Raspberry Pi Rezepte Teil 6

Das ADC des guten Geschmacks

Von **Tony Dixon** (UK)

Bislang ging es in den Elektor.POST-Projekten hauptsächlich um digitale Signale wie GPIO, serieller UART, SPI und I²C des Expansion Headers. Vollständig ist so ein System aber nur, wenn es etwas Analoges „schmecken“ kann. Thema dieser Folge sind daher ADCs (Analog/Digital-Converter), die via SPI angeschlossen werden können.

Fehlende Analogfunktionen ☹

Leider verblüfft der Expansion Header von RPi durch das vollständige Fehlen analoger Funktionen. Eigentlich ist das eine Schande, denn Boards wie Arduino und BeagleBone Black sind damit großzügig ausgestattet. Doch muss man nicht gleich die Flinte ins Korn bzw. RPi in die Ecke werfen. Dank SPI oder I²C kann man problemlos seriell ansteuerbare ADCs anschließen. Zur Demonstration wird gezeigt, wie man einen MCP3004 (vierkanaliger ADC mit 10 bit Auflösung) per SPI anschließt.

SPI-Interface

Schon in Elektor.POST Projekt Nr. 9 wurde SPI (Serial Peripheral Interface) detailliert besprochen. Hier nun eine kurze Übersicht:

Tabelle 1 listet die Signale des Expansion Headers auf. SPI liegt auf Pin 19 (MOSI), Pin 21 (MISO) und Pin 23 (SCK). Die beiden SPI-Chip-Enable-Signale sind Pin 24 (CE0) und Pin 26 (CE1).
Im Kasten *„Installation von Pythons SPI-Library“* steckt eine Anleitung, wie RPi für die Nutzung von SPI konfiguriert wird.

ADC-Hardware

In diesem ersten analogen RPi-Projekt kommt der Wandler MCP3004 von Microchip [1] zum Einsatz, der vier Kanäle bei 10 bit Auflösung bietet.
Bild 1 zeigt die einfache Beschaltung des MCP3004. Ausgangsseitig ist das IC direkt mit dem SPI-Interface von RPi verbunden. Per Jumper kann zwischen den SPI-CE-Sig-

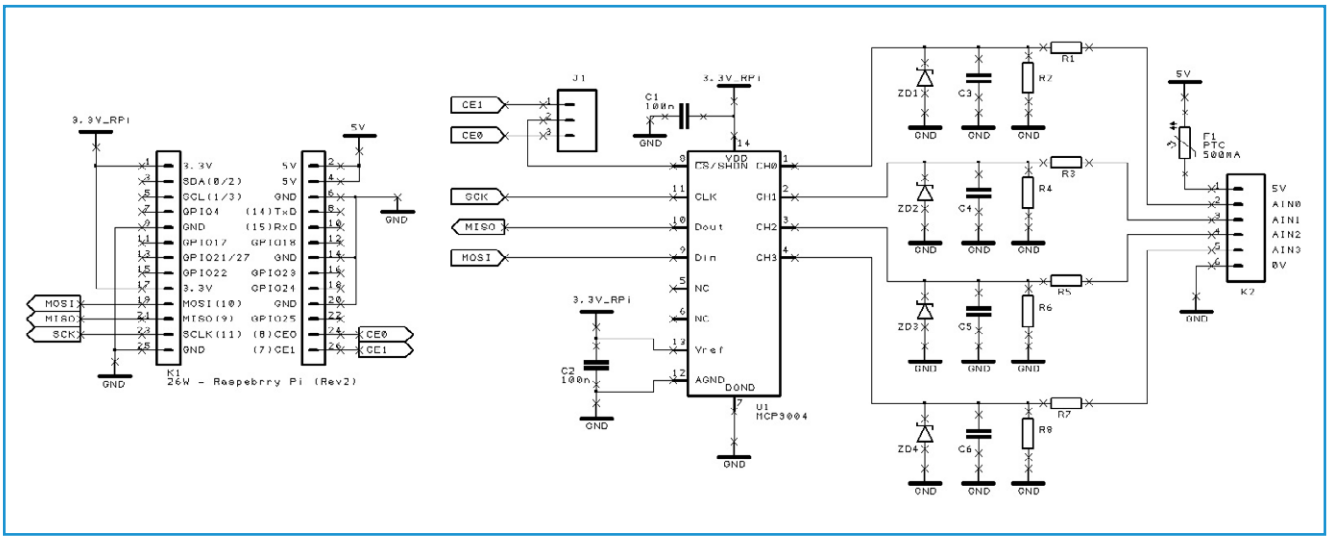

Bild 1. Schaltung der ADC-Erweiterung mit MCP3004 für Raspberry Pi.

nalen CE0 oder CE1 umgeschaltet werden. Die Schaltung ist so einfach, dass man sie gut auf einer Lochrasterplatine aufbauen kann.

