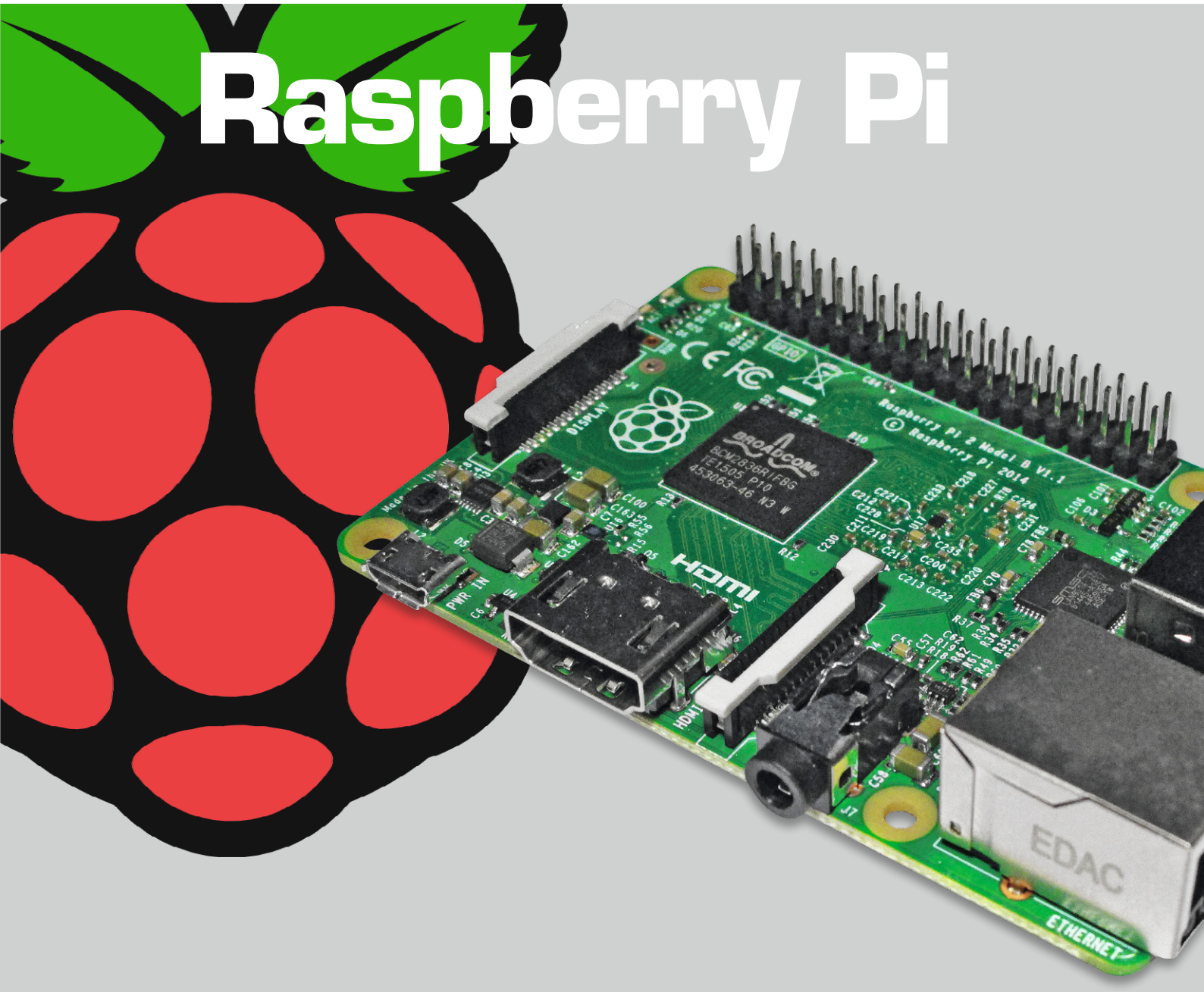
Vom Start bis zur Home-Automation

Raspberry Pi



In diesem Teil der Raspberry-Pi-Artikelserie werden Ein- und Ausgabemöglichkeiten des Raspberry Pi dargestellt. Speziell geht es um die Nutzung der I/O-Pins. Dabei bilden die im Teil 2 vorgestellten Programmiermöglichkeiten die Basis.

Teil 3 – Ein- und Ausgabe

Verbindungen zur Außenwelt

Auf der Raspberry-Pi-Platine sieht man sehr viele verschiedene Anschlussmöglichkeiten (Bild 1). Für die Grundinstallation und für die Interaktion über einen HDMI-Monitor werden genutzt: Spannungsversorgung, SD-Karte, HDMI, USB, LAN. Außerdem gibt es noch Anschlüsse für Audio, Kamera und Spezial-Display sowie den sogenannten GPIO-Header (= Erweiterungsanschluss/Erweiterungsleiste), auf den im Folgenden detaillierter eingegangen wird.

