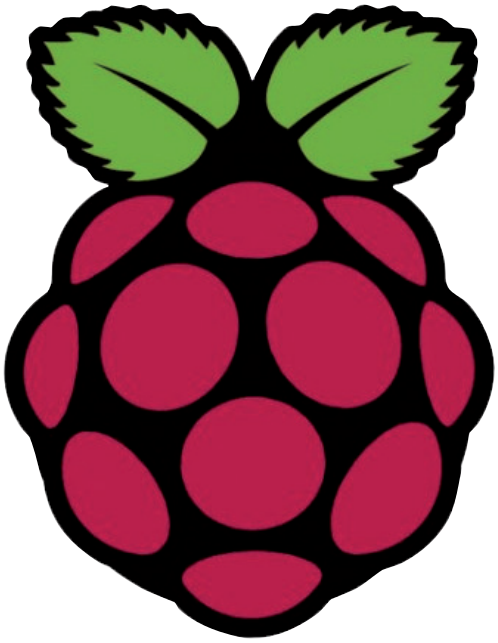
RPi-Rezepte Teil 4

RPi alias Raspberry (S)PI



Von **Tony Dixon** (UK)

Im letzten Teil ging es um die seriellen Schnittstellen des Expansion Headers von Raspberry Pi. Nun ist ein weiteres serielles Interface an der Reihe: der SPI-Bus.

SPI

Beim SPI (**S**erial **P**eripheral **I**nterface) handelt es sich um die zweite der drei seriellen Schnittstellen, die auf den Expansion Header von RPi herausgeführt sind. Die beiden anderen Interfaces sind die UART-Pins (siehe Teil 3) und die I²C-Schnittstelle.

Tabelle 1 zeigt die Signale des Expansion Headers. Die SPI-Signale sind MOSI (19), MISO (21) und SCK (23). Als Chip-Enable-Signale fungieren CE0 (24) und CE1 (26). SPI ist ein lockerer serieller Standard zum Anschluss von digitalen Schaltungen mit wenigen Leitungen nach dem Master/Slave-Prinzip. Beim folgenden Beispiel ist RPi der SPI-Master. Normalerweise braucht es für SPI vier physikalische Leitungen für korrekte Funktion: MOSI (**M**aster **O**ut, **S**lave **I**n), MISO (**M**aster **I**n, **S**lave **O**ut, SCLK (**S**erial **CL**oc**K**) und CE (**C**hip **E**nable, oft auch Chip Select oder Slave Select genannt).

Port-Expander-Hardware

In dem RPi-SPI-Projekt dieser Folge wird die Anzahl an I/O-Pins durch einen Port Expander erhöht. Die Arbeit erledigt das IC MCP23S17, ein Port-Expander-Chip mit 16 Kanälen von Microchip.

Bild 1 zeigt die simple MCP23S17-Beschaltung. Das IC wird per SPI an RPi angeschlossen, wobei man zwischen den beiden CE-Signalen CE0 oder CE1 per Jumper wählen kann.

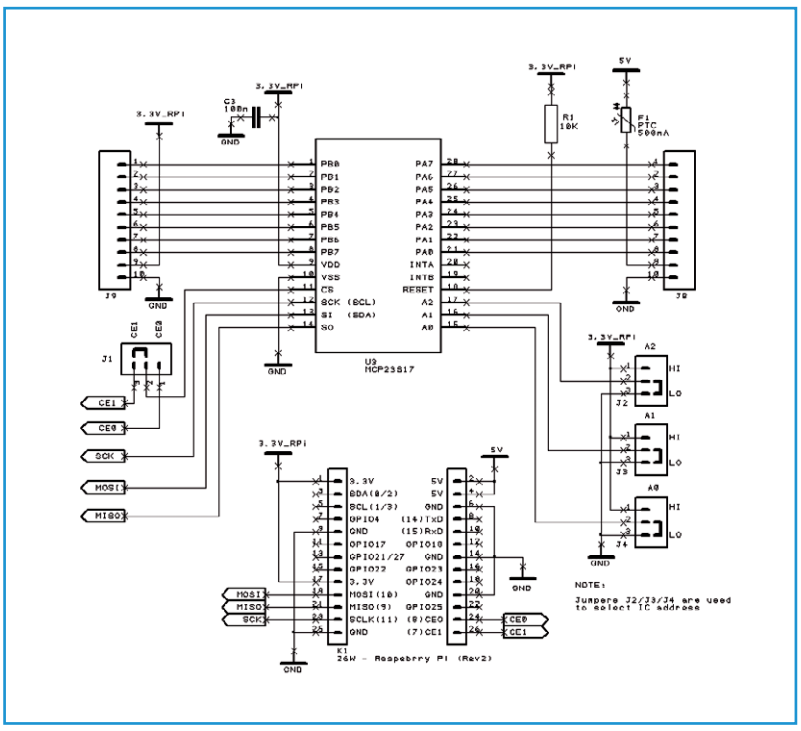


Bild 1.
Schaltung des RPi-Port-Expanders mit MCP23S17.

