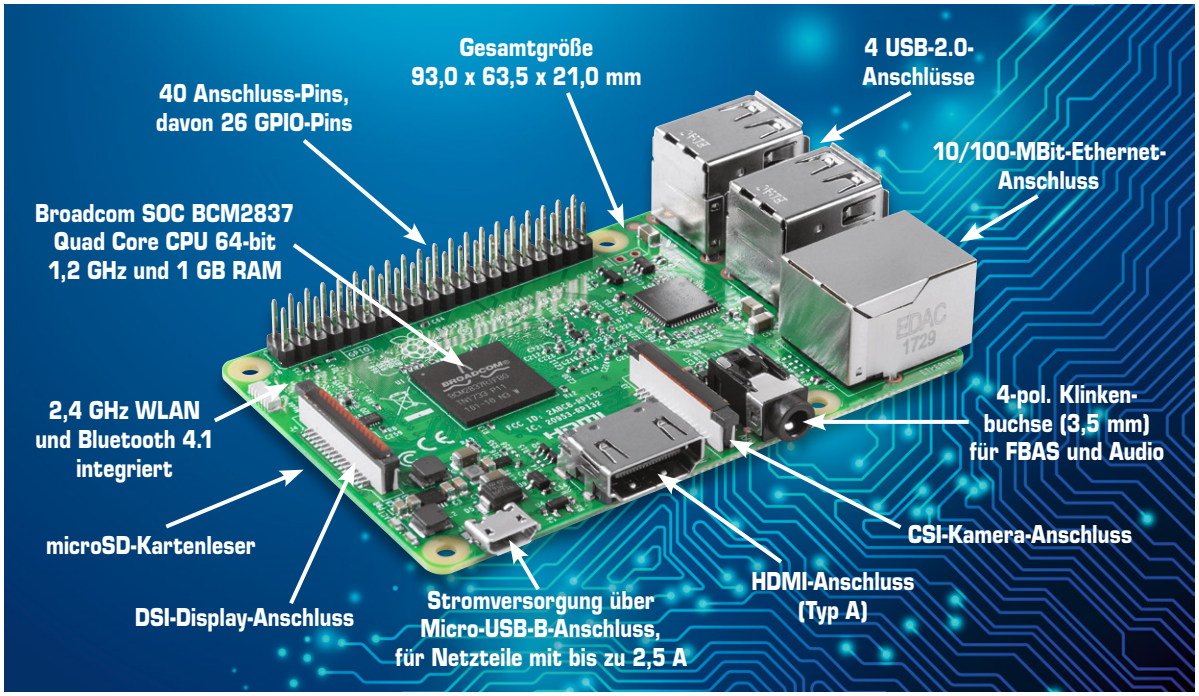
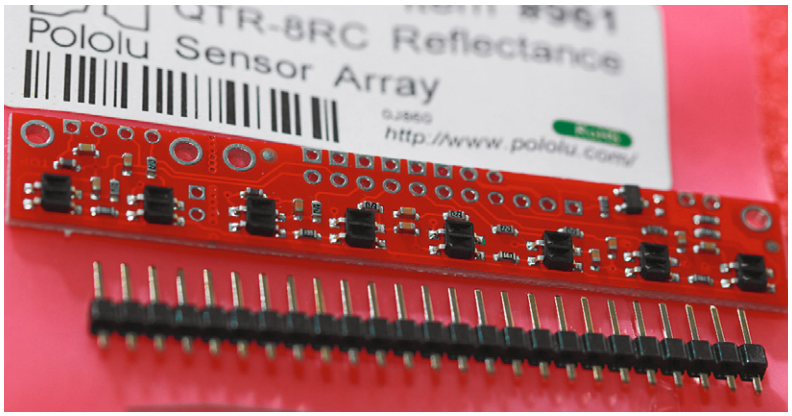
Raspberry Pi