In Bild 1 kann man sehen, dass der GPIO-Header für einen Raspberry Pi 2 aus zwei Reihen zu je 20 Pins besteht. (Je nach Raspberry-Version sind es ggf. auch zwei Reihen zu je 13 Pins). Der Pin 1 ist auf der Platinenunterseite quadratisch statt rund gekennzeichnet. Von oben gesehen ist Pin 1 der Pin, der dem Kartenslot am nächsten ist. Die untere Reihe enthält Pins mit ungerader Nummer. Die obere Reihe enthält die geraden Pin-Nummern.

GPIO steht für „General Purpose Input Output" – frei übersetzt: „Eingabe und Ausgabe zur universellen Verwendung". Die meisten Pins des GPIO-Headers lassen sich als Ausgangs-Pin oder als Eingangs-Pin nutzen. Einige Pins des GPIO-Headers liegen an 3,3 V, 5 V oder 0 V (Gnd). Viele Pins haben besondere Funktionen (über digitale Ausgabe/Eingabe hinaus). Die Spannung an einem Ausgangs-Pin ist 3,3 V (nicht 5 V) oder 0 V.

Man kann an einen als Ausgang definierten Pin zum Beispiel eine (Low-Current-)LED mit Vorwiderstand gegen Gnd anschließen (Bild 2). Achtung: Die Pins können nur sehr geringe Ströme liefern! Man muss daher größere Lasten per Treiber (Transistor, MOSFET, Treiber-IC) schalten. Man kann die LED mit dem Vorwiderstand auch mit ihrer Anode an einen 3,3-V-Pin anschließen und die Katodenseite an einen als Ausgang geschalteten Pin.

Ein GPIO-Pin kann per Software als Eingangs-Pin geschaltet werden. Das ist auch die Standardeinstellung. An einen Eingangs-Pin lässt sich zum Beispiel ein Taster, Schalter oder Kontakt anschließen. Je nach Projekt kann gegen 3,3 V bzw. gegen Gnd geschaltet werden. Beim Schalten nach Gnd sollte ein Pull-up-Widerstand (R2 in Bild 2, ca. 10 kΩ) verschaltet werden, damit bei offenem Kontakt klare Spannungspegel anliegen.

Bei Schalten gegen 3,3 V sollte ein Pull-down-Widerstand verwendet werden. Je nach Programmierumgebung kann man diesen externen Pull-up- bzw. Pull-down-Widerstand durch einen intern per Software aktivierten Widerstand ersetzen. Der Widerstand R3 in Bild 2 (ca. 1 kΩ) ist ein Schutzwiderstand für den Pin für den Fall, dass durch ungünstige