Ein MCP3004 benötigt 3,3 V. Man darf also keine zu großen Spannungen anschließen. Alle vier Eingänge sind mit Spannungsteilern versehen. Bei Kanal 1 sind dies R1 und R2. Das Eingangssignal wird daher folgendermaßen abgeschwächt:

$$V_{out} = \frac{R2}{R1 + R2} \times V_{in}$$

- Für einen Messbereich von 0...5 V wird R1 = R2 auf 10 kΩ gesetzt. Das ergibt dann eine Spannung am ADC-Eingang von 0...2,5 V.
- Für einen Messbereich von 0...10 V gilt: R1 = 10 kΩ und R2 = 22 kΩ, was am ADC-Eingang 0...3,125 V ergibt.
- Für einen Messbereich von 0...3,3 V schließlich wird R1 = 1 kΩ und R2 nicht bestückt. Am ADC liegt daher die Eingangsspannung von 0..3,3 V an.

Die Spannungsteiler für die Kanäle 2 bis 4 können nach Bedarf gleich oder auch für unterschiedliche Eingangsspannung bestückt werden. Um Signalstörungen zu reduzieren, wird noch je ein kleiner Kondensator von 1...10 nF für C3...C6 bestückt. Wer auf Nummer Sicher gehen will, kann für ZD1...ZD4 noch Z-Dioden mit einer Nennspannung von 3,6 V als Überspannungsschutz bestücken.

Tabelle 1. Pin-Belegung des Expansion Headers

Pin-Name	Pin-Funktion	Alternative	RPi.GPIO
P1-02	5,0V	-	-
P1-04	5,0V	-	-
P1-06	GND	-	-
P1-08	GPIO14	UART0_TXD	RPi.GPIO8
P1-10	GPIO15	UART0_RXD	RPi.GPIO10
P1-12	GPIO18	PWM0	RPi.GPIO12
P1-14	GND	-	-
P1-16	GPIO23		RPi.GPIO16
P1-18	GPIO24		RPi.GPIO18
P1-20	GND	-	-
P1-22	GPIO25		RPi.GPIO22
P1-24	GPIO8	**SPI0_CE0_N**	RPi.GPIO24
P1-26	GPIO7	**SPI0_CE1_N**	RPi.GPIO26

Pin-Name	Board Revision 1		Board Revision 2	
	Pin-Funktion	Alternative	Pin-Funktion	Alternative
P1-01	3,3V	-	3,3V	-
P1-03	GPIO0	I2C0_SDA	GPIO2	I2C1_SDA
P1-05	GPIO1	I2C0_SCL	GPIO3	I2C1_SCL
P1-07	GPIO4	GPCLK0	GPIO4	GPCLK0
P1-09	GND	-	GND	-
P1-11	GPIO17	RTS0	GPIO17	RTS0
P1-13	GPIO21		GPIO27	
P1-15	GPIO22		GPIO22	
P1-17	3,3V	-	3,3V	-
P1-19	GPIO10	**SPI0_MOSI**	GPIO10	**SPI0_MOSI**
P1-21	GPIO9	**SPI0_MISO**	GPIO9	**SPI0_MISO**
P1-23	GPIO11	**SPI0_SCLK**	GPIO11	**SPI0_SCLK**
P1-25	GND	-	GND	-

Achtung: I2C0_SDA und I2C0_SCL (GPIO0 & GPIO1) sowie I2C1_SDA und I2C1_SCL (GPIO2 & GPIO3) haben 1,8-kΩ-Pull-up-Widerstände gegen 3,3 V.

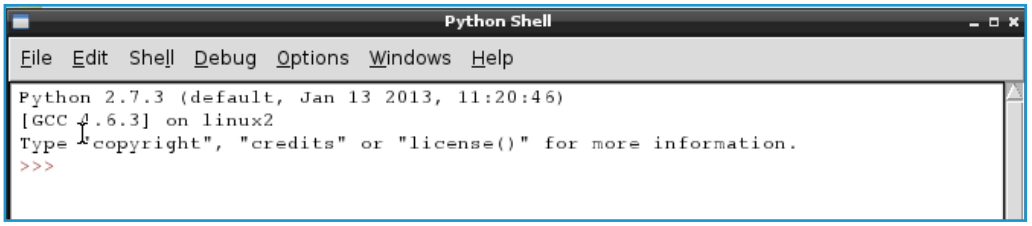

Bild 2.
IDLE-Python-Shell.

Beispielprogramm mcp3004.py

Nachdem die Schaltung aufgebaut und spidev installiert ist (siehe Kasten „Installation von Pythons SPI-Library“), schreibt man ein kleines Testprogramm, um Spannungen an Kanal 1 zu messen und anzuzeigen.