Teil 9: Infrarotsensor – Schwarz und Weiß auf der Linie

Artikelserie
im ELV Shop
#10036

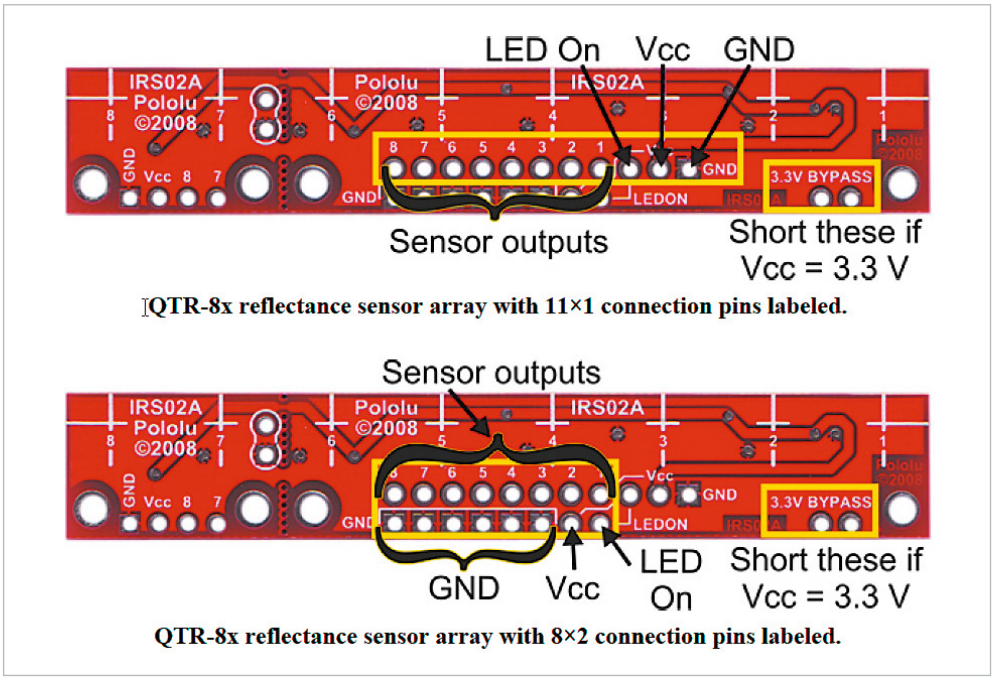


Mit einem Infrarot-Sensor lässt sich im Allgemeinen die Distanzänderung zwischen zwei Objekten feststellen – bei einer Reihe von Infrarot-Sensoren wie bei dem QTR-8RC-Sensor-Array lassen sich Bewegungen und Abstände relativ zuverlässig erkennen. Gerade bei vielen Selbstbauprojekten auf Arduino-Basis kommt das Pololu-QTR-8RC-Sensor-Array zum Einsatz, was sich jedoch auch mit dem Raspberry Pi verwenden lässt.



Lange Pinleiste: Neben den acht IR-Sensoren werden neben der Spannungsversorgung auch die Masse sowie der Schalter für die IR-LEDs angeschlossen.

Falls für den Sensor eine Stromversorgung von 3,3 V verwendet werden soll, ist laut Datenblatt noch eine 2-Pin-Steckbrücke in die Bohrungen des 3,3-V-BYPASS-Anschlusses einzulöten, damit dort später einfach per Jumper-Konfiguration die benötigte Spannung bequem eingestellt werden kann. Wird dann ein Jumper gesteckt, ist der Sensor für eine Spannung von 3,3 V eingerichtet, anderenfalls arbeitet der Sensor mit 5 V.



Verschiedene Anschlussmöglichkeiten sind im Datenblatt des QTR-8RC-Reflektor-Sensors von Pololu genannt: In diesem Fall wird der Sensor nach der 11x1-Methode angeschlossen. (Abbildung: Screenshot aus QTR-8RC-Datenblatt)

Wer aus Energiespargründen oder einfach nur so darüber informiert sein möchte, ob die LED-Sensoren nun ein- oder ausgeschaltet sind, kann dies über den LEDON-Pin steuern. Dieser Pin kann leer bleiben (ist in diesem Fall dann intern auf HIGH gesetzt), kann aber auch mit einem GPIO-Pin zwecks Schaltung gekoppelt werden. Liefert diese ein HIGH-Signal, wird die IR-LED-Reihe eingeschaltet. Wird der LEDON-Pin umgekehrt auf LOW gesetzt, bleiben die IR-LEDs aus. Gerade für den Akku-Betrieb dieses Sensor schont die Logik über den LEDON-Pin deutlich die Akku-Kapazität.