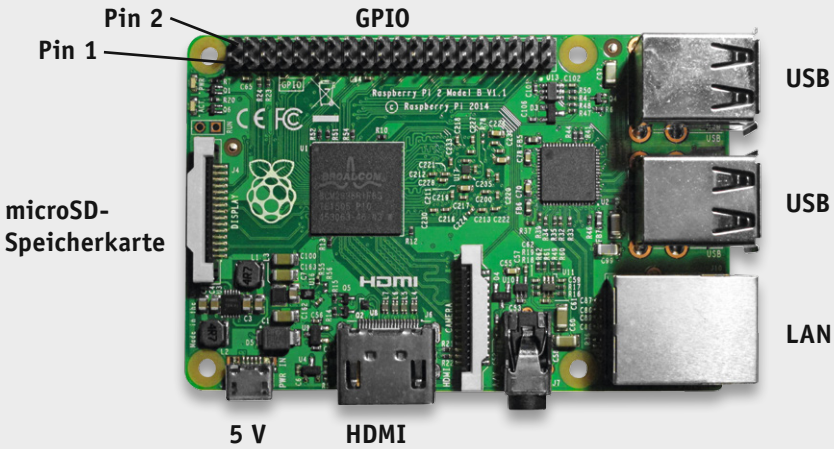


Bild 1: Die wichtigsten Schnittstellen des Raspberry Pi

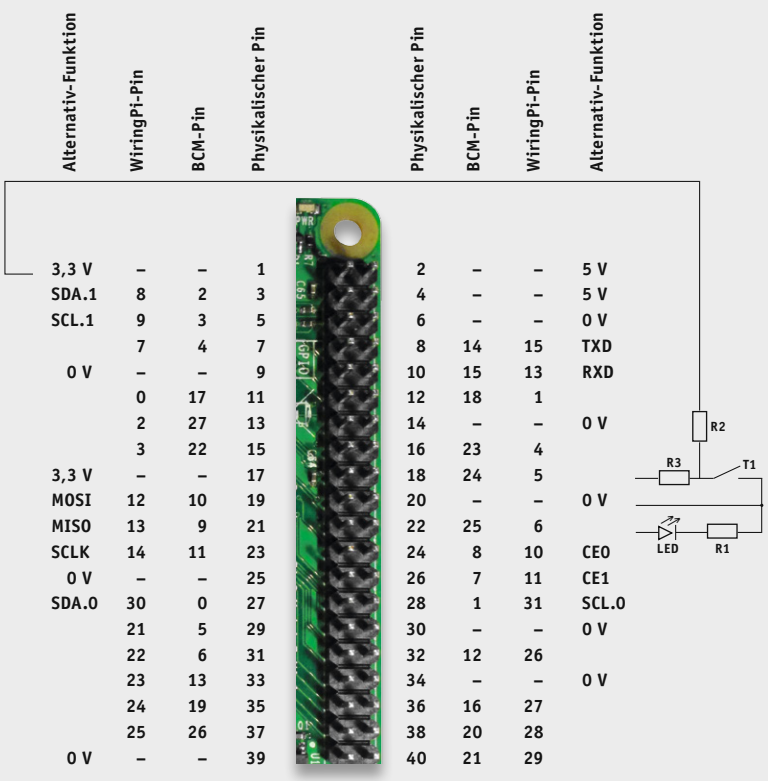


Bild 2: GPIO-Header des Raspberry Pi 2 inklusive angeschlossener LED und angeschlossenem Schalter/Taster/Kontakt

Konfigurationsverhältnisse 3,3 V direkt mit Gnd verbunden wäre.

Achtung: An einen GPIO-Pin dürfen niemals mehr als 3,3 V angelegt werden!

Die Pins 2 und 4 stellen jeweils 5 V zur Verfügung und lassen einen höheren Strom zu: Stromlieferung Netzteil (z. B. 1 A) minus Raspberry-Verbrauch (700 mA, je nach Modell) ergibt den möglichen Strom für die Pins (z. B. 300 mA).

Je nach Board-Revision hat der GPIO-Header insgesamt 26 bzw. 40 Pins und je nach Board-Revision gibt es Unterschiede bei der Pinbelegung. Man muss sich deshalb über die vorliegende Revision des Raspberry-Boards im Klaren sein.

Informationen über den eingesetzten Raspberry Pi kann man aus der Datei /proc/cpuinfo auslesen:

```
pi@raspberrypi ~ $ cat /proc/cpuinfo
...
Hardware:  BCM2709
Revision:  a01041
Serial:    00000000c9579632
pi@raspberrypi ~ $
```

Wichtig ist die Angabe der Revision (hier a01041). Aus Tabelle 1 (vorletzte Zeile) ersehen wir, dass ein Raspberry Pi 2 mit einem 40-Pin-GPIO-Header vorliegt. Die Belegung der Pins zeigt Bild 2.

Alternativ kann man auch in der Python-Umgebung mit *GPIO.RPI_REVISION* oder (neuer und um-

Tabelle 1

Raspberry-Modelle

	Modell A	Modell B Rev 1	Modell B Rev 2	Modell B+	Modell A+	Pi 2 Modell B
	„teilbestücktes Modell B“			„ersetzt Modell B“	„kompakte Neuentwicklung“	„ähnlich zu Modell B+, aber leistungsfähiger“
Monat/Jahr	4/’12 R1 9/’12 R2	4/’12	9/’12	7/’14	11/’14	2/’15
Platine Breite	56,0 mm	56,0 mm	56,0 mm	56,0 mm	56,0 mm	56,0 mm
Platine Länge	85,6 mm	85,6 mm	85,6 mm	85,6 mm	65,0 mm	85,6 mm
USB	1	2	2	4	1	4
Arbeitsspeicher	256 MB	256 MB	256 MB 512 MB	512 MB	256 MB	1024 MB
Kartenslot	SD	SD	SD	microSD	microSD	microSD
LAN	nein	ja	ja	ja	nein	ja
GPIO-Header	26	26	26	40	40	40
SoC System-on-a-Chip	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836
CPU-Architektur	ARMv6	ARMv6	ARMv6	ARMv6	ARMv6	ARMv7
Kerne	1	1	1	1	1	4
Takt	700 MHz	700 MHz	700 MHz	700 MHz	700 MHz	900 MHz
Leistungsaufnahme	2,5 W	3,5 W	3,5 W	3,0 W	1,2 W	4,0 W
Revision	0007 0008 0009	0002 0003	0004 0005 0006 000d 000e 000f	0010 0013	0012	A21041 A01041
PCB Revision	2.0	1.0	2.0	1.0 1.2	1.0	3.0

fangreicher) mit *GPIO.RPI_INFO* Informationen über den vorliegenden Raspberry Pi erhalten – und ggf. im Programm verwenden. Nach Aufruf der Python-Umgebung mit *python* (bzw. *python3*) werden die entsprechenden Befehle eingegeben und man erhält die entsprechenden Informationen über das Board. Das funktioniert unter Python 2.7 ebenso wie unter Python 3. Als Ausgabe erfährt man die PCB-Revision (hier 3):

```
pi@raspberrypi ~ $ python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO as GPIO
                    (oder   from RPi import GPIO )
>>> GPIO.RPI_REVISION
3
>>> GPIO.RPI_INFO
{'P1_REVISION': 3, 'RAM': '1024M', 'REVISION': 'a01041', 'TYPE': 'Pi2 Model B', 'PROCESSOR': 'BCM2836', 'MANUFACTURER': 'Sony'}
>>> GPIO.RPI_INFO['P1_REVISION']
3
>>> quit()
pi@raspberrypi ~ $
```

Kommandozeile

Als Einstieg bzw. für schnelle Aktionen kann man die GPIO-Pins von der Linux-Kommandozeile ansteuern. Dabei lernt man, dass die GPIO-Pins – wie bei Linux üblich – als (virtuelle) Dateien betrachtet werden. Standardmäßig besitzt ein Raspbian-System unter dem Root-Verzeichnis ein Verzeichnis mit dem Namen /sys/class/gpio (Bild 3). In diesem Verzeichnis gibt es eine Datei namens export.

