Systemtechnik

Hardwarenahe Programmierung

Protokoll 1: LCD

Verfasser:

- **Jonas Ruhland**
- **Clemens Rumpfhuber**

Übungsleiter: DI Mag. Ferdinand Hell

Theoretische Ausarbeitung

Textverarbeitung: sprintf

Der Befehl SPRINTF(…) bietet die optimale Möglichkeiten zur Ausgabeformatierung von Werten.

Syntax

*int sprintf (char * str, const char * format, …);*

str … Pointer zum Buffer, wo der formatierte String abgespeichert wird
format … Formatierter String
… … zusätzliche Argumente

Format	Bedeutung
%d %i	Decimal signed Integer
%o	Octal Integer
%x %X	Hex Integer
%u	Unsigned Integer
%c	Character
%s	String
%f	Double
%e %E	Double
%g %G	Double
%p	Zeiger
%n	Number of characters written by this printf. No argument expected
%% %	No argument expected.

Beispiel

```
#include <stdio.h>

int main ()
{
  char buffer [50];
  int n, a=5, b=3;

  n=sprintf (buffer, "%d plus %d is %d", a, b, a+b);
  printf ("[%s] is a string %d chars long\n“, buffer, n);

  return 0;
}
```

Sensoren

Auslesen der Raspberry Pi CPU Temperatur

```c
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "../libs/lcd_i2c.h"
#include <wiringPi.h>

#define LCD_I2C_BACKLIGHT_ON(lcd_p) lcd_i2c_backlight(lcd_p, 1)
#define LCD_I2C_BACKLIGHT_OFF(lcd_p) lcd_i2c_backlight(lcd_p, 0)

// option flags
int8_t opt_address = 0x27;
int8_t opt_cols = 16;
int8_t opt_rows = 2;

FILE *temperatureFile;
double temperature;
char formatedText[30];

int main() {
    // setup wiring pi
    wiringPiSetupPhys();

    // init i2c
    lcd_i2c_t lcd = {0};
    if (lcd_i2c_setup(&lcd, opt_address) == -1)
        fprintf(stderr, "Error intialising PCF8574 at address i2c 0x%02x: %s\n",
opt_address, strerror(errno));

    // init lcd
    lcd_i2c_init(&lcd);

    // geometry
    lcd.rows = opt_rows;
    lcd.cols = opt_cols;
    // turn backlight on
    LCD_I2C_BACKLIGHT_ON(&lcd);

    // open file
    temperatureFile = fopen ("/sys/class/thermal/thermal_zone0/temp", "r");

    // check if file could be opened
    if (temperatureFile != NULL) {
        // read temeperature
        fscanf (temperatureFile, "%lf", &temperature);

        // process temperature
        temperature /= 1000;

        // close file stream
        fclose (temperatureFile);
    }

    // format temp into string
    sprintf(formatedText, "Temp: %.2f °C", temperature);

    // set string on display
    lcd_i2c_puts(&lcd, formatedText);

    return 0;
}
```

