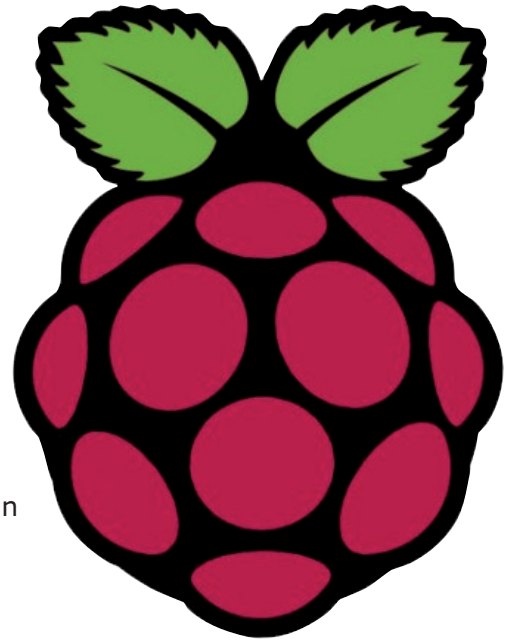
Raspberry Pi Recepten (deel 4)

Raspberry (S)PI



Tony Dixon
(Verenigd Koninkrijk)

De vorige keer hebben we gekeken naar de seriële UART-interface op de uitbreidingsconnector van de Raspberry Pi. In dit project kijken we naar één van de andere seriële interfaces van de Raspberry Pi: de SPI-bus.

SPI-interface

De Serial Peripheral Interface (SPI) is de tweede van de drie seriële interfaces op de uitbreidingsconnector van de Raspberry Pi. De andere twee zijn de UART-interface (zie deel 3) en de I²C-interface.

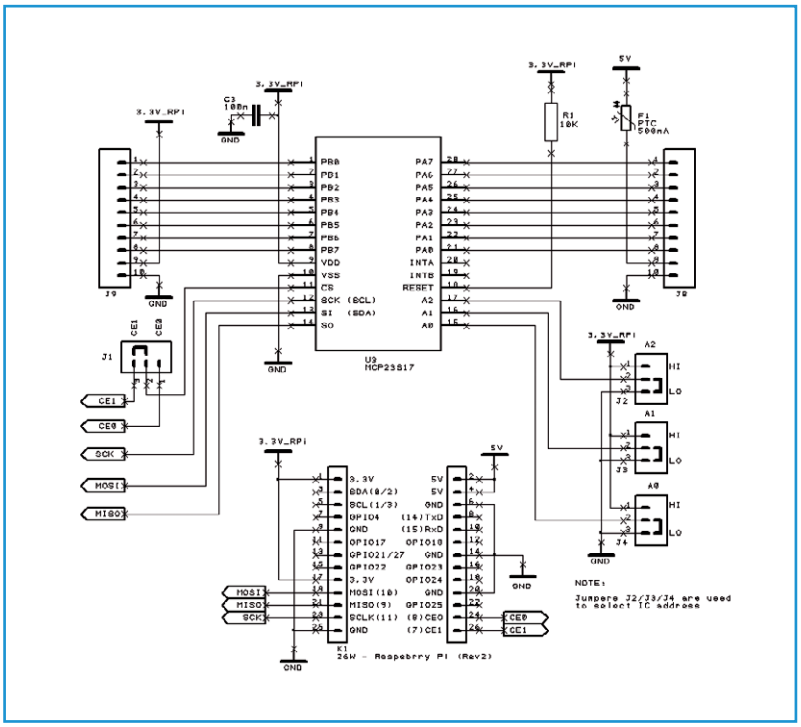
In **tabel 1** zien we een overzicht van de signalen op de uitbreidingsconnector. De SPI-interface is te vinden op pen 19 (MOSI), pen 21 (MISO), pen 23 (SCK). Er zijn twee CE-lijnen voor de SPI-interface: pen 24 (CE0) en pen 26 (CE1).

SPI is ontworpen om met andere apparaten te communiceren in een master/slave-configuratie met een minimaal aantal aansluitpennen. In onze voorbeelden werkt de Raspberry Pi altijd als SPI-master. Een SPI-interface heeft normaal gesproken 4 signaallijnen nodig om te kunnen werken. De signalen zijn *Master Out, Slave In* (MOSI), *Master In, Slave Out* (MISO), *Serial Clock* (SCK) en *Chip Enable* (CEx).