Ausgabe (LED ansteuern):

Mit einem echo-Befehl in der Kommandozeile und dem Umleitungszeichen (>) wird die Nummer eines GPIO-Pins in die export-Datei geschrieben:

```
pi@raspberrypi ~ $ echo "25" > /sys/class/gpio/export
```

Dadurch wird ein entsprechendes Verzeichnis (gpio25) mit verschiedenen Dateien, zum Beispiel direction und value, angelegt. Die Dateien direction bzw. value müssen mit entsprechenden Schreibrechten versehen werden:

```
pi@raspberrypi ~ $ sudo chmod 666 /sys/class/gpio/gpio25/direction
pi@raspberrypi ~ $ sudo chmod 666 /sys/class/gpio/gpio25/value
```

In die Datei direction wird nun „out" (für einen Ausgabe-Pin) bzw. „in" (für einen Eingabe-Pin) geschrieben:

```
pi@raspberrypi ~ $ echo "out" > /sys/class/gpio/gpio25/direction
```

Nachdem der Pin auf diese Weise als Ausgabe-Pin definiert wurde, kann der Pin durch Schreiben einer „1" bzw. einer „0" auf „high" (3,3 V) bzw. „low" (0 V) geschaltet werden:

```
pi@raspberrypi ~ $ echo "1" > /sys/class/gpio/gpio25/value
pi@raspberrypi ~ $ echo "0" > /sys/class/gpio/gpio25/value
....
pi@raspberrypi ~ $ echo "1" > /sys/class/gpio/gpio25/value
```

Eine angeschlossene LED bzw. ein anderer Verbraucher (über eine Treiberstufe) kann damit ein- und ausgeschaltet werden. Am Schluss sollte man das System wieder „aufräumen", indem man die GPIO-Pins wieder freigibt:

```
pi@raspberrypi ~ $ echo "25" > /sys/class/gpio/unexport
pi@raspberrypi ~ $
```

Eingabe (Schalter/Taster abfragen):

Das Ansprechen von Eingabe-Pins funktioniert ähnlich. Erzeugen des entsprechenden GPIO-Verzeichnisses durch Schreiben in die Datei export:

```
pi@raspberrypi ~ $ echo "24" > /sys/class/gpio/export
```

Schreibrechte definieren:

```
pi@raspberrypi ~ $ sudo chmod 666 /sys/class/gpio/gpio24/direction
pi@raspberrypi ~ $ sudo chmod 666 /sys/class/gpio/gpio24/value
```

Abfragen des am Pin anliegenden Spannungslevels durch Auslesen der Datei value mit dem cat-Befehl. Als Ergebnis wird eine 0 bzw. eine 1 ausgegeben:

```
pi@raspberrypi ~ $ cat /sys/class/gpio/gpio24/value
0
pi@raspberrypi ~ $ cat /sys/class/gpio/gpio24/value
1
pi@raspberrypi ~ $ cat /sys/class/gpio/gpio24/value
0
```

Auf diese Weise kann abgefragt werden, ob ein angeschlossener Kontakt offen oder geschlossen ist.

Zum Schluss sollten die GPIO-Pins wieder freigegeben werden:

```
pi@raspberrypi ~ $ echo "25" > /sys/class/gpio/unexport
pi@raspberrypi ~ $
```

Skript-Programmierung

Das Eingeben der einzelnen Befehle in der Kommandozeile ist sehr zeitaufwendig und fehlerträchtig. In der Praxis wird man daher – besonders für wiederkehrende Aufgaben – lieber ein kleines Skript erstellen.

Mit *sudo nano blinken.sh* und Eingeben der Befehle im Editor wird beispielsweise ein Skript mit dem Namen blinken.sh erstellt (Bild 4), welches eine LED 10-mal blinken lässt.

Erläuterungen:

Das Doppelkreuz leitet Kommentare ein. Durch Schreiben in die export-Datei wird der Pin für die Benutzung eingerichtet. Nach Einrichtung der Schreibrechte für die Dateien direction und value und Schreiben von „out" in die Datei direction kann eine „1" oder eine „0" in die Datei

```
/
  sys
    class
      gpio
        export
        gpio24
          direction
          value
        gpio25
          direction
          value
        unexport
```