Auslesen eines DHT11

```
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "../libs/lcd_i2c.h"

#include <wiringPi.h>
#include <wiringPiI2C.h>

#define LCD_I2C_BACKLIGHT_ON(lcd_p) lcd_i2c_backlight(lcd_p, 1)
#define LCD_I2C_BACKLIGHT_OFF(lcd_p) lcd_i2c_backlight(lcd_p, 0)

// option flags
int8_t opt_address = 0x27;
int8_t opt_cols = 16;
int8_t opt_rows = 2;

#define DHTMAXTIMINGS 85
#define DHTPIN 11

void readDHT(double *temperature, double *humidity, int *worked) {
    int dht11_dat[5] = {0, 0, 0, 0, 0};
    int dhtCounter = 0;

    while (*humidity == 0 && *temperature == 0 && dhtCounter < 5) {
        // predefine values
        uint8_t laststate = HIGH, counter, j = 0, i;
        dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;

        // set pin mode
        pinMode(DHTPIN, OUTPUT);
        digitalWrite(DHTPIN, LOW);
        delay(18);
        digitalWrite(DHTPIN, HIGH);
        delayMicroseconds(40);
        pinMode(DHTPIN, INPUT);

        // digital read
        for (i = 0; i < DHTMAXTIMINGS; i++) {
            counter = 0;
            while (digitalRead(DHTPIN) == laststate) {
                counter++;
                delayMicroseconds(1);
                if (counter == 255) { break; }
            }
            laststate = digitalRead(DHTPIN);

            if (counter == 255) break;

            if ((i >= 4) && (i % 2 == 0)) {
                dht11_dat[j / 8] <<= 1;
                if (counter > 50)
                    dht11_dat[j / 8] |= 1;
                j++;
            }
        }

        // calculate values
        if ((j >= 40) && (dht11_dat[4] == ((dht11_dat[0] + dht11_dat[1] + dht11_dat[2] +
dht11_dat[3]) & 0xFF))) {
            *temperature = dht11_dat[0] + (dht11_dat[1] / 10);
            *humidity = dht11_dat[2] + (dht11_dat[3] / 10);
            *worked = HIGH;
        } else
            *worked = LOW;

        dhtCounter++;
```

```
    }
}

int main() {
    char formatedText[30];
    int worked = 0;
    double temperature = 0, humidity = 0;

    // setup wiring pi
    wiringPiSetupPhys();

    // init i2c
    lcd_i2c_t lcd = {0};
    if (lcd_i2c_setup(&lcd, opt_address) == -1)
        fprintf(stderr, "Error intialising PCF8574 at address i2c 0x%02x: %s\n",
opt_address, strerror(errno));

    // init lcd
    lcd_i2c_init(&lcd);

    // geometry
    lcd.rows = opt_rows;
    lcd.cols = opt_cols;

    // turn backlight on
    LCD_I2C_BACKLIGHT_ON(&lcd);

    // read bh1750 for 20 seconds
    int i = 0;
    while (i++ < 20) {
        // read sensor
        readDHT(&temperature, &humidity, &worked);

        lcd_i2c_clear(&lcd);

        if (worked == 1) {
            lcd_i2c_gotoxy(&lcd, 0, 0);
            sprintf(formatedText, "Temp: %.2f", temperature);
            lcd_i2c_puts(&lcd, formatedText);

            lcd_i2c_gotoxy(&lcd, 0, 1);
            sprintf(formatedText, "Humi: %.2f", humidity);
            lcd_i2c_puts(&lcd, formatedText);
        } else
            lcd_i2c_puts(&lcd, "not working...");

        // wait a second
        delay(1000);
    }

    // write finish msg
    lcd_i2c_clear(&lcd);
    lcd_i2c_puts(&lcd, "finished!");

    return 0;
}
```

Auslesen eines BH1750

```c
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../libs/lcd_i2c.h"
#include <wiringPi.h>
#include <wiringPiI2C.h>

#define LCD_I2C_BACKLIGHT_ON(lcd_p) lcd_i2c_backlight(lcd_p, 1)
#define LCD_I2C_BACKLIGHT_OFF(lcd_p) lcd_i2c_backlight(lcd_p, 0)

// option flags
int8_t opt_address = 0x27;
int8_t opt_cols = 16;
int8_t opt_rows = 2;

void readBH(double *lux) {
    int handle = wiringPiI2CSetup(0x5C);
    wiringPiI2CWrite(handle, 0x10);
    delay(1000);
    int word = wiringPiI2CReadReg16(handle, 0x00);
    *lux = ((word & 0xff00) >> 8) | ((word & 0x00ff) << 8);
}

int main() {
    double lux = 0;
    char formatedText[30];

    // setup wiring pi
    wiringPiSetupPhys();

    // init i2c
    lcd_i2c_t lcd = {0};
    if (lcd_i2c_setup(&lcd, opt_address) == -1)
        fprintf(stderr, "Error intialising PCF8574 at address i2c 0x%02x: %s\n",
opt_address, strerror(errno));
    // init lcd
    lcd_i2c_init(&lcd);
    // geometry
    lcd.rows = opt_rows;
    lcd.cols = opt_cols;
    // turn backlight on
    LCD_I2C_BACKLIGHT_ON(&lcd);

    // read bh1750 for 20 seconds
    int i = 0;
    while (i++ < 20) {
        // read sensor
        readBH(&lux);

        // format output
        sprintf(formatedText, "BH1750: %.2f lx", lux);

        // clear text and write to lcd
        lcd_i2c_clear(&lcd);
        lcd_i2c_puts(&lcd, formatedText);

        // wait a second
        delay(1000);
    }

    // write finish msg
    lcd_i2c_clear(&lcd);
    lcd_i2c_puts(&lcd, "finished!");

    return 0;
}
```

