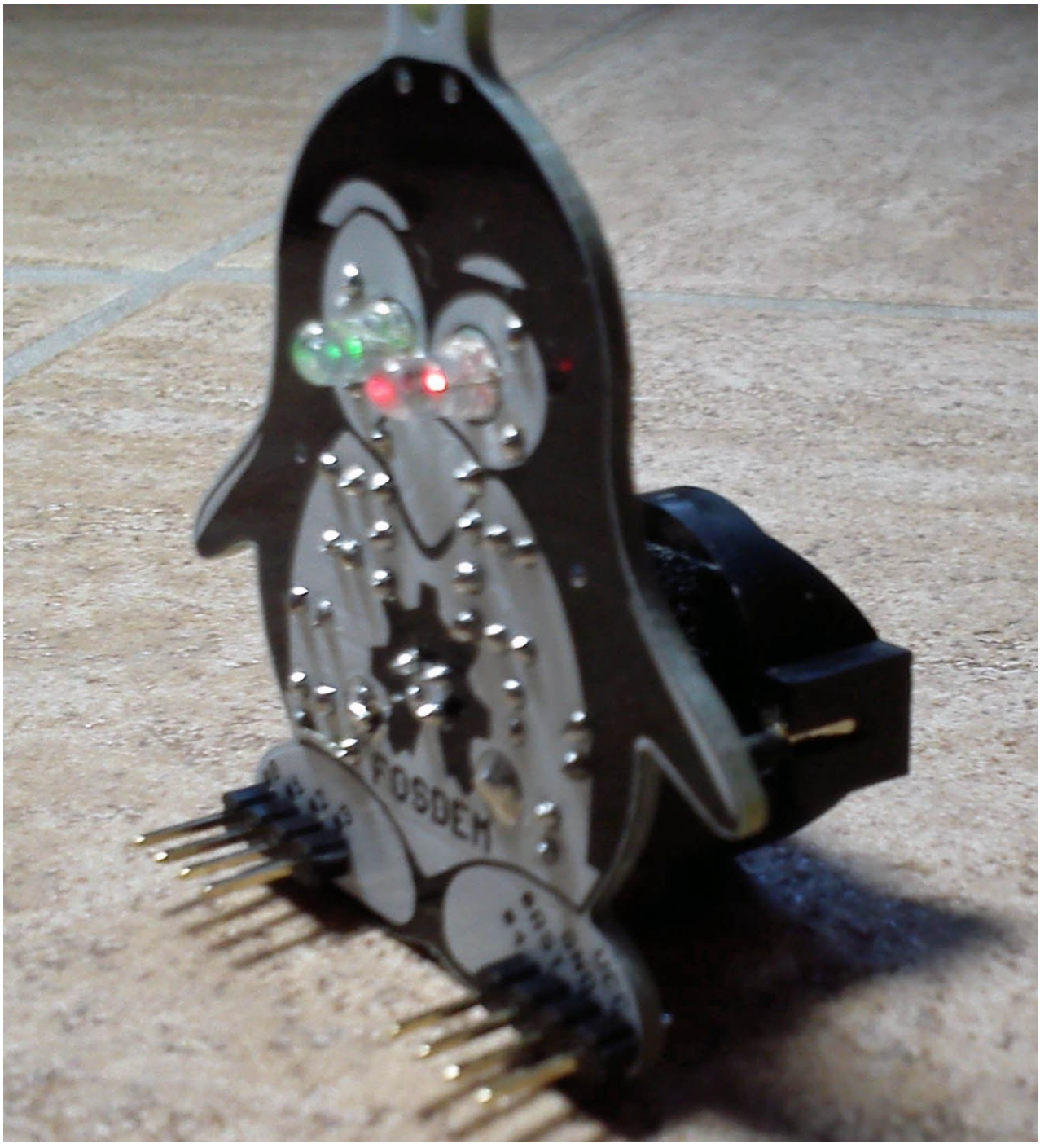
FOSDEM-85 as a temperature sensor and display

What is to do with a FOSDEM-85-

penguin? A work for a rainy day

Stefan Goetze / 2014-02-12 / rebuilt in 2017



I bought a FOSDEM-85-penguin from the Olimex web shop. After some days it arrived in Germany. I soldered it without any problem. It looks nice so I decided to give it a place at my desk as a thermometer. I use the ATTiny85 internal chip sensor for measuring the temperature.