Ein Doppelklick auf das IDLE-Icon auf dem Desktop von RPi startet die Python-Shell und die IDE (**Bild 2**). Mit der File-Option des Menüs erzeugt man ein neues

Installation von Pythons SPI-Library

Pythons SPI-Library [2] wurde schon in Elektor.POST Projekt Nr. 9 installiert. Falls Sie das noch nicht erledigt haben, gibt es hier noch einmal eine Anleitung. Zunächst startet man eine LXTerminal-Session (**Bild 4**) und tippt folgende Befehle ein:

```
sudo apt-get install git-core

cd ~
git clone git://github.com/doceme/py-spidev
cd py-spidev/
sudo python setup.py install
```

Alternativ dazu kann man auch den Python Package Installer „Pip“ verwenden:

```
sudo apt-get install git-core python-dev
sudo apt-get install python-pip
sudo pip install spidev
```

Die SPI-Hardware ist normalerweise deaktiviert. Um dies zu ändern muss man die Blacklist-Datei editieren:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Nun sucht man die Zeile mit ***blacklist spi-bcm2708*** und fügt am Anfang dieser Zeile ein # (Doppelkreuz) ein, um den folgenden Befehl auszukommentieren. Nach dem Sichern der geänderten Datei kommt ein Reboot via:

```
sudo reboot
```

Nach dem Booten startet man eine neue LXTerminal-Session und tippt:

```
ls /dev/spi*
```

Damit kann man prüfen, ob zwei SPI-Geräte gelistet werden (je eines pro CE-Signal). Es sollte sich also ergeben:

```
/dev/spidev0.0
/dev/spidev0.1
```

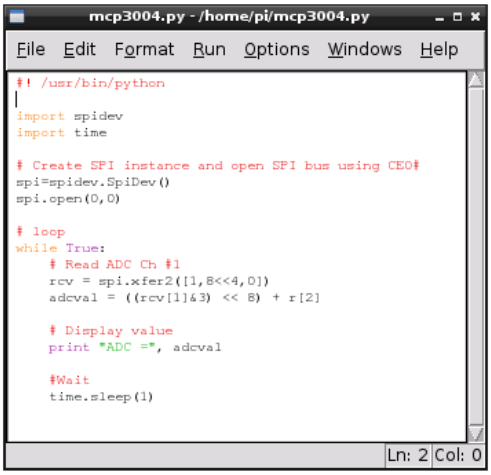

Bild 3. IDLE-Editor.

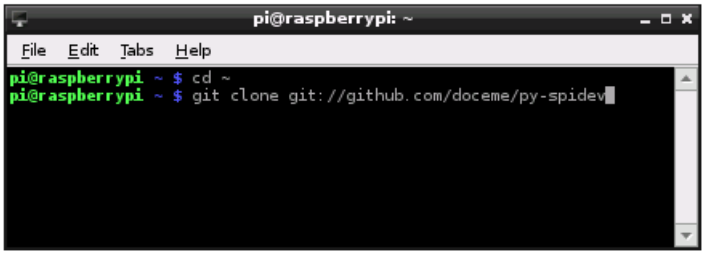

Bild 4. LXTerminal.

Programm. Dies startet außerdem den IDLE-Editor (**Bild 3**), mit dem man das Programm von **Listing 1** eingibt.
Nach der Eingabe des Programms sichert man es und wechselt zu LXTerminal. Dort tippt man den folgenden Befehl ein, um das Programm ausführbar zu machen:

```
chmod +x mcp3004.py
```

Anschließend kann man das Programm mit dem folgenden Befehl ausführen:

```
sudo ./mcp3004.py
```

(130260)

Weblinks

[1] http://goo.gl/eMSOQ

[2] https://github.com/doceme/py-spidev

Listing 1.

```
#! /usr/bin/python

import spidev
import time

# Create SPI instance and open SPI bus using CE0
spi = spidev.SpiDev()
spi.open(0,0)

# Loop
while True:

    # Read ADC Ch #1
    rcv = spi.xfer2 ([1,8<<4, 0])
    adcval = ((rcv [1]&3) << 8) + rcv[2]

    # Display value
    Print “ADC = “, adcval

    # Wait
    time.sleep(1)
```

Kurze Liste der spidev-Befehle	
spi.open (0,0)	Öffnet den SPI-Bus 0 mit CE0.
spi.open (0,1)	Öffnet den SPI-Bus 0 mit CE1.
spi.close ()	Trennt die Peripherie vom Interface.
spi.writebytes ([array of bytes])	Schickt ein Array mit Bytes zum SPI-Gerät.
spi.readbytes (len)	Liest len Bytes vom SPI-Gerät.
spi.xfer2 ([array of bytes])	Schickt ein Array mit Bytes und behält währenddessen CEx bei.
spi.xfer ([array of bytes])	Schickt ein Array mit Bytes und prüft CEx mit jedem gesendeten Byte.