

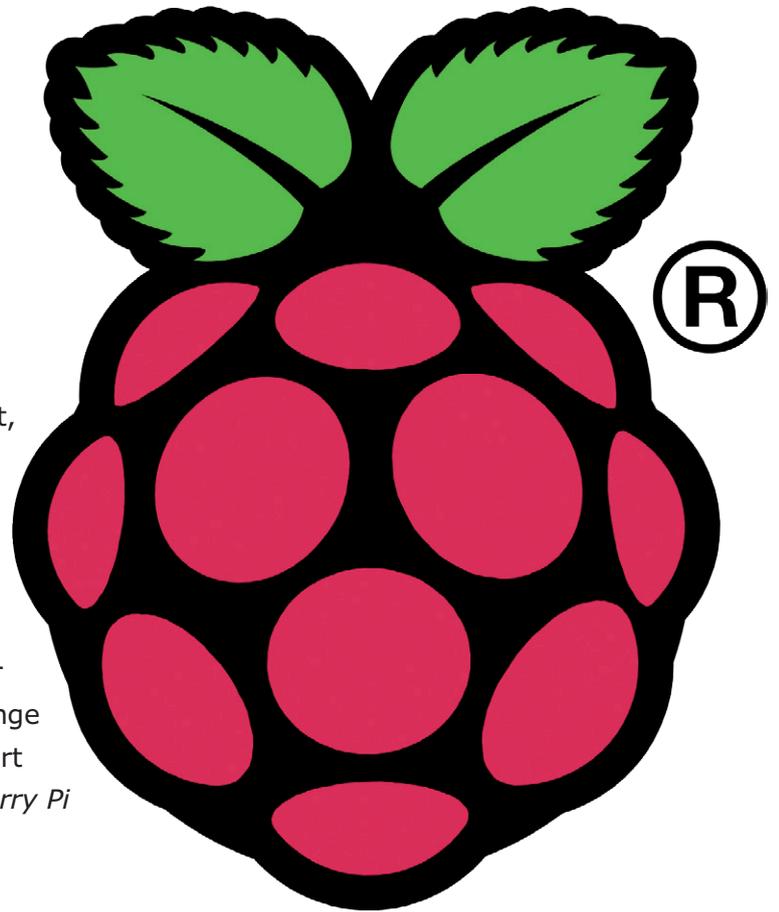
Raspberry-Pi-Rezepte

Teil 2

Die Messer wetzen und bereit legen...

Von Tony Dixon (UK)

Im ersten .POST-Projekt zu RPi wurde gezeigt, wie Raspbian installiert und RPi startbereit gemacht wird. Letztes Mal wurde versprochen, dass es sich in dieser Folge um den „Expansion Header“ drehen wird und darum, wie man seine GPIO-Pins programmiert. Wenn Sie die März-Ausgabe 2013 von Elektor gelesen haben, dann werden Ihnen einige Dinge bekannt vorkommen, da der Erweiterungs-Port auch im Artikel *Prototyping-Board für Raspberry Pi* [1] behandelt wurde.



Expansion Header: Kontakt mit der Welt

Für hardware-orientierte Elektroniker ist der Expansion Header sicherlich das Interessanteste an RPi – direkt nach dem unglaublich niedrigen Preis für das Board. Der Header befindet sich in der Ecke neben der Buchse für Composite-Video. Es handelt sich um eine 2x13-polige Stiftleiste im selbstbaufreundlichen Rastermaß von 1/10“.

An den 26 Pins liegen drei Arten von Signalen an:

- Power: +5 V und 3,3 V sowie Masse (Hinweis: Die 3,3-V-Quelle liefert nur etwa 50 mA)
- Ein-/Ausgang: General Purpose Input/Output (GPIO)
- Kommunikation: Serieller UART, SPI und I²C

Der Expansion Header P1 liefert insgesamt 17 GPIO-Signale. Die meisten dieser Pins haben

eine alternative Funktion und bieten so wahlweise auch Verbindung via UART, SPI und I²C — siehe **Tabelle 1**.

Jeder GPIO-Anschluss kann je nach Konfiguration seiner Treiberleistung zwischen 2 und 16 mA liefern. Die Konfiguration dieser „drive strength“ wird über ein spezielles Register erledigt. Nach einem Reset ist hier als Standard ein Strom von 8 mA eingestellt.