Material:

– FOSDEM-85 from Olimex Ltd.

– CR2032 + holder (backpack) or 2xAAA with holder (as platform)

Modification:

I wanted to have two leds for temperature display so I unsoldered the VCC-side of resistor R1 out of the PCB and connected it to PB2 (Pin 7 of ATTiny85). So I can control two leds by program. Best is, if you solder the FOSDEM-85 – let this resistor pin out of the PCB-hole.
Look at the schematic on the FOSDEM-85-website.

I didn't want an USB-tail connected to the FOSDEM-85 so I took a small CR2032 battery together with a battery holder as a backpack. It's easy to connect it to the battery connector on the FOSDEM-85-PCB (see my photo). CR2032 has only 3 V and possibly that's why the program did not start. There is a diode D3 reducing the voltage. I shorted this diode D3. The ATTiny85 gets the whole voltage of 3 V and that's enough for running.

Software:

I installed the Arduino IDE offered by Digispark. A link is on the FOSDEM-85-website or ask Google. The sketch is attached starting at the following page of this doc.

You have to change the calibration of the internal temperature sensor. There is a link as a comment in the sketch to a description of Atmel how to do this.

I have tried to find every possibility to reduce power consumption for the mobile use with a small battery. But I have no measuring instrument to measure small currents. So I cannot say if I was successful enough for the use of a CR2032 cell but 2xAAA batteries is definitly for long time use.