QTR-8RC-Sensor mit Raspberry Pi/GertDuino nutzen

Für die Sensoren, die Treiber und das Zubehör stellt Pololu auf der Programmierer-Website Github für diverse Microcontroller Dokumentation und Datenblätter sowie manchmal einen praktischen Beispielcode zum Ausprobieren zur Verfügung – so auch für den QTR-8xEC-Sensor (https://github.com/pololu/qtr-sensors-arduino). Diese Beispiele lassen sich nicht nur für die Sensorreihe mit 8 Sensoren verwenden, sondern auch für die kleineren Modelle, die sich je nach Anzahl der verbauten Sensoren unterscheiden. Hier ist dann schlichtweg nur die Anzahl der Sensoren im Quellcode einzutragen bzw. auch die Anschlussreihenfolge der Pins anzupassen, falls die Digital-Pins auf dem GertDuino-Board anders als im Quellcode dokumentiert bestückt werden. Doch bevor es so weit ist, laden Sie zunächst die Code-Beispiele für den Arduino auf den Raspberry Pi und richten die passende Library für die Sensoren in der Arduino-Software ein.

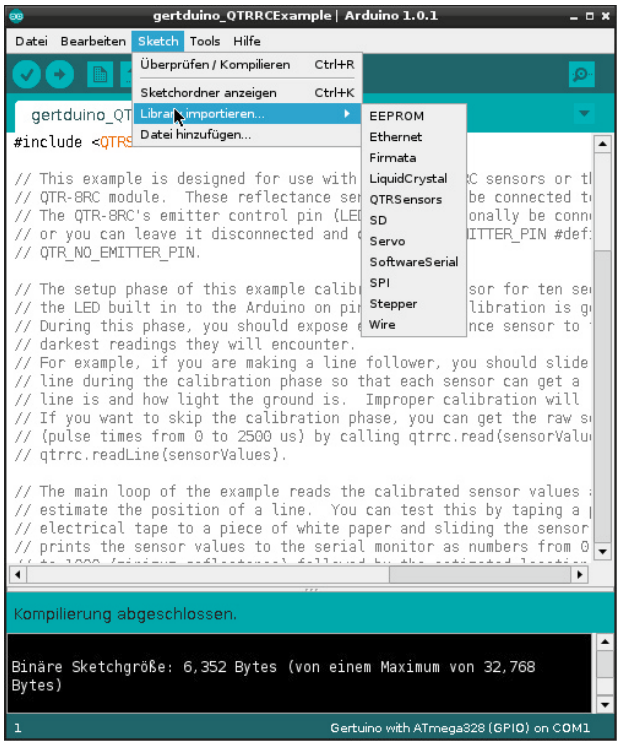
```
mkdir qtr8rc/
cd qtr8rc/
wget https://github.com/pololu/qtr-sensors-arduino/archive/master.zip
mv master.zip qtr-sensors-arduino.zip
unzip qtr-sensors-arduino.zip
cd qtr-sensors-arduino-master/
ls
sudo mv QTRSensors/ /usr/share/arduino/libraries/
```