Bei der Revision 2 von RPi wurde mit P5 noch ein zweiter, kleinerer Expansion Header vorgesehen — siehe **Tabelle 2**. Damit kommen noch vier weitere GPIO-Pins hinzu. Wichtiger aber sind die Audio-Signale, da man so einen Anschluss an das PCM-Audio-Interface des Chips 2835 von Broadcom erhält.

Dagegen wurde P1 bei der RPi-Revision 2 nur leicht modifiziert: Statt der Signale von I2C0 stehen nun die des I2C1-Interface zur Ver-

Tabelle 1. Expansion Header Pin-Bezeichnungen

| Pin-Name | Pin-Funktion | Alternative | RPi.GPIO | Board Versie 1 | | Board Versie 2 | | |
|----------|--------------|-------------|------------|----------------|-------------|----------------|-------------|-----------|
| | | | | Funktion | Alternative | Funktion | Alternative | |
| P1-02 | 5,0 V | - | - | P1-01 | 3,3 V | - | 3,3 V | - |
| P1-04 | 5,0 V | - | - | P1-03 | GPIO0 | I2C0_SDA | GPIO2 | I2C1_SDA |
| P1-06 | GND | - | - | P1-05 | GPIO1 | I2C0_SCL | GPIO3 | I2C1_SCL |
| P1-08 | GPIO14 | UART0_TXD | RPi.GPIO8 | P1-07 | GPIO4 | GPCLK0 | GPIO4 | GPCLK0 |
| P1-10 | GPIO15 | UART0_RXD | RPi.GPIO10 | P1-09 | GND | - | GND | - |
| P1-12 | GPIO18 | PWM0 | RPi.GPIO12 | P1-11 | GPIO17 | RTS0 | GPIO17 | RTS0 |
| P1-14 | GND | - | - | P1-13 | GPIO21 | - | GPIO27 | - |
| P1-16 | GPIO23 | - | RPi.GPIO16 | P1-15 | GPIO22 | - | GPIO22 | - |
| P1-18 | GPIO24 | - | RPi.GPIO18 | P1-17 | 3,3 V | - | 3,3 V | - |
| P1-20 | GND | - | - | P1-19 | GPIO10 | SPI0_MOSI | GPIO10 | SPI0_MOSI |
| P1-22 | GPIO25 | - | RPi.GPIO22 | P1-21 | GPIO9 | SPI0_MISO | GPIO9 | SPI0_MISO |
| P1-24 | GPIO8 | SPI0_CE0_N | RPi.GPIO24 | P1-23 | GPIO11 | SPI0_SCLK | GPIO11 | SPI0_SCLK |
| P1-26 | GPIO7 | SPI0_CE1_N | RPi.GPIO26 | P1-25 | GND | - | GND | - |

Hinweis: I2C0_SDA und I2C0_SCL (GPIO0 & GPIO1) sowie I2C1_SDA und I2C1_SCL (GPIO2 & GPIO3) sind mit Pull-up-Widerständen von 1,8 kΩ gegen die Versorgung (3,3 V) beschaltet.

Tabelle 2. P5-Pinbelegung

| Pin-Name | Funktion | Alternative |
|----------|----------|-------------|
| P5-01 | 5,0 V | |
| P5-02 | 3,3 V | |
| P5-03 | GPIO28 | PCM_CLK |
| P5-04 | GPIO29 | PCM_FS |
| P5-05 | GPIO30 | PCM_DIN |
| P5-06 | GPIO31 | PCM_DOUT |
| P5-07 | GND | |
| P5-08 | GND | |

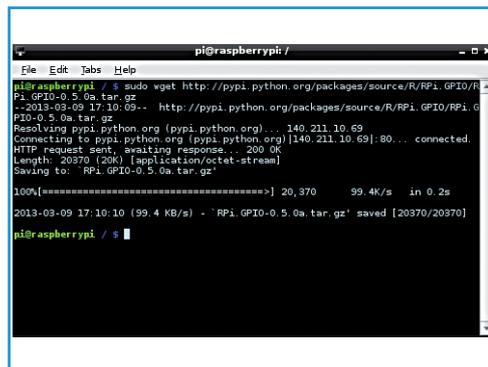


Bild 1. LXTerminal.

fügung. Die Änderung ist zwar klein, aber wichtig, wenn man Peripherie per I²C anschließen will.