Have fun

Stefan Goetze

```
/* FOSDEM-85 shows the temperature
 *
 * hardware modification: resistor R1 for GREEN LED cut from Vcc and connected to PB2
 * see: https://www.olimex.com/Products/Duino/AVR/FOSDEM-85/resources/FOSDEM-85_schematic.pdf
 *
 * Most code taken from
 * http://www.re-innovation.co.uk/web12/index.php/en/blog-75/306-sleep-modes-on-attiny85
 * and
 * http://forum.arduino.cc/index.php/topic,26299.0.html
 * and new for measuring vcc
 * http://www.mikrocontroller.net/topic/315667
 * -- Thanks to all for sharing this code --
 *
 * Fuses for 1 MHz:       09.04.2015 - !!! you should use this !!! - I measured less current
 *   Extended - 0xFF
 *   High -     0xD7
 *   Low -      0x62                  //clock divided by 8 - Bit 7 makes the difference
 *
 * Fuses for 8 MHz:
 *   Extended - 0xFF
 *   High -     0xD7
 *   Low -      0xE2
 *
 * in another small current project (SparkFun Big Time watch kit)
 * is reported that 8 MHz needs less current (but it is ATmega 328P)
 * change of clock needs recalibration !!!
 *
 * code revised & added some blinking
 * added lipo control (lipo connected with diode -added back production state)
 * S. Goetze
 *
 * added new FOSDEM-85 Volker S.
 * 2014-11-16
 *
 * added new for Ulrich P., changed MYATTINY45V1 to NORMANATTINY45V1 FOSDEM-85 given to Norman S.
 * added 1/10 display for calibration
 * 2015-02-14
 *
 * added avr/power.h functions and use of PRR during sleep for further power reduction
 * 2015-04-09
*/

#include <avr/io.h>
#include <avr/sleep.h>
#include <avr/interrupt.h>
#include <avr/power.h>
#include "arduino.h"

//*****************************************************
//uncomment only one definition of this area between asterisks
//#define CALIBRATION           //uncomment for calibration of temperature sensor and of led dimming
//#define MYATTINY85V1          //my SMD ATTINY85V on SMD adapter
//#define NORMANATTINY45V1                      //Norman S. ATTINY45V first test mcu
//#define CHATTINY45V1                          //Christian ATTINY45V mcu
//#define VOLKERATTINY45V1                //Volker S. ATTINY45V mcu
//#define ULRICHATTINY45V1                      //Ulrich P. ATTINY45V mcu // .
#define TWOATTINY45V1                   //ATTINY45V mcu // ..
//#define ONEATTINY85                           //ATTINY85 mcu // |
//#define TWOATTINY85                           //ATTINY85 mcu // ||

//add a new label and new #ifdef code
//or use code of an #ifdef section of these labels
//*****************************************************

//IO
#define ZEHNER_LED 2
#define EINER_LED 1
#define SHORT_FOR_VCC_DISPLAY 0     //shorten D0 to GND for display switching to voltage display

//voltage for lipo control (lipo voltage reduced by one diode = -0.7V)
#define IS_LIPO 3.4                         //4.1V - 0.7V for diode
#define WARN_LIPO 2.8         //3.5V - 0.7V
#define END_LIPO 2.5    //3.2V - 0.7V (let 0.2V over absolute minimum)

//brightness 0(min)...9(max - no dimming) for vcc = 1.7, 1.8 ... 3.0 (14 values)  - change so that both leds show (some more) the same
brightness
#ifdef CALIBRATION
//no dimming during calibration
const uint8_t zehnerdimm[14] = {9,9,9,9,9,9,9,9,9,9,9,9,9,9};     //let you display voltage value and change for your needs - left led
const uint8_t einerdimm[14] =  {9,9,9,9,9,9,9,9,9,9,9,9,9,9};   //change for right led - best to do if you have a variable power supply 1.7..3.0V
#endif
#ifdef MYATTINY85V1
//my SMD ATTINY85V and FOSDEM-85 with original LEDs and resistors (10k, 1.5k), clock with 1 MHz
const uint8_t zehnerdimm[14] = {9,9,9,9,9,9,9,9,9,8,7,6,5,4};
```

```
const uint8_t einerdimm[14] =  {0,0,0,0,0,3,7,8,9,9,9,9,9,9};
#endif
#ifdef MYATTINY45V1
//Norman S. ATTiny45V and FOSDEM-85 with 61 ohm (left yellow led) and 47 ohm (right red led) resistors, clock with 8 MHz
const uint8_t zehnerdimm[14] = {9,9,9,8,7,7,6,6,6,5,5,5,4,3};
const uint8_t einerdimm[14] =  {1,2,3,4,4,4,4,5,4,4,4,4,3,3};
#endif
#ifdef CHATTINY45V1
//Christians ATTiny45V and FOSDEM-85 with 1k (left yellow led) and 1.5k  (right red led) resistors, clock with 8 MHz
const uint8_t zehnerdimm[14] = {9,9,9,9,9,9,9,9,9,9,9,9,9,9};
const uint8_t einerdimm[14] =  {5,7,9,9,9,9,9,9,9,9,9,9,9,9};