Auslesen eines ADS1115

```c
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wiringPi.h>
#include <math.h>

#include "../libs/lcd_i2c.h"

// option flags
int8_t opt_address = 0x27;
int8_t opt_cols = 16;
int8_t opt_rows = 2;

#define LCD_I2C_BACKLIGHT_ON(lcd_p) lcd_i2c_backlight(lcd_p, 1)
#define LCD_I2C_BACKLIGHT_OFF(lcd_p) lcd_i2c_backlight(lcd_p, 0)

#define ADSPIN 120
#define SHH_V_IN 5.0
#define SSH_K 9.5
#define ADS_MAX (1 < 14)
const double SHH_A = 1.129148e-3;
const double SHH_B = 2.34125e-4;
const double SHH_C = 8.76741e-8;

double steinhartHart(double resistance) {
    double log_r  = log(resistance);
    double log_r3 = log_r * log_r * log_r;
    return 1.0 / (SHH_A + SHH_B * log_r + SHH_C * log_r3);
}

double getTempKelvin(double adcRaw, double _resistance) {
    double voltage = adcRaw / ADS_MAX * SHH_V_IN;
    double resistance = ((ADS_MAX * _resistance / adcRaw) - _resistance);
    // Account for dissipation factor K
    return steinhartHart(resistance) - voltage * voltage / (SSH_K * _resistance);
}

double getTempCelsius(double adcRaw, double _resistance) {
    return getTempKelvin(adcRaw, _resistance) - 273.15;
}

void readADS(double *light, double *temperature) {
    double sensorLight = analogRead(ADSPIN + 2), sensorTemperature = analogRead(ADSPIN +
1);
    *temperature = sensorTemperature / 1000;
    *light = sensorLight / 100;
}

void readSteinhartHart(double *temperature, double _resistance) {
    *temperature = getTempCelsius(analogRead(ADSPIN + 1), _resistance);
}

int main() {
    double light = 0, temperature = 0;
    char formatedText[30];

    // setup wiring pi
    wiringPiSetupPhys();

    // init i2c
    lcd_i2c_t lcd = {0};
    if (lcd_i2c_setup(&lcd, opt_address) == -1)
        fprintf(stderr, "Error intialising PCF8574 at address i2c 0x%02x: %s\n",
opt_address, strerror(errno));

    // init lcd
    lcd_i2c_init(&lcd);
```

```
    // geometry
    lcd.rows = opt_rows;
    lcd.cols = opt_cols;

    // turn backlight on
    LCD_I2C_BACKLIGHT_ON(&lcd);

    // read ads1115 for 20 seconds
    int i = 0;
    while (i++ < 20) {
        // read sensor
        readADS(&light, &temperature);

        // clear lcd
        lcd_i2c_clear(&lcd);

        // print line 1 of lcd
        lcd_i2c_gotoxy(&lcd, 0, 0);
        sprintf(formatedText, "Light: %.2f", light);
        lcd_i2c_puts(&lcd, formatedText);

        // print line 2 of lcd
        lcd_i2c_gotoxy(&lcd, 0, 1);
        sprintf(formatedText, "Temp: %.2f", temperature);
        lcd_i2c_puts(&lcd, formatedText);

        // wait a second
        delay(1000);
    }

    // write finish msg
    lcd_i2c_clear(&lcd);
    lcd_i2c_gotoxy(&lcd, 0, 0);
    lcd_i2c_puts(&lcd, "finished!");

    return 0;
}
```