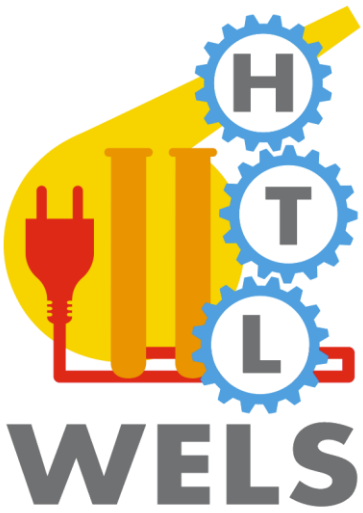


Systemtechnik

Hardwarenahe Programmierung

Thema:

- **Liquid-Crystal Display**

Professor:

- **DI Mag. Ferdinand Hell**

Autor:

- **Serifi Argjent**
- **Steinmaurer Rene**

Datum:

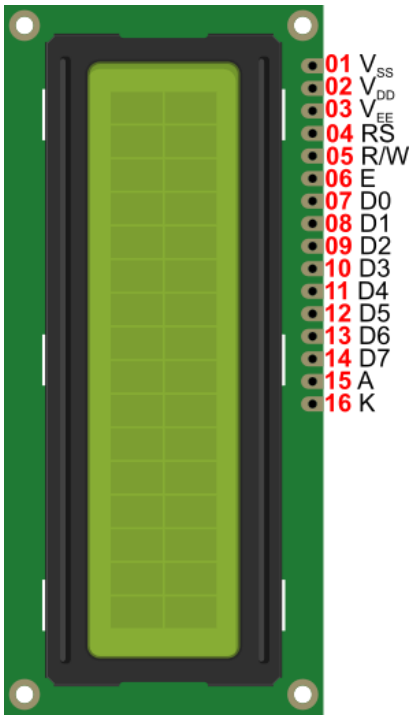
- **16.10.2021**
- **30.10.2021**

Inhaltsverzeichnis

Theorie

Ein LCD (Liquid Crystal Display / Flüssigkristallanzeige) besitzt kleine Segmente, in denen Flüssigkristalle im ungeordneten Zustand nahezu kein Licht passieren lassen. Legt man eine Spannung an, richten sie sich so aus, dass sie lichtdurchlässig werden. Die Gestaltung der einzelnen Segmente ist beliebig. Für die Darstellung von beliebigen Zeichen oder Symbolen wird eine Punktmatrix verwendet. Es existieren verschiedenen Größen, die sich durch die Anzahl der Zeilen und Zeichen pro Zeile unterscheiden. Da es aufgrund der hohen Anzahl unmöglich ist, jeden einzelnen Punkt separat anzusteuern, besitzen marktübliche LCD bereits eine Ansteuerungselektronik. Es gibt sie entweder mit parallelem Bus (in der Regel 14 oder 16 Anschlusspins) oder seriellem Bus (meist I^2C).

Pinbelegung



Pin 1 (V_{SS}) Pin 2 (V_{DD})	Dient zur Stromversorgung des Displays und der Ansteuerungselektronik. Pin 1 ist auf Masse zu legen, Pin 2 sind +5 V Versorgungsspannung zuzuführen.
Pin 3 (V_{EE})	Der analoge Eingang dient zur Kontrastregelung des Displays. Der Wert muss zwischen 0 V und +5 V liegen.
Pin 4 (RS)	Der digitale Eingang bestimmt, ob die zum Display übermittelten Datenbits als Befehl (LOW) oder Zeichendaten (HIGH) interpretiert werden sollen.
Pin 5 (R/W)	Der digitale Eingang entscheidet, ob Daten auf dem Display geschrieben (LOW) oder vom Display eingelesen (HIGH) werden sollen.
Pin 6 (E)	Ein digitaler Eingang, der auf HIGH geschaltet werden muss, damit das Display die an den Datenpins anliegenden Bits ausliest.
Pin 7 – Pin 14 (D0 – D7)	8 Bits des bidirektionalen, parallelen Datenbusses.
Pin 15 (A) Pin 16 (K)	Existieren nur bei eingebauter Hintergrundbeleuchtung und dienen zu dieser Stromversorgung. An Pin 15 (Anode) kommt die Versorgungsspannung, Pin 16 (Kathode) wird auf Masse gelegt.

Quelle: https://rotering-net.de/tut/arduino/lcd-ansteuern.html

Befehle

Einige relevante Befehle in Hex Code, welche indirekt zur Anwendung kamen:

0F	LCD an, cursor aus
0E	Display an, cursor blinken
OC	Display an, cursor aus
08	Display aus, cursor aus
01	Display leeren
04	cursor reduzieren (cursor nach links verschieben)
06	cursor erhöhen (cursor nach rechts verschieben)
05	Display rechts verschieben
07	Display links verschieben
3C	Aktivierung der 2. Zeile
C1	Sprung zur 2. Zeile 1. Position
C2	Sprung zur 2. Zeile 2. Position
80	Cursor zur 1. Zeile zwingen
C0	Cursor zur 2. Zeile zwingen