Ist die Beispiel-Bibliothek von Pololu auf der Speicherkarte geladen, versehen Sie diese zunächst mit einer passenden Bezeichnung, damit Sie später sofort wissen, was sich in der zip-Datei befindet. Nach dem Entpacken der Datei kommen einige Beispiele und Bibliotheken zum Vorschein, die Sie direkt mit dem Arduino bzw. mit dem Raspberry Pi samt GertDuino einsetzen können. Dafür ist es notwendig, das Verzeichnis *QTRSensors* in das Arduino-Verzeichnis */usr/share/arduino/libraries/* zu verschieben.

```
pi@raspiBreakout: ~/qtr8rc
pi@raspiBreakout ~/qtr8rc $ wget https://github.com/pololu/qtr-sensors-arduino/archive/master.zip
--2014-01-22 18:56:22--  https://github.com/pololu/qtr-sensors-arduino/archive/master.zip
Resolving github.com (github.com)... 192.30.252.128
Connecting to github.com (github.com)|192.30.252.128|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/pololu/qtr-sensors-arduino/zip/master [following]
--2014-01-22 18:56:28--  https://codeload.github.com/pololu/qtr-sensors-arduino/zip/master
Resolving codeload.github.com (codeload.github.com)... 192.30.252.147
Connecting to codeload.github.com (codeload.github.com)|192.30.252.147|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: `master.zip'

    [ <=>                                                                   ] 17,997      --.-K/s   in 0.002s

2014-01-22 18:56:34 (7.56 MB/s) - `master.zip' saved [17997]

pi@raspiBreakout ~/qtr8rc $ mv master.zip qtr-sensors-arduino.zip
pi@raspiBreakout ~/qtr8rc $ unzip qtr-sensors-arduino.zip
Archive:  qtr-sensors-arduino.zip
aacbfc5cb0c96d4148069c426ff9215849ab7044
   creating: qtr-sensors-arduino-master/
  inflating: qtr-sensors-arduino-master/LICENSE.txt
   creating: qtr-sensors-arduino-master/QTRSensors/
  inflating: qtr-sensors-arduino-master/QTRSensors/QTRSensors.cpp
  inflating: qtr-sensors-arduino-master/QTRSensors/QTRSensors.h
   creating: qtr-sensors-arduino-master/QTRSensors/examples/
   creating: qtr-sensors-arduino-master/QTRSensors/examples/QTRAExample/
  inflating: qtr-sensors-arduino-master/QTRSensors/examples/QTRAExample/QTRAExample.ino
   creating: qtr-sensors-arduino-master/QTRSensors/examples/QTRARawValuesExample/
  inflating: qtr-sensors-arduino-master/QTRSensors/examples/QTRARawValuesExample/QTRARawValuesExample.ino
   creating: qtr-sensors-arduino-master/QTRSensors/examples/QTRRCExample/
  inflating: qtr-sensors-arduino-master/QTRSensors/examples/QTRRCExample/QTRRCExample.ino
   creating: qtr-sensors-arduino-master/QTRSensors/examples/QTRRCRawValuesExample/
  inflating: qtr-sensors-arduino-master/QTRSensors/examples/QTRRCRawValuesExample/QTRRCRawValuesExample.ino
  inflating: qtr-sensors-arduino-master/QTRSensors/keywords.txt
  inflating: qtr-sensors-arduino-master/README.textile
pi@raspiBreakout ~/qtr8rc $
```

Nach dem Entpacken verschieben Sie mit dem mv-Kommando das QTRSensors-Verzeichnis in die Arduino-Bibliothek, damit der Zugriff über die Arduino-IDE sichergestellt ist.

Für die beiden verfügbaren Infrarot-Sensoren der Baureihe liefert Pololu hier jeweils zwei Beispieldateien mit, die Sie umgehend testen können. Damit lässt sich feststellen, ob zum einen der Sensor auch ordnungsgemäß und korrekt mit dem GertDuino-Board bzw. dem Raspberry Pi verbunden ist und zum anderen die verwendeten Anschlüsse funktionieren. Um nun die Bespielsketche zu nutzen, öffnen Sie über *Datei / File -> Beispiele / Examples* den Eintrag QTRSensors. Befinden sich die Sensor-Bibliotheken am richtigen Platz, können Sie loslegen – ansonsten starten Sie die Arduino-Software neu, um die Library zu initialisieren.