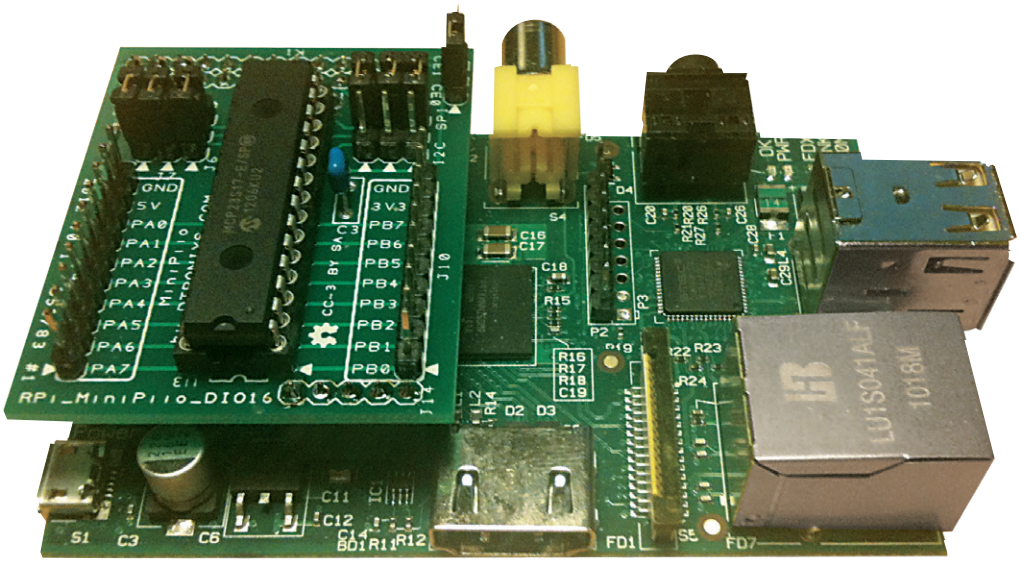


Bild 2.
RPi und MCP23S17-Zusatzplatine.

Bild 2 zeigt den Hardware-Aufbau. Auf der kleinen Zusatzplatine [3] stehen dann die zusätzlichen I/O-Signale an Stiftleisten zur Verfügung.

Installation der Python-SPI-Library

Bei der Software zu diesem Projekt setzen wir auf Python 2. Auch wenn Python schon standardmäßig in der Raspbian-Distribution installiert ist, wird SPI leider noch nicht unterstützt. Zwecks Nachrüstung muss man also den passenden Wrapper bzw. die SPI-Library für Python installieren. Hierzu startet man eine LXterminal-Session (siehe **Bild 3**) und gibt die folgenden Kommandos ein:

```
cd ~

git clone git://github.com/doceme/
   py-spidev

cd py-spidev/

sudo python setup.py install
```

oder

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

Nach der Installation muss man noch ein bisschen aufräumen und Raspbian beibringen, dass die SPI-Hardware verwendet werden soll. Die SPI-Hardware ist nämlich normalerweise deaktiviert. Um dies zu ändern, muss man die Blacklist-Datei editieren:

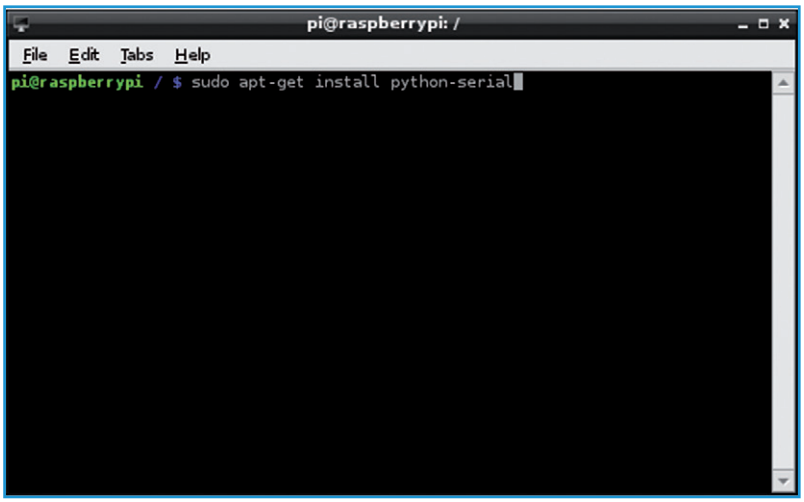
```
sudo nano /etc/modprobe.d/raspi-
   blacklist.conf
```

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:

Bild 3.
LXTerminal.



```
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
```

Beispiel: mcp23s17.py

Mit installiertem „spidev“ kann man nun ein kleines Testprogramm schreiben, um an den Portexpander angeschlossene LEDs leuchten zu lassen.

Hierzu doppelklickt man das IDLE-Icon auf dem Desktop von RPi, um die Python-Shell und die IDE zu starten (**Bild 4**).

Mit dem Menüeintrag „File“ erstellt man ein neues Programm. Dadurch wird der Editor der IDE gestartet. Im IDLE-Editor (**Bild 5**) gibt man nun das Programm aus dem **Listing** ein. Nach Eingabe sollte man die Datei sichern und dann zu einer LXTerminal-Session wechseln, um das erstellte Programm mit dem folgenden Befehl ausführbar zu machen:

```
chmod +x mcp23s17.py
```

Nach getaner Arbeit kann das Programm so gestartet werden:

```
sudo ./mcp23s17.py
```

(130211)

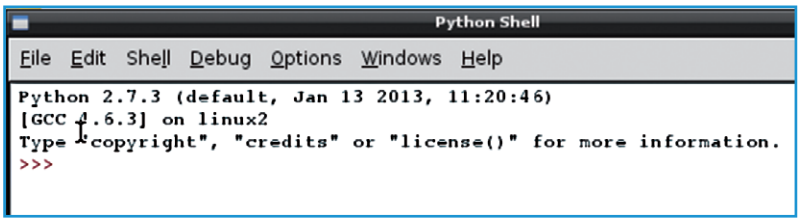


Bild 4. IDLE Python Shell.

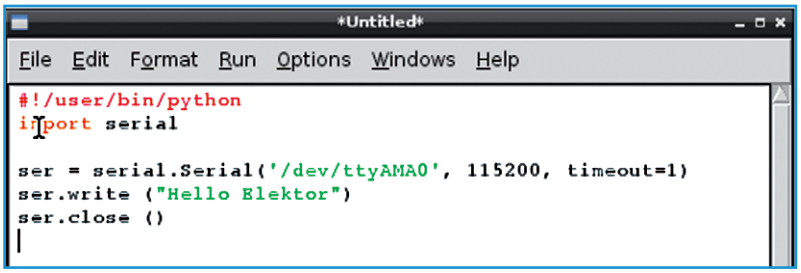


Bild 5. IDLE Editor.

Listing

```
#!/usr/bin/python

import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)

while True:
    spi.xfer([0,0,0])     # turn all lights off
    time.sleep(1)
    spi.xfer([1,255,254]) # turn all lights on
    time.sleep(1)
```

LISTE der Spidev-Befehle	
`spi.open (0,0)`	Aktiviert den SPI-Bus 0 mit CE0.
`spi.open (0,1)`	Aktiviert den SPI-Bus 0 mit CE1.
`spi.close ()`	Deaktiviert den SPI-Bus.
`spi.writebytes ([array of bytes])`	Sendet ein Byte-Array zum SPI-Slave.
`spi.readbytes (len)`	Liest *len* Bytes vom SPI-Slave.
`spi.xfer2 ([array of bytes])`	Sendet ein Byte-Array, dabei bleibt CEx dauerhaft aktiv.
`spi.xfer ([array of bytes])`	Sendet ein Byte-Array, CEx wird vor jedem Byte aktiv und dann wieder inaktiv.

Weblinks

[1] www.raspberrypi.org

[2] www.github.com/doceme/py-spidev

[3] www.dtronixs.com

Tabelle 1. Pinbelegung des Expansion Headers.

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	Board-Revision 1		Board-Revision 2	
	Funktion	Alternative	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	-

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