Bild 3: Dateisystem für GPIO

```
#!/bin/bash
# Aufruf mit     ./blinken.sh    nachdem die Datei mir sudo chmod 777 blinken.sh ausführbar gemacht
wurde
# GPIO Ausgabe
# LED blinkt an GPIO25
echo „25“ > /sys/class/gpio/export                 # GPIO25 anlegen
sudo chmod 666 /sys/class/gpio/gpio25/direction    # Schreibrechte vergeben
sudo chmod 666 /sys/class/gpio/gpio25/value        # Schreibrechte vergeben
echo „out“ > /sys/class/gpio/gpio25/direction      # Richtung: „out“ = Ausgabe    „in“ = Eingabe

for i in {1..10}                                   # Schleife fuer 1 bis 10
do
  echo -n „Durchlauf„                              # -n damit Ohne Zeilenumbruch
  echo „$i“                                        # Laufvariable i ausgeben
  echo „1“ > /sys/class/gpio/gpio25/value          # „1“ = Ausgabe von 3,3 Volt am GPIO-Pin GPIO25
  sleep 1                                          # 1 Sekunde warten
  echo „0“ > /sys/class/gpio/gpio25/value          # „0“ = Ausgabe von 0 Volt
  sleep 1                                          # 1 Sekunde warten
done

echo „25“ > /sys/class/gpio/unexport               # Bereinigung
```

Bild 4: Skript-Programm blinken.sh im Editor

value geschrieben werden, was im Beispielskript in einer for-Schleife erfolgt. Am Schluss des Skripts wird der Pin wieder deaktiviert (unexport).

Die Ausführung der Skript-Datei erfolgt mit:
pi@raspberrypi ~ $ bash blinken.sh
Die Rechte für die Datei blinken.sh sind standardmäßig: rw-r--r-- (644).
Nun werden Ausführungsrechte vergeben (statt 644 = rw-r—r-- auf 777 = rwxrwx):
pi@raspberrypi ~ $ sudo chmod 777 blinken.sh
und dann erfolgt der Aufruf mit:
pi@raspberrypi ~ $ ./blinken.sh

```
pi@raspberrypi ~ $ ./blinken.sh
Durchlauf 1
Durchlauf 2
Durchlauf 3
Durchlauf 4
Durchlauf 5
Durchlauf 6
Durchlauf 7
Durchlauf 8
Durchlauf 9
Durchlauf 10
pi@raspberrypi ~ $
```

Bild 5: Ausführung der Skript-Datei blinken.sh (die LED blinkt im Sekundenrhythmus).

Jetzt könnte man noch das User-Verzeichnis (~ also /home/pi) in die PATH-Umgebungsvariable aufnehmen oder umgekehrt die Datei in ein Verzeichnis verschieben/kopieren, das in PATH enthalten ist (z. B. /usr/bin). Bild 5 zeigt die Ausführung der Skript-Datei. Die Ausgabe der Texte erfolgt zur Ablaufkontrolle. Wesentlich ist, dass die am Pin angeschlossene LED blinkt. Ein Skript zur Verwendung eines GPIO-Pins als Eingabe-Pin (zum Abfragen eines Kontaktes) zeigt Bild 6 und die Ausführung des Skript-Programms zeigt Bild 7.

```
#!/bin/bash
# Aufruf mit     ./eingabe.sh    nachdem die Datei mit sudo chmod 777 eingabe.sh ausführbar gemacht
wurde
# GPIO Eingabe
# Taster/Schalter/Jumper an GPIO24
echo „24“ > /sys/class/gpio/export                 # GPIO24 anlegen
sudo chmod 666 /sys/class/gpio/gpio24/direction    # Schreibrechte vergeben
sudo chmod 666 /sys/class/gpio/gpio24/value        # Schreibrechte vergeben
echo „in“ > /sys/class/gpio/gpio24/direction       # Richtung: „out“ = Ausgabe    „in“ = Eingabe

for i in {1..10}                                   # Schleife
do
  echo -n „Durchlauf „                             # -n damit ohne Zeilenumbruch
  echo -n „$i: „                                   # Laufvariable i ausgeben

  # Nur Anzeige des Pin-Zustandes ginge mit:
  # cat /sys/class/gpio/gpio24/value               # „1“ = 3,3 V liegt an GPIO-Pin GPIO24
                                                   # „0“ = 0 V liegt an GPIO-Pin GPIO24

  # Mit Auswertung:
  wert_am_pin=$(cat /sys/class/gpio/gpio24/value)  # Wert am Pin einer Variablen zuweisen
  if [ $wert_am_pin -eq 1 ]                        # Wenn „1“ am Pin anliegt ...
  then                                             # .. dann ..
    echo „Taster offen“                            # „Taster offen“ ausgeben
  else                                             # sonst ..
    echo „Taster geschlossen“                      # „Taster geschlossen“ ausgeben
  fi
  sleep 1                                          # 1 Sekunde warten
done

echo „24“ > /sys/class/gpio/unexport               # Bereinigung
```

Bild 6: Skript eingabe.sh

Erläuterungen:
Die Vorbereitungen (export, Benutzerrechte) entsprechen dem Output-Beispiel oben. Dann wird „in" in die Datei direction geschrieben und mit dem cat-Befehl wird der Spannungspegel am Pin als 1 bzw. 0 ausgegeben. Im Skript in Bild 6 wurde dieser Zustand mit IF abgefragt und ein entsprechender Text ausgegeben.

```
pi@raspberrypi ~ $ bash eingabe.sh
Durchlauf 1: Taster offen
Durchlauf 2: Taster offen
Durchlauf 3: Taster offen
Durchlauf 4: Taster offen
Durchlauf 5: Taster offen
Durchlauf 6: Taster geschlossen
Durchlauf 7: Taster geschlossen
Durchlauf 8: Taster geschlossen
Durchlauf 9: Taster geschlossen
Durchlauf 10: Taster geschlossen
pi@raspberrypi ~ $
```

Bild 7: Ausführung der Skript-Datei eingabe.sh

Mit WiringPi:

Eine sehr mächtige Bibliothek mit Befehlen für die Nutzung der GPIO-Pins stellt WiringPi dar („Elektronikwissen" und [1]).