#endif
#ifdef VOLKERATTINY45V1
//Volker S. ATTiny45V and FOSDEM-85 with 100 ohm (left yellow led) and 100 ohm (right red led) resistors, clock with 8 MHz
const uint8_t zehnerdimm[14] = {9,9,9,7,5,5,5,5,5,4,3,3,3,3};
const uint8_t einerdimm[14] =  {1,2,8,8,8,9,9,9,9,9,9,9,9,9};
#endif
#ifdef ULRICHATTINY45V1
//Ulrich P. ATTiny45V and FOSDEM-85 with 220 ohm (left yellow led) and 47  (right red led) resistors, clock with 8 MHz
const uint8_t zehnerdimm[14] = {9,9,9,9,9,9,9,9,9,9,9,9,9,9};
const uint8_t einerdimm[14] =  {2,5,6,6,7,7,8,9,9,9,8,7,6,5};
#endif
#ifdef TWOATTINY45V1
//ATTiny45V and FOSDEM-85 with 220 ohm (left yellow led) and 47  (right red led) resistors, clock with 1 MHz
const uint8_t zehnerdimm[14] = {9,9,9,9,9,9,9,9,9,9,9,9,9,9};
const uint8_t einerdimm[14] =  {2,5,6,6,7,7,8,9,9,9,8,7,6,5};
#endif
#ifdef ONEATTINY85
//ATTiny85 and FOSDEM-85 with 180 ohm (left blue led) and 560 ohm  (right green led) resistors, clock with 1 MHz, no more ATtiny45V left
over
const uint8_t zehnerdimm[14] = {9,9,9,9,9,9,9,9,9,9,9,9,9,9};
const uint8_t einerdimm[14] =  {9,9,9,9,9,9,9,9,9,9,8,7,6,5};
#endif
#ifdef TWOATTINY85
//ATTiny85 and FOSDEM-85 with 180 ohm (left blue led) and 560 ohm  (right green led) resistors, clock with 1 MHz, no more ATtiny45V left
over
const uint8_t zehnerdimm[14] = {9,9,9,9,9,9,9,9,9,9,9,9,9,9};
const uint8_t einerdimm[14] =  {9,9,9,9,9,9,9,9,9,9,8,7,6,5};
#endif

//calibration of sensor has to be changed for your ATtiny85
//look here: http://www.atmel.com/Images/doc8108.pdf
//
float chipTemp(float raw) {
#ifdef CALIBRATION
  const float chipTempOffset = 273.0;             // calibration
  const float chipTempCoeff = 1.0;
#endif
#ifdef MYATTINY85V1
  const float chipTempOffset = 274.5;             // my ATtiny85V with 1 MHz
  const float chipTempCoeff = 1.067;
#endif
#ifdef NORMANATTINY45V1
  const float chipTempOffset = 281.5;             // Norman S. ATtiny45V
  const float chipTempCoeff = 1.26;
#endif
#ifdef CHATTINY45V1
  const float chipTempOffset = 269.2;             // Christians ATtiny45V
  const float chipTempCoeff = 1.18;
#endif
#ifdef VOLKERATTINY45V1
const float chipTempOffset = 281.7;             // Volker S. ATtiny45V
const float chipTempCoeff = 1.0;
#endif
#ifdef ULRICHATTINY45V1
const float chipTempOffset = 285.3;             // Ulrich P. ATtiny45V
const float chipTempCoeff = 1.13;
#endif
#ifdef TWOATTINY45V1
const float chipTempOffset = 279.4;             // ATtiny45V with 1 MHz
const float chipTempCoeff = 1.073;
#endif
#ifdef ONEATTINY85
const float chipTempOffset = 279.61;             // ATTINY85 with 1 MHz
const float chipTempCoeff = 1.085;
#endif
#ifdef TWOATTINY85
const float chipTempOffset = 283.12;             // ATTINY85 with 1 MHz
const float chipTempCoeff = 1.116;
#endif

  return((raw - chipTempOffset) / chipTempCoeff);
}
```

```
//watchdog times for blinking / see function sleep_watchdog()

#define LED_START 1                     // 300 ms
#define LED_LONG 6        // 1 sec
#define LED_SHORT 2       // 64 ms
#define LED_GAP 5         // 500 ms
#define LED_LONGGAP 6     // 1 s
#define LED_PERIOD 9      // 8 sec

uint32_t gapTime(uint8_t gapCode) {   //map gapcode above to ms values for softPWM
  switch (gapCode) {
  case LED_LONG:  return 1000L;
  case LED_SHORT:               return 64L;
  case LED_GAP:                 return 500L;
  case LED_START:   return 300L;
  }
}

// Variables for LIPO detection
uint8_t firstStart = 0;
uint8_t isLipo;

// Variable for the Sleep/power down mode (not needed)
volatile uint8_t f_wdt;

// watchdock handling
// 0=16ms, 1=32ms,2=64ms,3=128ms,4=250ms,5=500ms
// 6=1 sec,7=2 sec, 8=4 sec, 9= 8sec
void sleep_watchdog(uint8_t ii) {
  uint8_t bb, mPRR;
  if (ii > 9 ) ii=9;
  bb = ii & 7;
  if (ii > 7)  bb |= _BV(WDP3);
  bb &= ~_BV(WDCE);                   //following 8.4.1.2, no. 2, page 43 ATtiny25/45/85 datasheet

  MCUSR &= ~_BV(WDRF);
  // start timed sequence
  WDTCR |= _BV(WDCE) | _BV(WDE);       //following 8.4.1.2, no. 1, page 43 ATtiny25/45/85 datasheet
  // set new watchdog timeout value
  WDTCR = bb;
  WDTCR |= _BV(WDIE);               //enable Interrupt
  f_wdt = 0;                        //not used

