LK-Temp2

Inhaltsverzeichnis

Bild

Technische Daten / Kurzbeschreibung

Chipsatz: DS18B20 | Kommunikationsprotokoll: 1-Wire

9- 12Bit genaue Temperaturmessung im Meßbereich von –55°C bis +125°C

Bitte beachten Sie, dass veränderte Versionen der Zusatzbibliotheken „OneWire“ und „DallasTemperature Control Library“ erforderlich sind, um den Sensor problemlos verwenden zu können.

Die originale OneWire Bibliothek, ursprünglich von Paul Stoffregen entwickelt, ist auf dem Arduino Playground verfügbar.

Die originale DallasTemperature Control Library, entwickelt von Tom De Boer, ist auf GitHub verfügbar.

Die Bibliotheken, sowie der Beispielcode, sind außerdem im Downloadbereich hier erhältlich.

Codebeispiel Arduino

```
// Benötigte Libraries werden importiert
#include
#include
// Hier wird der Eingangs-Pin deklariert, an dem das Sensor-Modul angeschlossen ist
#define KY001_Signal_PIN 4
// Libraries werden konfiguriert
OneWire oneWire(KY001_Signal_PIN);
DallasTemperature sensors(&oneWire);
void setup() {
// Initialisierung Serielle Ausgabe
Serial.begin(9600);
Serial.println("KY-001 Temperaturmessung");
// Sensor wird initialisiert
sensors.begin();
}
//Hauptprogrammschleife
void loop()
{
// Temperaturmessung wird gestartet...
sensors.requestTemperatures();
// ... und gemessene Temperatur ausgeben
Serial.print("Temperatur: ");
Serial.print(sensors.getTempCByIndex(0));
Serial.write(176); // UniCode-Angabe eines char-Symbols für das "°-Symbol"
Serial.println("C");
delay(1000); // 1s Pause bis zur nächsten Messung
}
```

Codebeispiel Raspberry

```
import glob
import time
from time import sleep
import RPi.GPIO as GPIO
# An dieser Stelle kann die Pause zwischen den einzelnen Messungen eingestellt werden
sleeptime = 1
# Der One-Wire EingangsPin wird deklariert und der integrierte PullUp-Widerstand aktiviert
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# Nach Aktivierung des Pull-UP Widerstandes wird gewartet,
# bis die Kommunikation mit dem DS18B20 Sensor aufgebaut ist
print 'Warte auf Initialisierung...'
base_dir = '/sys/bus/w1/devices/'
while True:
```



```
        try:
                device_folder = glob.glob(base_dir + '28*')[0]
                break
        except IndexError:
                sleep(0.5)
                continue
device_file = device_folder + '/w1_slave'
# Funktion wird definiert, mit dem der aktuelle Messwert am Sensor ausgelesen werden kann
def TemperaturMessung():
        f = open(device_file, 'r')
        lines = f.readlines()
        f.close()
        return lines
# Zur Initialisierung, wird der Sensor einmal "blind" ausgelesen
TemperaturMessung()
# Die Temperaturauswertung: Beim Raspberry Pi werden erkennte one-Wire Slaves im Ordner
# /sys/bus/w1/devices/ einem eigenen Unterordner zugeordnet. In diesem Ordner befindet sic
# in dem Die Daten, die über dem One-Wire Bus gesendet wurden gespeichert.
# In dieser Funktion werden diese Daten analysiert und die Temperatur herausgelesen und au
def TemperaturAuswertung():
        lines = TemperaturMessung()
        while lines[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines = TemperaturMessung()
        equals_pos = lines[1].find('t=')
        if equals_pos != -1:
                temp_string = lines[1][equals_pos+2:]
                temp_c = float(temp_string) / 1000.0
                return temp_c
# Hauptprogrammschleife
# Die gemessene Temperatur wird in die Konsole ausgegeben - zwischen den einzelnen Messung
# ist eine Pause, deren Länge mit der Variable "sleeptime" eingestellt werden kann
try:
        while True:
                print '---------------------------------------'
                print "Temperatur:", TemperaturAuswertung(), "°C"
                time.sleep(sleeptime)
except KeyboardInterrupt:
        PIO.cleanup()
```

Downloads

LK-Temp2_Anleitung (Deutsch).pdf

LK-Temp2_Manual (English).pdf

LK-Temp2_Arduino-Libraries.zip

LK-Temp2_Arduino-Example.zip

LK-Temp2_Raspberry-Example.zip