Zum Testen, ob WiringPi installiert ist und welcher Raspberry vorliegt, gibt man *gpio –v* ein:

```
pi@raspberrypi ~ $ gpio -v
gpio version: 2.26
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty
Raspberry Pi Details:
Type: Model 2, Revision: 1.1, Memory: 1024MB,
Maker: Sony
```

Sehr nützlich ist der Befehl *gpio readall*, mit dem man eine Tabelle mit den Pinbelegungen und -stati bekommt (Bild 8). Bild 8 zeigt neben den „normalen"/ BCM-Pinbezeichnungen auch die Pin-Nummern nach dem WiringPi-System und die Alternativfunktionen der Pins. Außerdem wird angezeigt, ob ein Pin als Ausgang oder als Eingang konfiguriert ist und wie der Wert des Pins ist.

Bei Benutzung von WiringPi in der Kommandozeile (oder anderen Umgebungen) werden zunächst wieder die gewünschten Pins als Ausgabe- oder Eingabe-Pin definiert:

```
pi@raspberrypi ~ $ gpio export 25 out
pi@raspberrypi ~ $ gpio export 24 in
```

Danach können Ausgaben mit *gpio write* und Eingaben mit *gpio read* erfolgen.

Dabei kann man (mit dem Parameter -g) die BCM-GPIO-Nummerierung, (mit Parameter -1) die physikalische Pin-Nummerierung oder (ohne Parameter) die WiringPi-Pin-Nummerierung verwenden.

Mit BCM-GPIO-Nummerierung (mit Parameter -g):

```
pi@raspberrypi ~ $ gpio -g write 25 1
pi@raspberrypi ~ $ gpio -g write 25 0
pi@raspberrypi ~ $ gpio -g read 24
1
pi@raspberrypi ~ $ gpio -g read 24
0
```

Mit physikalischer Nummerierung (mit Parameter -1):

```
pi@raspberrypi ~ $ gpio -1 write 22 1
pi@raspberrypi ~ $ gpio -1 write 22 0
pi@raspberrypi ~ $ gpio -1 read 18
1
pi@raspberrypi ~ $ gpio -1 read 18
0
```

Mit WiringPi-Nummerierung (ohne Parameter -g oder -1):

```
pi@raspberrypi ~ $ gpio write 6 1
pi@raspberrypi ~ $ gpio write 6 0
pi@raspberrypi ~ $ gpio read 5
1
pi@raspberrypi ~ $ gpio read 5
0
```

```
pi@raspberrypi ~ $ gpio readall
+-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
|     |     |    3.3v |      |   |  1 ||  2 |   |      | 5v      |     |     |
|   2 |   8 |   SDA.1 |   IN | 1 |  3 ||  4 |   |      | 5V      |     |     |
|   3 |   9 |   SCL.1 |   IN | 1 |  5 ||  6 |   |      | 0v      |     |     |
|   4 |   7 | GPIO. 7 |   IN | 0 |  7 ||  8 | 1 | ALT0 | TxD     |  15 |  14 |
|     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     |  16 |  15 |
|  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 |   1 |  18 |
|  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
|  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 |   4 |  23 |
|     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 |   5 |  24 |
|  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
|   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | OUT  | GPIO. 6 |   6 |  25 |
|  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     |  10 |   8 |
|     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     |  11 |   7 |
|   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   |  31 |   1 |
|   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
|   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 |  26 |  12 |
|  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
|  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 |  27 |  16 |
|  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 |  28 |  20 |
|     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 |  29 |  21 |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
```

Bild 8: WiringPi-Pinanzeige

```
#!/usr/bin/python
# -*- coding: utf-8 -*-              Damit Umlaute verwendet werden können
# Aufruf mit   sudo blinken.py      oder  ./blinken.py  nachdem die Datei mit sudo chmod 777 blinken.py
ausführbar gemacht wurde
import RPi.GPIO as GPIO              # Zugriff zu GPIO-Pins ermoeglichen
boardRevision = GPIO.RPI_REVISION    # Boardrevision der Variablen zuweisen
print „Boardrevision:“,              # Das Komma verhindert Newline. Nur ein Space
print boardRevision                  # Variable mit  Boardrevision ausgeben ( 1 oder 2 oder 3 )
from time import sleep               # Benötigt, um sleep benutzen zu koennen

GPIO.setmode(GPIO.BCM)               # echte  GPIO-Nummerierung benutzen:  GPIO25 usw.
GPIO.setwarnings(False)              # Irritierende Warnungen unterdruecken

GPIO.setup(25,GPIO.OUT)              # GPIO25 an Pin 22 als Output setzen

print „ELV“                          # Bei Python 3 alle print als print(..)
print “Hallo Welt  Test: äöüß-”

while True:                          # Endlosschleife
  print “LED an”
  GPIO.output(25,1)                  # LED an GPIO25 ein
  sleep(1)                           # 1 Sekunde warten
  print „LED aus“
  GPIO.output(25,0)                  # LED an GPIO 25 aus
  sleep (1.0)                        # 1 Sekunde warten
```