Quelle: https://rotering-net.de/tut/arduino/lcd-ansteuern.html

Praxis

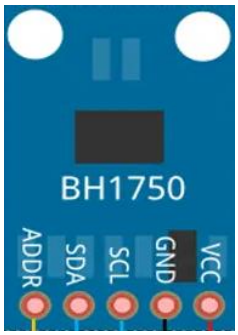
Programm

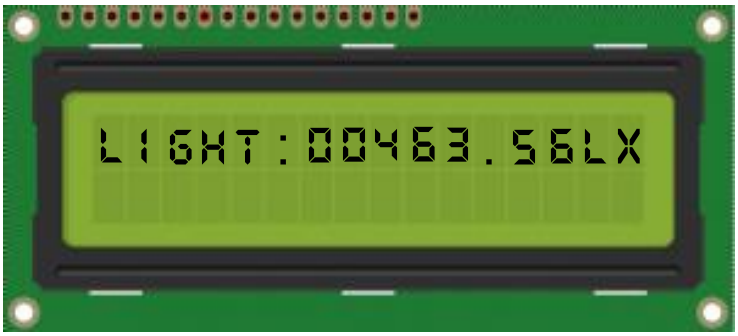
```
#include <stdio.h>
#include <stdint-gcc.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <stdlib.h>
#include "lcd_i2c.h"

#define DHTMAXTIMINGS 85
#define DHTPIN 11

int8_t opt_cursor = 0; // 0 an, 1 aus, 2 blinken
int8_t opt_address = LCD_I2C_PCF8574_ADDRESS_DEFAULT; // Adresse des LCDs
int8_t opt_backlight = 1; // -1 nichts machen, 0 ausschalten, 1 einschalten
int8_t opt_clear = 1; // 0 vorhandenen Text am Display behalten,
                      1 vorhandenen Text am Display entfernen
int8_t opt_cols = 16; // 16 Zeichen pro Zeile
int8_t opt_rows = 2; // 2 Zeilen am Display
int8_t opt_init; // Initialisierung
int8_t opt_x = 0; // Positierung x-Achse
int8_t opt_y = 0; // Positierung y-Achse

void printOnLCD(char *array) {
    // i2c Initialisierung
    lcd_i2c_t lcd = {0}; // lcd vom struct lcd_i2c_t erstellen
    if (lcd_i2c_setup(&lcd, opt_address) == -1) {
        // Bei einem Fehler (return -1) eine Fehlermeldung mit der Adresse ausgeben
        printf(stderr, "Error intialising PCF8574 at address i2c 0x%02x: %s\n", opt_address);
    }

    // LCD Initialisierung
    if (opt_init)
        lcd_i2c_init(&lcd);

    // Backlight des LCD ein-/ausschalten
    if (opt_backlight == 0)
        LCD_I2C_BACKLIGHT_OFF(&lcd);
    else if (opt_backlight == 1)
        LCD_I2C_BACKLIGHT_ON(&lcd);

    // vorherigen Text am LCD entfernen/behalten
    if (opt_clear)
        lcd_i2c_clear(&lcd);

    // Positionierung des Textes am LCD
    lcd_i2c_gotoxy(&lcd, opt_x, opt_y);

    // Ausgabe des Textes am LCD
    lcd_i2c_puts(&lcd, array);
}
```

```
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);
}

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

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

        // Pinmode setzen
        pinMode(DHTPIN, OUTPUT);
        digitalWrite(DHTPIN, LOW);
        delay(18);
        digitalWrite(DHTPIN, HIGH);
        delayMicroseconds(40);
        pinMode(DHTPIN, INPUT);

        // einlesen der Werte
        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++;
            }
        }

        // Werte berechnen
        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);
        }

        dhtCounter++;
    }
}
```

```
void printBrightnessOnLCD() {
    double lux = 0.0; //Variable für die Helligkeit
    char lcdOutput[16] = {}; //Char Array für das LCD

    // Helligkeit einlesen
    readBH(&lux);

    // Double von Helligkeit in Char Array umwandeln
    sprintf(lcdOutput, "Light: %6.2lflx", lux);

    // Helligkeit am LCD ausgeben
    printOnLCD(lcdOutput);
}

void printTemperatureOnLCD() {
    double humidity = 0.0, temperature = 0.0; Variable für die Temperatur
    char lcdOutput[16] = {}; Char Array für das LCD

    // Temperatur einlesen
    readDHT(&temperature, &humidity);

    // Double von Temperatur in Char Array umwandeln
    sprintf(lcdOutput, "Temp: %7.2lf°C", temperature);

    // Temperatur am LCD ausgeben
    printOnLCD(lcdOutput);
}

int main() {
    wiringPiSetupPhys();
    //printTemperatureOnLCD();
    printBrightnessOnLCD();
}
```

Resultat





sprintf()

Diese Funktion erhält man mit der stdio.h Bibliothek und dient zur Formatierung eines Textes, indem es ein Char-Array aus verschiedenen Variablen erstellt. In unserem Fall mussten wir einen Double (Helligkeit oder Temperatur) in ein Char-Array umwandeln.

```
sprintf(ziel, "%x.ylf", quelle);
```

ziel – in unserem Fall das Char-Array wohin es formatiert werden soll

% - beginnt ein Formatelement / Anweisung an diese Logik

x – Stellen vor dem Komma

y – Stellen nach dem Komma

lf – Datentyp: in unserem Fall longfloat

quelle – in unserem Fall der Double Wert, der formatiert werden soll

%c	char
%i	int
%u	unsigned
%f	float
%s	string