Die GPIO-Library von Python

Da die Beispiele in Python geschrieben sind, trifft es sich gut, dass Python automatisch mit der Raspbian-Distribution installiert wird. Zum Zugriff auf die GPIOs muss allerdings noch eine passende Hardware-I/O-Library installiert werden. Es sind einige Libraries einsetzbar; für unsere Zwecke ermöglicht die Python-Library RPi.GPIO den Zugriff auf die RPi.GPIO-Pins.

Wenn Sie die Python-Development-Tools und die Python-GPIO-Library noch nicht herunter-

geladen haben, dann verwenden Sie dazu das LXTerminal (siehe **Bild 1**) auf RPi.

Die Python-Development-Tools lädt man mit:

```
sudo apt-get install python-dev
```

Dann wird das Archiv mit der Python-GPIO-Library [1] installiert:

```
wget http://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.5.0a.tar.gz
```

Nach dem Download muss das Archiv expandiert werden:

```
tar -zxf RPi.GPIO-0.5.0a.tar.gz
```

Anschließend wird ein neues Verzeichnis mit den Python-Dateien erzeugt. Man tippe:

```
cd RPi.GPIO-0.5.0a
```

Das Paket wird dann so installiert:

```
sudo python setup.py install
```

Nun sollte die Python-RPi.GPIO-Library installiert sein.

Beispiel: blinky.py

Nach der Installation von RPi.GPIO bietet sich das Schreiben eines Test-Programms an, das eine LED blinken lässt. **Bild 2** zeigt den Aufbau. Mit Hilfe eines kleinen Steckbretts (Mini-Piio ProtoBoard [2]) wird eine LED über einen 680-Ω-Vorwiderstand zwischen GPIO17 (Pin 11) und Masse (0 V) gelegt.

Nachdem Bauteile und Drahtstücke gesteckt sind, wird per Doppelklick auf das IDLE-Icon auf dem Desktop des RPi die Python Shell und die IDE (**Bild 3**) gestartet. Mit dem Menüpunkt „File“ wird ein neues Programm erzeugt. Hierdurch startet dann der Editor der IDE.

Im IDLE-Editor (**Bild 4**) wird nun das Programm wie in **Listing 1** eingegeben. Ist die Eingabe erledigt, sichert man es. Um das Programm ausführbar zu machen, gibt man im LXTerminal folgende Befehle ein:

```
chmod +x blinky.py
```

Anschließend kann das Programm durch folgende Befehle gestartet werden:

```
sudo ./blinky.py
```

Tipp: Wenn IDLE via LXTerminal gestartet wird und dabei **sudo** vor dem Programm-Namen steht (z.B. **sudo idle**), dann verfügt man über die nötigen Rechte, um das RPi.GPIO-Programm in IDLE auszuführen.

RPi.GPIO: Pin-Nummern

Bei RPi.GPIO gibt es zwei Arten der Pin-Nummerierung von RPi. Bei eigenen RPi.GPIO-Programmen muss daher das Schema der GPIO-Nummerierung angegeben werden: entweder `GPIO.setmode(GPIO.BOARD)` oder

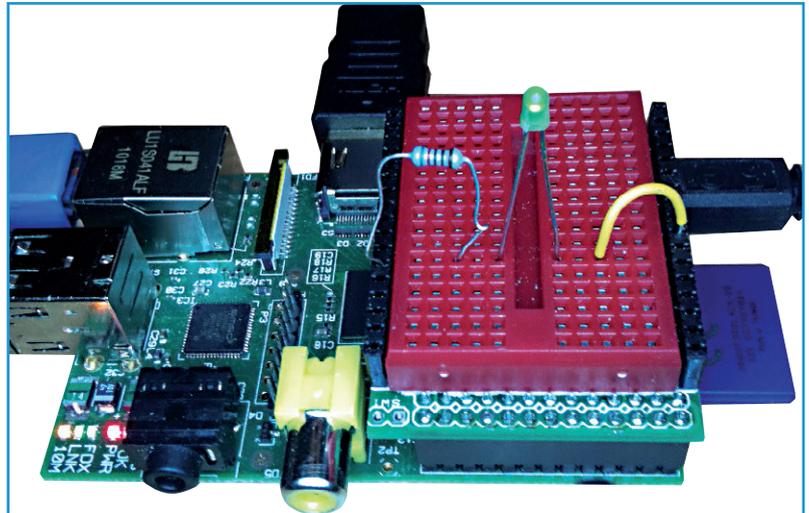


Bild 2. RPi mit Steckbrett.