Bild 9: Python-Programm LED-Blinker

```
pi@raspberrypi ~ $ sudo python blinken.py
Boardrevision: 3
ELV
Hallo Welt  Test: äöüß-
LED an
LED aus
LED an
LED aus
LED an
LED aus
LED an
LED aus
LED an
LED aus
LED an
LED aus
LED an
```

Bild 10: Laufzeit blinken.py

Weitere Infos:

- Raspberry Pi: www.raspberrypi.org/about
- Raspbian: www.raspbian.org
- Deutsches Raspberry Pi Forum: www.forum-raspberrypi.de
- Englischsprachiges Raspberry Pi Forum: www.raspberrypi.org/forums
- [1] WiringPi: www.wiringpi.com
- Python: www.python.org
- Python-Kurs: www.python-kurs.eu/kurs.php
- Tkinter: https://wiki.python.org/moin/TkInter
- Tkinter-Tutorial (deutsch): www.python-kurs.eu/python_tkinter.php

```
#!/usr/bin/python
# -*- coding: utf-8 -*-              Damit Umlaute verwendet werden können
# Aufruf mit   sudo eingabe.py     oder  ./eingabe.py  nachdem die Datei mit sudo chmod 777 eingabe.py
ausführbar gemacht wurde
import RPi.GPIO as GPIO                          # Zugriff zu GPIO-Pins ermoeglichen
boardRevision = GPIO.RPI_REVISION                # Boardrevision der Variablen zuweisen
print „Boardrevision:“,                          # Das Komma verhindert Newline. Nur ein Space
print boardRevision                              # Variable mit  Boardrevision ausgeben ( 1 oder 2 oder 3 )
from time import sleep                           # Benötigt, um sleep benutzen zu koennen

GPIO.setmode(GPIO.BCM)                           # echte  GPIO-Nummerierung benutzen:  GPIO25 usw.
GPIO.setwarnings(False)                          # Irritierende Warnungen unterdruecken

GPIO.setup(25,GPIO.OUT)                          # GPIO25 an Pin 22 als Output setzen
GPIO.setup(24,GPIO.IN,pull_up_down=GPIO.PUD_UP)  # GPIO024 als Input mit Pullup

print „ELV“                                      # Bei Python 3 alle print als print(..)
print “Hallo Welt  Test: äöüß-”

while True:                                      # Endlosschleife
  print “LED an”
  GPIO.output(25,1)                              # LED an GPIO25 ein
  sleep(1)                                       # 1 Sekunde warten
  print „LED aus“
  GPIO.output(25,0)                              # LED an GPIO 25 aus
  sleep (1.0)                                    # 1 Sekunde warten

  if GPIO.input(24)==0:                          # GPIO-Pin 24 abfragen.  Wenn 0 ..
     print „Taste gedrückt“                      # .. dann Status ‚geschlossen‘ ausgeben
  else:                                          # .. sonst ..
     print „Taste nicht gedrückt (offen)“        # Status ‚offen‘ ausgeben
```