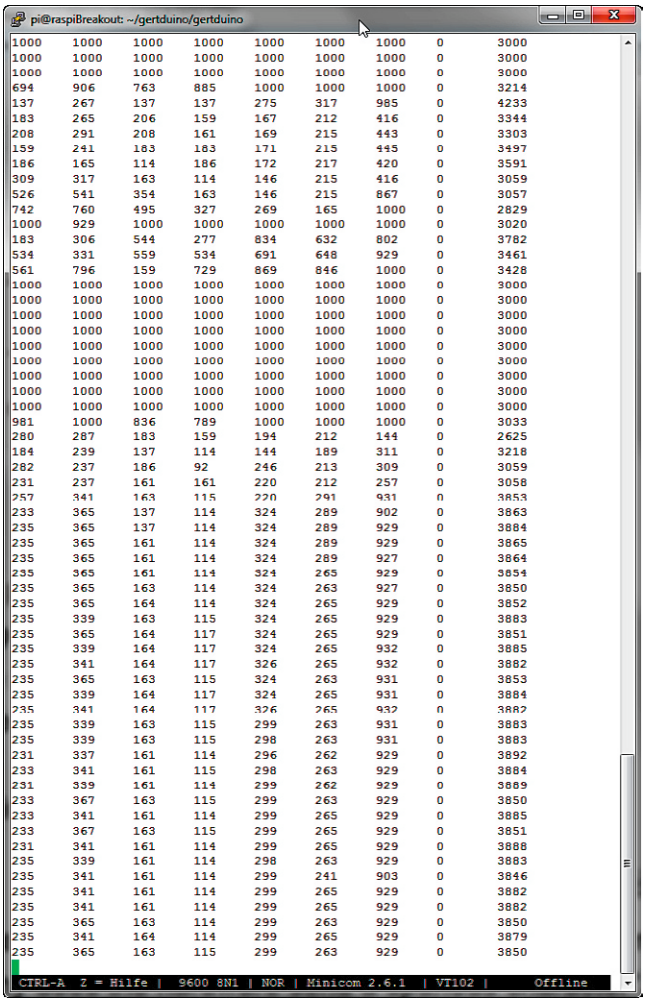


Im ersten Schritt wurde hier die Beispieldatei QTRRCExample geladen. Vorsichtshalber wählen Sie vor der Ausführung über Sketch -> Library importieren -> QTRSensors die frisch installierte Pololu-Library aus.

Falls notwendig, können Sie im Quellcode im Bearbeitungsfenster in der Datei

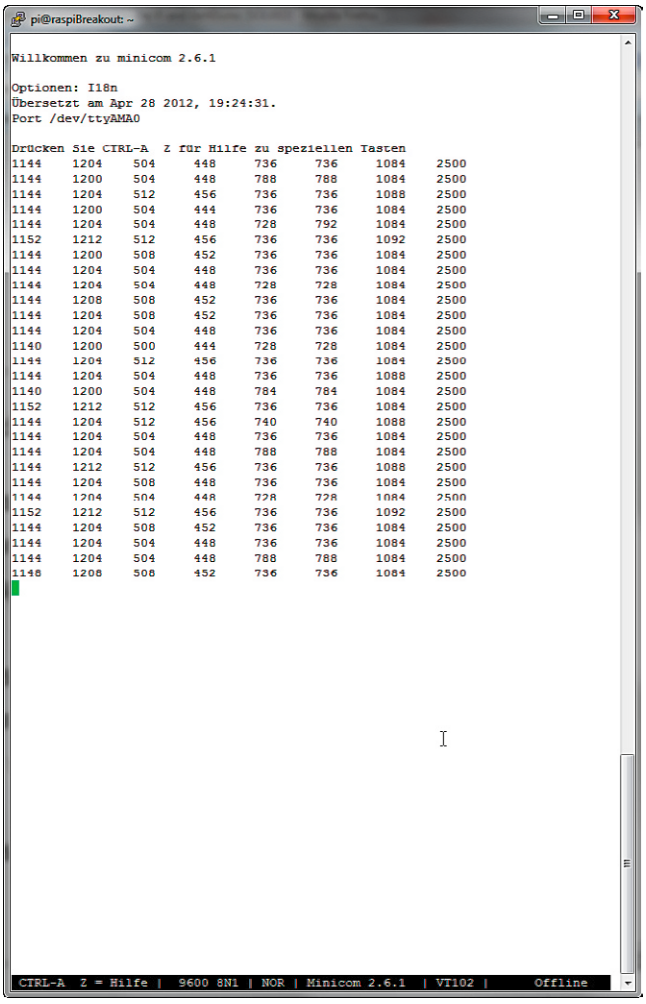
`QTRRCExample`

ggf. noch die Digital-Pins in der Beispieldatei anpassen, falls Sie nicht die vorgeschlagene Anschlussreihenfolge der Sensoren (3, 4, 5, 6, 7, 8, 9, 10) sowie Pin 2 für den LEDON-Schalter verwendet haben. In diesem Fall sichern Sie die Datei unter einer anderen Dateibezeichnung, damit das Original unberührt bleibt. Je nach verwendetem Testbeispiel wirft nun die serielle Konsole permanent eine Reihe von Messwerten aus – der Sensor sollte nun ordnungsgemäß angeschlossen sein.



GertDuino im Sensor-Einsatz: Die von Pololu mitgelieferte Beispieldatei QTRRCExample lässt sich über das GertDuino-Board auf dem Raspberry Pi erfolgreich starten und nutzen.

Im Vergleich dazu können Sie auch den Beispielsketch QTRRCRaw ValuesExample über die Arduino-Software in das GertDuino-Board laden und parallel dazu auf dem Raspberry Pi über minicom auf der seriellen Konsole die Messwerte live mitverfolgen. Der Unterschied zwischen den beiden Sketchen ist, dass im Gegensatz zu den Rohwerten im Beispiel QTRRCRawValues Example im Sketch QTRRCExample bereits die Kalibrierung und eine einfache Linienerkennung implementiert ist.