Listing 1: blinky.py

```
#!/usr/bin/python
import time
import RPi.GPIO as GPIO

# Configure Pi's GPIO pins
GPIO.setmode (BCM)
GPIO.setup (17,GPIO.OUT)

# Program loop
while True :
    GPIO.output (17,True)
    time.sleep (1)
    GPIO.output (17,False)
    time.sleep (1)
```

Bild 3. Python-Shell IDLE.



GPIO.setmode(GPIO.BCM).
 Bei GPIO.setmode(GPIO.BOARD) bezieht man sich direkt auf die Pin-Nummern des Expansion Headers P1 auf dem RPi-Board. Bei GPIO.setmode(GPIO.BCM) werden die Signalnamen des Controllers verwendet.
 Da dies verwirrend aussieht, stellt die **Tabelle 3** die P1-Pin-Nummern und die GPIO Signalnamen gegenüber.

(130110)

Weblinks

- [1] www.elektor.de/120483
- [2] <https://pypi.python.org/pypi/RPi.GPIO>
- [3] www.dtronixs.com

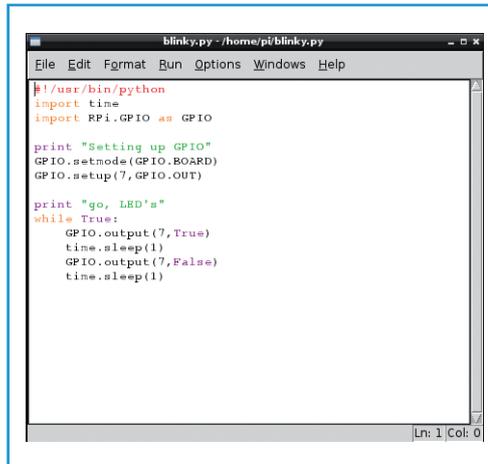


Bild 4. IDLE-Editor.

| Tabelle 3. GPIO.setmode(GPIO.BCM) und GPIO.setmode(GPIO.BOARD) | | | |
|--|--------------|--------------|------------|
| Pin-Name | Pin-Funktion | GPIO.setmode | |
| | | GPIO.BCM | GPIO.BOARD |
| P1-01 | 3,3 V | - | - |
| P1-02 | 5,0 V | - | - |
| P1-03 | GPIO0/2* | 0/2 | 3 |
| P1-04 | 5.0V | - | - |
| P1-05 | GPIO1/3* | 1/3 | 5 |
| P1-06 | GND | - | - |
| P1-07 | GPIO4 | 4 | 7 |
| P1-08 | GPIO14 | 14 | 8 |
| P1-09 | GND | - | - |
| P1-10 | GPIO15 | 15 | 10 |
| P1-11 | GPIO17 | 17 | 11 |
| P1-12 | GPIO18 | 18 | 12 |
| P1-13 | GPIO21/27* | 21 | 13 |
| P1-14 | GND | - | - |
| P1-15 | GPIO22 | 22 | 15 |
| P1-16 | GPIO23 | 23 | 16 |
| P1-17 | 3,3 V | - | - |
| P1-18 | GPIO24 | 24 | 18 |
| P1-19 | GPIO10 | 10 | 19 |
| P1-20 | GND | - | - |
| P1-21 | GPIO9 | 9 | 21 |
| P1-22 | GPIO25 | 25 | 22 |
| P1-23 | GPIO11 | 11 | 23 |
| P1-24 | GPIO8 | 8 | 24 |
| P1-25 | 0 V | - | - |
| P1-26 | GPIO7 | 7 | 26 |

Hinweis: * Bei Revision 2 geändert.