  mPRR = PRR;
  PRR = 0xFF;                                                                   // max. power reduction
  set_sleep_mode(SLEEP_MODE_PWR_DOWN); // sleep mode is set here
  sleep_enable();
  sleep_mode();                     // System actually sleeps here
  sleep_disable();                  // System continues execution here when watchdog timed out
  PRR = mPRR;
}

// Watchdog Interrupt Service / is executed when watchdog timed out
ISR(WDT_vect) {
  f_wdt=1;  // set global flag - used as noop
}

//Blinking both LEDs with PWM made by programm
void softPWM (uint8_t zehnerVal, uint8_t einerVal, uint8_t gapVal, uint8_t vccVal) {
  uint8_t i;
  uint32_t waitTime = gapTime(gapVal);
  uint32_t starttime = millis();
  while ((uint32_t)(millis() - starttime) <= waitTime) {
              i=0;
              while (i < 10) {
                digitalWrite(ZEHNER_LED, (i <= zehnerdimm[vccVal])?zehnerVal:LOW);
                digitalWrite(EINER_LED, (i <= einerdimm[vccVal])?einerVal:LOW);
                i++;
              }
  }
  digitalWrite(ZEHNER_LED, LOW);
  digitalWrite(EINER_LED, LOW);
}

//common code for both sources of adc conversion
int getADC() {
  ADCSRA  |=_BV(ADSC);                                                          // Start conversion
  while((ADCSRA & _BV(ADSC)));         // Wait until conversion is finished
  return ADC;
}

void setup() {
  uint8_t i;
  for (i=0; i<6; i++); {          //all pins input - unused pins stay so
```

```
    pinMode(i, INPUT);
            digitalWrite(i, LOW);               //switching pullups off
  }
  power_timer1_disable();                       //only timer0 for for delay()
  power_usi_disable();                                  //no usi
  // one adc conversion to discard
  ADCSRA &= ~(_BV(ADATE) |_BV(ADIE));   // Clear auto trigger and interrupt enable
  ADCSRA |= _BV(ADEN);                  // Enable AD and start conversion
  ADMUX = 0xF | _BV( REFS1 );           // ADC4 (Temp Sensor) and Ref voltage = 1.1V;
  delay(100);                           // Settling time min 1 ms we have time 100 ms
  getADC();
  firstStart = 1;                       //for LIPO detection
}

void loop() {
  int t_celsius, i;
  uint8_t vccIndex, zehner, einer;
  float rawTemp;
  float rawVcc;
  boolean tempMes = true;

  //measure temperature
//  ADCSRA &= ~(_BV(ADATE) |_BV(ADIE));   // Clear auto trigger and interrupt enable
  ADCSRA |= _BV(ADEN);                  // Enable AD and start conversion
  ADMUX = 0xF | _BV( REFS1 );           // ADC4 (Temp Sensor) and Ref voltage = 1.1V;
  delay(100);                           // Settling time min 1 ms we have time 100 ms

  rawTemp = (float)getADC();            // use next sample as initial average
  for (int i=2; i<2000; i++) {
    rawTemp += ((float)getADC() - rawTemp) / float(i); // calculate running average
  }
  ADCSRA &= ~(_BV(ADEN));               // disable ADC

  //measure vcc
//  ADCSRA &= ~(_BV(ADATE) |_BV(ADIE));         // Clear auto trigger and interrupt enable
  ADCSRA |= _BV(ADEN);                          // Enable ADC
  ADMUX  = 0x0c | _BV(REFS2);                   // Use VCC as voltage reference, select bandgap reference as ADC input
  delay(100);                           // Settling time min 1 ms we have time 100 ms

  rawVcc = (float)getADC();             // use next sample as initial average
  for (int i=2; i<2000; i++) {
            rawVcc += ((float)getADC() - rawVcc) / float(i); // calculate running average
  }
  ADCSRA &= ~(_BV(ADEN));               // disable ADC

  rawVcc = 1024 * 1.1f / rawVcc;
  vccIndex = min(max(17,(uint8_t)(rawVcc * 10)),30) - 17;       //index 0..13 for vcc 1.7 ... 3.0

  if (firstStart) {                                                     //run once after setup() for LiPo detection
            isLipo = (rawVcc >= IS_LIPO);
            firstStart = 0;
            if (isLipo) {
              pinMode(ZEHNER_LED, OUTPUT);
              pinMode(EINER_LED, OUTPUT);
              for (i=0; i<3; i++) {         //LiPo detected -> green off, red+green 3 x alternately blinking
                        softPWM(LOW, HIGH, LED_START, vccIndex);
                        softPWM(HIGH, LOW,  LED_START, vccIndex);