Beispieldatei QTRRCRawValuesExample auf dem GertDuino/ Raspberry-Pi-Board im Einsatz: Eine Reihe von Daten in einer einheitlichen Reihenfolge wird hier im Rohformat ausgeworfen.

Nun haben Sie technisch auch die Möglichkeit, den angeschlossenen Sensor über das GertDuino-Board zu nutzen, vonseiten des Raspberry Pi sind hier sämtliche Voraussetzungen gegeben. Ein möglicher Anwendungszweck wäre die Kopplung beider Welten – hier können Sie den Infrarot-Sensor über Arduino/GertDuino autark betreiben und nur die Änderungszustände an den Raspberry Pi übermitteln.

QTR-8RC-Sensor am Analog-digital-Wandler MCP3008

Egal, ob analoge oder digitale Verarbeitung – Infrarot-Sensoren scheren sich nicht groß um die dahinterliegende Technik – die gemessene Zustandsänderung sorgt für eine Spannungsänderung, die am einfachsten mit einem günstigen Analog-digital-Wandler gemessen werden kann. In diesem Projekt wurde der Pololu-Sensor QTR-8RC mit dem MCP3008-IC und dem Raspberry Pi verkabelt. Die Grundinstallation des MCP3008-ICs wurde bereits mehrmals beschrieben und wird in diesem Abschnitt vorausgesetzt.

QTR-8RC IR-Pin	QTR-8RC-IR-Funktion	Raspberry-Pi-Pin-Nr.	MCP3008-Kanal	MCP3008-Pin
VCC	5V	2	-	-
GND	Masse	6	-	-
1	Sensor 1	-	0	1
2	Sensor 2	-	1	2
3	Sensor 3	-	2	3
4	Sensor 4	-	3	4
5	Sensor 5	-	4	5
6	Sensor 6	-	5	6
7	Sensor 7	-	6	7
8	Sensor 8	-	7	8
LEDON	Schalter	22 (GPIO25, Wiring Pi 6)	-	-