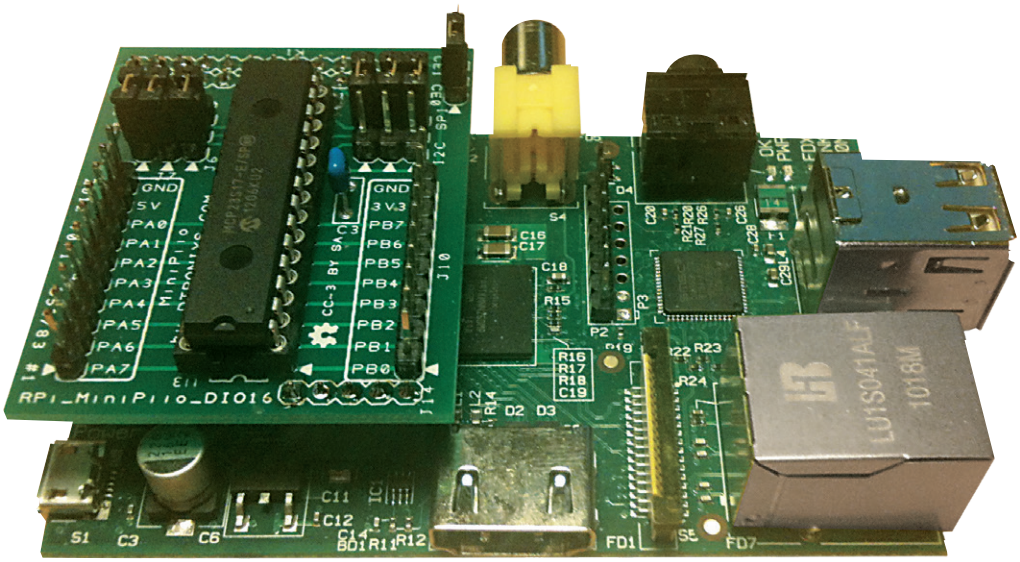
Port-expander hardware

In ons SPI-project voor de Raspberry Pi breiden we het aantal GPIO-lijnen uit door gebruik te maken van een port-expander. We maken gebruik van een 16-kanaals port-expander van het type MCP23S17 van Microchip.

In **figuur 1** zien we een eenvoudig schema ter illustratie van het gebruik van de MCP23S17. De chip is verbonden met de SPI-interface van de RPi. Met een jumper kunnen we één



Figuur 1.
Schema voor de MCP23S17 Raspberry Pi Port-Expander



Figuur 2.
Pi met MCP23S17-opsteekprint.

van de twee CE-lijnen van de SPI kiezen (CE0 of CE1).

Figuur 2 geeft onze hardware weer. Met behulp van een kleine extra kaart [3] verbinden we de interface met de MC23S17.

Installeren van de SPI-library voor Python

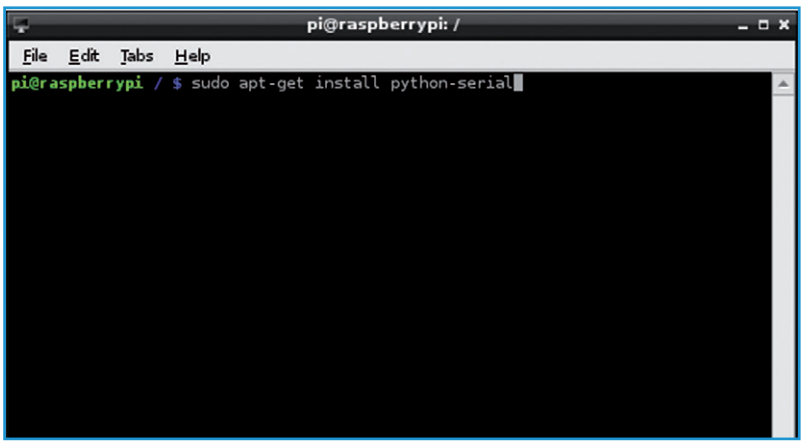
We hebben voor onze voorbeeldprojecten gekozen om Python 2 te gebruiken als programmeertaal. Vorige keer hebben we gezien dat Python al standaard is geïnstalleerd in de Raspbian-distributie.
Maar er is geen voorziening voor de SPI-interface. Om dat op te lossen moeten we de SPI Python wrapper/library installeren. Start dus een LXTerminal-sessie (zie **figuur 3**) en type de volgende commando's:

```
cd ~

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

cd py-spidev/

sudo python setup.py install

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

Na het installeren moeten we wat huishoudelijke maatregelen treffen om Raspbian duidelijk te maken, dat we de SPI-interface willen gebruiken. Standaard is de hardware-SPI uitgeschakeld, dus we moeten dat veranderen door de blacklist-file aan te passen:

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

Zoek de regel met ***blacklist spi-bcm2708*** en plaats een # (hekje) aan het begin van de regel. Daarmee verandert de regel in een commentaarregel. Sla daarna het bestand op. Na het opslaan moeten we het systeem opnieuw booten met:



Figuur 3.
LXTerminal.