              }
            }
  }

  if (isLipo) {                                                 //check if LiPo
              if (rawVcc <= END_LIPO) {                         //voltage down - LiPo empty
                        sleep_watchdog(LED_PERIOD);             //don't switch on LEDs anymore - only sleep and hope for LiPo change soon
                        return;                                 /finish this loop here
              }
              if (rawVcc <= WARN_LIPO) {
                pinMode(ZEHNER_LED, OUTPUT);
                pinMode(EINER_LED, OUTPUT);
                for (i=0; i<5; i++) {         //LiPo battery low warning -> green off, red+green 5 x alternately blinking
                          softPWM(LOW, HIGH, LED_START, vccIndex);
                          softPWM(HIGH, LOW,  LED_START, vccIndex);
                 }
                        pinMode(ZEHNER_LED, INPUT);     //is said to save current
                        pinMode(EINER_LED, INPUT);
                        sleep_watchdog(LED_PERIOD);
                        return;
              }
  }

#ifdef CALIBRATION
  t_celsius = (int)(chipTemp(rawTemp));                                 // not rounding
#else
```

```
//  t_celsius = (int)(chipTemp(rawTemp) + (float)vccIndex / 13.0f);  //new: compensation: temperature measured with 3.0 V VCC is 1degree lower than measured with 1.7 V VCC - not for all ATtiny - taken out
  t_celsius = (int)(chipTemp(rawTemp));                    // possibly rounding up with + 0.5f ??? - my impression was that displayed temperature is a little too high - so I took it out
#endif

  zehner = abs(t_celsius/10);
  einer = abs(t_celsius % 10);

  digitalWrite(SHORT_FOR_VCC_DISPLAY,HIGH);                                    //enable pull up
  delay(1);
  if (digitalRead(SHORT_FOR_VCC_DISPLAY) == LOW) {  //display vcc instead of temp if LOW
    zehner = (uint8_t)rawVcc;
    einer =  ((uint8_t)(rawVcc * 10)) % 10;
              t_celsius = 0;                                                   //for displaying no minus with voltage display - sure is sure
              tempMes = false;                                                 //no display of 1/10 degree in calibration mode
  }
  digitalWrite(SHORT_FOR_VCC_DISPLAY,LOW);                                     //disable pullup

  pinMode(ZEHNER_LED, OUTPUT);
  pinMode(EINER_LED, OUTPUT);

  for (i=0; i<3; i++) {          //temp >= 0 -> green and red long / < 0  -> green long on, red 3 x blinking
    softPWM(HIGH, HIGH, LED_START, vccIndex);
    softPWM(HIGH, (t_celsius < 0)?LOW:HIGH, LED_START, vccIndex);
  }
  sleep_watchdog(LED_LONGGAP);

  if (zehner >= 5) {            //zehner >= 5  - one time long for 5
    softPWM(HIGH, LOW, LED_LONG, vccIndex);
              sleep_watchdog(LED_GAP);
              zehner -= 5;
  }
  for (i=0; i<zehner; i++) {    //remaining 0..4 - short blinking
    softPWM(HIGH, LOW, LED_SHORT, vccIndex);
              sleep_watchdog(LED_GAP);
  }

  if (einer >= 5) {             //einer >= 5  - one time long for 5
              softPWM(LOW, HIGH, LED_LONG, vccIndex);
              sleep_watchdog(LED_GAP);
              einer -= 5;
  }
  for (i=0; i<einer; i++) {     //remaining 0..4 - short blinking
              softPWM(LOW, HIGH, LED_SHORT, vccIndex);
              sleep_watchdog(LED_GAP);
  }

#ifdef CALIBRATION                                                             //display 1/10 degree with "zehner"-LED only for calibration

  if (tempMes) {
    zehner = abs(((int)(chipTemp(rawTemp) * 10.0f)) % 10);
    sleep_watchdog(LED_LONGGAP);

    if (zehner >= 5) {            //zehner >= 5  - one time long for 5
                softPWM(HIGH, LOW, LED_LONG, vccIndex);
                sleep_watchdog(LED_GAP);
                zehner -= 5;
               }
    for (i=0; i<zehner; i++) {    //remaining 0..4 - short blinking
                softPWM(HIGH, LOW, LED_SHORT, vccIndex);
                sleep_watchdog(LED_GAP);
    }
  }

#endif

  pinMode(ZEHNER_LED, INPUT);      //is said to save current
  pinMode(EINER_LED, INPUT);

  sleep_watchdog(LED_PERIOD);      //the long sleep - thinking about reset was a wrong way
}
```