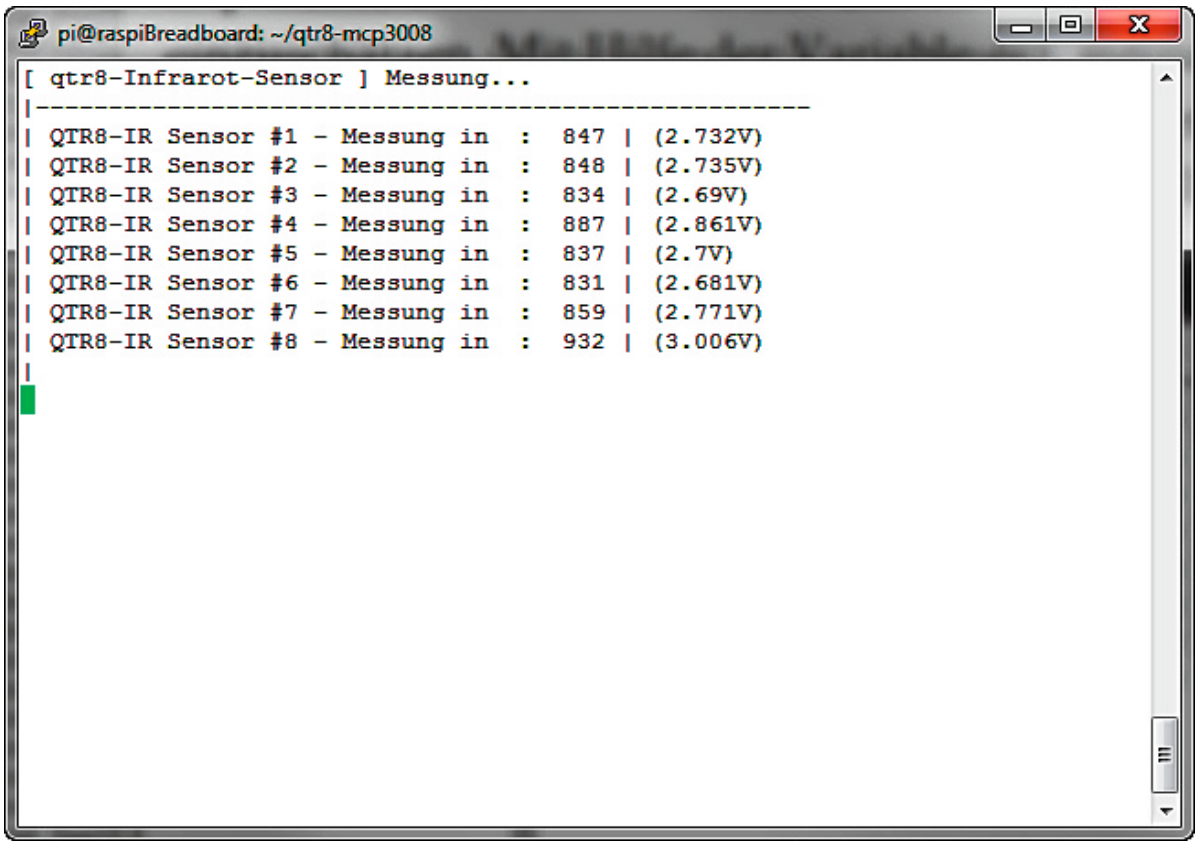
Der abgedruckte Python-Code zum QTR-8RC-IR-Sensor ist vom Aufbau her grundsätzlich recht ähnlich zu den Sensor-Beispielen des MCP3008-ICs. Dieser vergleicht die Eingangsspannung an den jeweiligen Eingängen mit der anliegenden Referenzspannung von 3,3 V und stellt den Messwert als Ganzzahl im Bereich 0–1023 dar. Davon abhängig lässt sich demnach auch die anliegende Spannung ausgeben, die als Gradmesser für die Sensitivität des IR-Sensors dient.

```
# -*- coding: utf-8 -*-
#!/usr/bin/python
#--------------------------------------------------------------------
# Das Skript nutzt die Analog-Eingaenge des MCP3008 IC
# und liest diese ueber SPI Bus aus
# Datei qtr8-step01.py
# --------------------------------------------------------------------
import spidev
# wget https://raw.github.com/doceme/py-spidev/master/setup.py
# wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
import time
import os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)
# SPI Objekt erstellen
spi = spidev.SpiDev()
spi.open(0,0)
# Funktion um Daten aus dem MCP3008 IC zu lesen
# Kanalzaehlung von 0-7
def getadcChannel(channel):
  # Nur Kanal 0-7 erlaubt
  if ((channel > 7) or (channel < 0)):
    return -1
  rdata = spi.xfer2([1,(8+channel)<<4,0])
  data = ((rdata[1]&3) << 8) + rdata[2]
  return data
# Funktion um Werte in Spannungswerte zu konvertieren
# (auf 2^10 Einheiten)
def convertVolts(data,places):
  volts = (data * 3.3) / 1023 # 3.3V Referenzspannung!
  volts = round(volts,places) # und runden
  return volts
```

```
# ------------------------------------------------
# Start Python-Skript
# ------------------------------------------------
os.system('clear')
#print("[ qtr8-Infrarot-Sensor ] Messung...")
# Kanalnummer auswaehlen
qtrChannel1 = 0 # rechts
qtrChannel2 = 1
qtrChannel3 = 2
qtrChannel4 = 3
qtrChannel5 = 4
qtrChannel6 = 5
qtrChannel7 = 6
qtrChannel8 = 7 # links
# ---------->  max 7 !
#
# Pause zwischen Messungen
delay = 2 # Sekunden
# Anzahl der Dezimalstellen
places = 3
try:
# Dauer-Schleife
  anz_messungen = 2
  while True:
    GPIO.output(25,GPIO.HIGH) # LED einschalten
    print("[ qtr8-Infrarot-Sensor ] Messung...")
    # Sensor - 5 messungen
    tmp_level1 = []
    tmp_level2 = []
    tmp_level3 = []
    tmp_level4 = []
    tmp_level4 = []
    tmp_level5 = []
    tmp_level6 = []
    tmp_level7 = []
    tmp_level8 = []
    for i in range (0,anz_messungen):
      tmp_level1.append(getadcChannel(qtrChannel1))
      tmp_level2.append(getadcChannel(qtrChannel2))
      tmp_level3.append(getadcChannel(qtrChannel3))
      tmp_level4.append(getadcChannel(qtrChannel4))
      tmp_level5.append(getadcChannel(qtrChannel5))
      tmp_level6.append(getadcChannel(qtrChannel6))
      tmp_level7.append(getadcChannel(qtrChannel7))
      tmp_level8.append(getadcChannel(qtrChannel8))
    qtrChannel1_level = sum(tmp_level1)/anz_messungen
    qtrChannel2_level = sum(tmp_level2)/anz_messungen
    qtrChannel3_level = sum(tmp_level3)/anz_messungen
    qtrChannel4_level = sum(tmp_level4)/anz_messungen
    qtrChannel5_level = sum(tmp_level5)/anz_messungen
    qtrChannel6_level = sum(tmp_level6)/anz_messungen
    qtrChannel7_level = sum(tmp_level7)/anz_messungen
    qtrChannel8_level = sum(tmp_level8)/anz_messungen
    qtrChannel1_volts = convertVolts(qtrChannel1_level, places)
    qtrChannel2_volts = convertVolts(qtrChannel2_level, places)
    qtrChannel3_volts = convertVolts(qtrChannel3_level, places)
    qtrChannel4_volts = convertVolts(qtrChannel4_level, places)
    qtrChannel5_volts = convertVolts(qtrChannel5_level, places)
    qtrChannel6_volts = convertVolts(qtrChannel6_level, places)
    qtrChannel7_volts = convertVolts(qtrChannel7_level, places)
    qtrChannel8_volts = convertVolts(qtrChannel8_level, places)
    GPIO.output(25,GPIO.LOW) # LED ausschalten
    # Datenausgabe
```

```
        print("|-----------------------------------------------------")
        print("| QTR8-IR Sensor #1 - Messung in  :  {} | ({}V)".format(qtrChannel1_level, qtrChannel1_volts))
        print("| QTR8-IR Sensor #2 - Messung in  :  {} | ({}V)".format(qtrChannel2_level, qtrChannel2_volts))
        print("| QTR8-IR Sensor #3 - Messung in  :  {} | ({}V)".format(qtrChannel3_level, qtrChannel3_volts))
        print("| QTR8-IR Sensor #4 - Messung in  :  {} | ({}V)".format(qtrChannel4_level, qtrChannel4_volts))
        print("| QTR8-IR Sensor #5 - Messung in  :  {} | ({}V)".format(qtrChannel5_level, qtrChannel5_volts))
        print("| QTR8-IR Sensor #6 - Messung in  :  {} | ({}V)".format(qtrChannel6_level, qtrChannel6_volts))
        print("| QTR8-IR Sensor #7 - Messung in  :  {} | ({}V)".format(qtrChannel7_level, qtrChannel7_volts))
        print("| QTR8-IR Sensor #8 - Messung in  :  {} | ({}V)".format(qtrChannel8_level, qtrChannel8_volts))
        print("| ")
        # und warten bis zur naechsten Messung
        time.sleep(delay)
        os.system('clear')
    except KeyboardInterrupt:
      # CTRL-C gedrueckt
      print("[ QTR8-IR Sensor ] Messung abgebrochen.")
    # ---------------------- EOF  --------------------------
```