```
sudo reboot
```

Start na de reboot opnieuw een LXTerminal-sessie en type...

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

...om te controleren of we nu twee SPI-apparaten hebben (één voor elk SPI Chip Select-signaal). Dat moet er als volgt uit zien:

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

Voorbeeldprogramma: mcp23s17.py

Nu spidev is geïnstalleerd kunnen we een klein testprogramma schrijven om LED's aan te sturen via de port-expander GPIO.

Dubbelklik het pictogram IDLE op het bureaublad van de Pi om de Python Shell en IDE te starten (zie **figuur 4**).

Kies nu de optie File in het menu en maak een nieuw programma. Dit start de IDE-editor. Voer nu met de IDLE-editor het programma in de **listing** in (zie **figuur 5**).

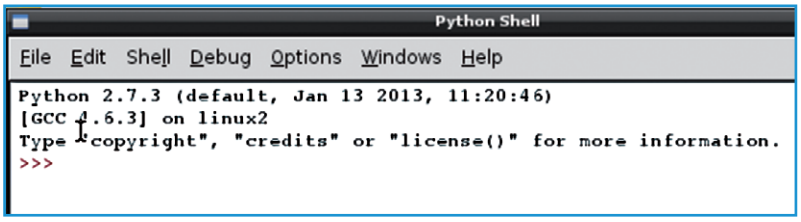
Sla het programma na het intypen op en schakel over naar een LXTerminal. Geef dan het volgende commando om het programma uitvoerbaar te maken:

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

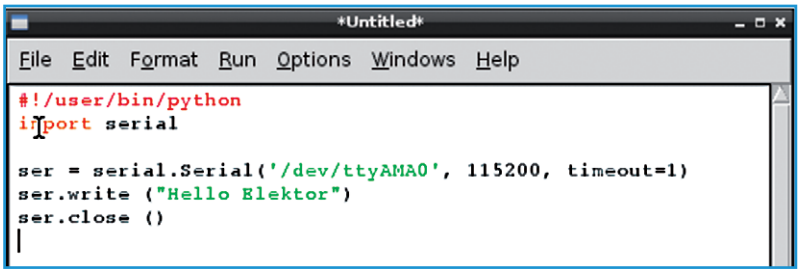
Nu kan het programma worden uitgevoerd met het commando:

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

(130211)



Figuur 4. IDLE Python-Shell.



Figuur 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)
```

Commando's van spidev	
`spi.open (0,0)`	Opent SPI-bus 0 met CE0.
`spi.open (0,1)`	Opent SPI-bus 0 met CE1.
`spi.close ()`	Verbreekt de verbinding met de interface.
`spi.writebytes ([array of bytes])`	Schrijft een array van bytes naar het SPI-device.
`spi.readbytes (len)`	Leest *len* bytes van het SPI-device.
`spi.xfer2 ([array of bytes])`	Stuurt een array van bytes, waarbij CEx continu actief gehouden wordt.
`spi.xfer ([array of bytes])`	Stuurt een array van bytes, waarbij CEx telkens tussen twee bytes gedeactiveerd wordt.

Weblinks

[1] www.raspberrypi.org

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

[3] www.dtronixs.com

Tabel 1: Pinbezetting van de uitbreidingsconnector

Naam	Functie	Alternatief	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

Naam	Board Revision 1		Board Revision 2	
	Functie	Alternatief	Functie	Alternatief
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	-

Opmerking: I2C0_SDA, I2C0_SCL (GPIO0 & GPIO1), I2C1_SDA en I2C1_SCL (GPIO2 & GPIO3) hebben pullup-weerstanden van 1,8 kΩ naar 3V3.