Für das erste Kennenlernen und die Inbetriebnahme ist der erste Testballon völlig ausreichend. Nach dem Start der Python-Datei wirft das Skript laufend die Messwerte sowie die dazugehörigen Spannungswerte aus. Experimentieren Sie etwas mit den Abständen und der Beleuchtung, um die Sensitivität des Sensors einzuschätzen. Mithilfe der Variable *anz_messungen* (in diesem Beispiel belegt mit dem Wert 2) legen Sie die Anzahl der Messungen fest, die anschließend in einem Mittelwert zusammengefasst werden, um „Ausreißer" bei den Messwerten zu normalisieren.



Je „weißer" der Untergrund, desto höher ist der Messwert, der bis zu 1023 bzw. 3,3 V betragen kann. Erkennt der Sensor einen dunklen Fleck oder eine Linie, dann sinkt der Wert. Dunkelt man den Sensor mit einem schwarzen Untergrund ab, dann wird in diesem Testaufbau ein Wert um die 700 bei 2,3 V erreicht.

Der Umweg über den Analog-digital-Wandler funktioniert, ist jedoch je nach Kalibrierung etwas ungenau. Hier sollte der Sensor möglichst immer im gleichen Abstand zum Objekt verwendet werden. In Sachen Robotik und Linienerkennung (Stichwort: Line Follower Robot) ist es empfehlenswert, den QTR8-IR-Sensor in einem Abstand von ca. 1 cm vom Boden am Fahrzeug anzubringen, um auch bei Tageslicht auswertbare Ergebnisse zu erhalten. ELV