Bild 11: Python-Programm Eingabe

```
#!/usr/bin/python
# LEDs schalten mit Pushbuttons mit Tkinter
# und einen GPIO-Pin abfragen.   Mit Flanken-INTERRUPT
# Aufruf mit:     sudo python ein_ausgabe.py
from Tkinter import *                          # Fuer Grafikumgebung.  Bei Python 3: tkinter
import RPi.GPIO as GPIO                        # Zugriff auf Pins ermoeglichen
GPIO.setmode(GPIO.BCM)                         # Echte GPIO-Nummerierung  GPIO25 usw.
GPIO.setwarnings(False)                        # Irritierende Warnungen unterdruecken

GPIO.setup(25,GPIO.OUT)                        # GPIO 25 als Output setzen
GPIO.setup(24,GPIO.IN,pull_up_down=GPIO.PUD_UP)  # GPIO24 mit Pullup-Widerstand aktivieren

root=Tk()
ledstatus=StringVar()                          # Stringvariable deklarieren
ledstatus.set(„Anfang“)                        # Zuweisung an Stringvariable
# Unterprogramme:
def an() :
  GPIO.output(25,1)                            # GPIO 25 einschalten
  return

def aus():
  GPIO.output(25,0)                            # GPIO 25 ausschalten
  return

def zustand_isr(pin):                          # Unterprogramm fuer Interrupt definieren
  if GPIO.input(24)==0:                        # Wenn 0 V am Pin anliegt ..
    ledstatus.set(„geschlossen“)               # und vor allem: Textvariable fuer Textfeld setzen
  else:
    ledstatus.set(„offen     „)                # Textvariable fuer Textfeld setzen
  return   #geht auch ohne Return(?)

root.title(„LEDs schalten“)
root.geometry(‚400x300‘)                        # Groesse des Hauptfensters
Label(text=“Klicken zum Schalten der LEDs“, fg=“#0A116B“).pack()

Label(root, text=‘\n     LED wird ein-      \n      bzw. ausgeschaltet     \n‘).pack()
Button(text=“LED  an“, command=an, background=“#33D638“, foreground=“#FFFFFF“).pack()   # Button fuer LED einschalten
Button(text=“LED aus“, command=aus, background=“#1DE4F2“, foreground=“#FFFFFF“).pack()  # Button fuer LED ausschalten
Button(text=“Zustand Taster?“, command=zustand_isr(24), background=“#DC0F16“, foreground=“#FFFFFF“).pack()
# GPIO24-Zustand abfragen
Label(root,textvariable=ledstatus,fg=“black“).pack()          # Textfeld fuer Anzeige, ob Taster offen oder geschlosssen ist
GPIO.add_event_detect(24, GPIO.BOTH, callback=zustand_isr)  # Bei steigender oder fallender Flanke (BOTH) an GPIO24 wird
                                                              die zustand_isr aufgerufen

root.mainloop()
```

Bild 13: Tkinter-Programm

Am Ende werden die GPIO-Pins wieder freigegeben:
pi@raspberrypi ~ $ gpio unexportall
Mit *man gpio* bekommt man das WiringPi-Manual.

Python

Das Listing in Bild 9 zeigt das Python-Programm eines LED-Blinkers. Es wird im Nano-Editor erstellt durch:
pi@raspberrypi ~ $ sudo nano blinken.py

Erläuterungen:
Durch den import-Befehl werden die benötigten GPIO-Module eingebunden. Mit *GPIO.setmode* wird festgelegt, welche Nummerierungsart verwendet werden soll. Durch *GPIO.setwarnings* wird eingestellt, dass nicht unnötig viele Warnungen ausgegeben werden. Mit *GPIO.setup* erfolgt die Festlegung als Ausgabe-Pin und mit *GPIO.output* wird in einer Schleife der Zustand des Pins auf 1 bzw. auf 0 gesetzt. Das Ausführen des Programms erfolgt mit:
pi@raspberrypi ~ $ sudo python blinken.py
Das „sudo“ ist nötig, weil Root-Rechte erforderlich sind. Zur Laufzeit sieht es auf dem Bildschirm aus wie in Bild 10, wobei eine am Pin angeschlossene LED im Sekundentakt blinkt.
Ein Programm zum Testen eines GPIO-Pins als Eingabe-Pin wird erstellt durch:
pi@raspberrypi ~ $ sudo nano eingabe.py
und ist in Bild 11 zu sehen. Hier wird der GPIO-Pin 24

```
pi@raspberrypi ~ $ sudo python eingabe.py
Boardrevision: 3
ELV
Hallo Welt  Test: äöüß-
LED an
LED aus
Taste nicht gedrückt (offen)
LED an
LED aus
Taste nicht gedrückt (offen)
LED an
LED aus
Taste gedrückt
LED an
LED aus
Taste gedrückt
```

Bild 12: Laufzeit Python-Programm blinken.py

Bild 14: Tkinter LED an/aus und Taster abfragen

mit *GPIO.setup* als Eingabe-Pin mit intern aktiviertem Pull-up-Widerstand definiert. Durch *GPIO.input* wird der Status eines GPIO-Pins abgefragt.

Zur Laufzeit sieht es auf dem Bildschirm aus wie in Bild 12. Dabei blinkt die angeschlossene LED und der Eingabe-Pin wird abgefragt.

Grafische Oberfläche mit Tkinter programmieren

Mit der bereits im ELVjournal 5/2015 vorgestellten Tkinter-Umgebung lassen sich sehr schnell grafische Anwendungen zur Ansteuerung der GPIO-Pins erstellen, weil ein Tkinter-Programm ein Pythonprogramm ist und deshalb wiederum dieselben Techniken wie oben benutzt werden können. Eine Tkinter-Datei mit dem Namen ein_ausgabe.py und dem Programm gemäß Bild 13 wird erstellt mit:

```
pi@raspberrypi ~ $ sudo nano ein_ausgabe.py
```

In Bild 13 sieht man, dass zur Benutzung der GPIO-Pins die Anweisungen wie oben im Python-Programm benutzt werden. Der Eingabe-Pin GPIO24 wird in diesem Beispiel durch einen Interrupt überwacht (GPIO.add_event_detect).

Alternativ könnte man den Pin auch per Polling regelmäßig abfragen. Zur Ausführung des Tkinter-Programms startet man mit *startx* die grafische Oberfläche von Raspbian:

```
pi@raspberrypi ~ $ startx
```

Dann öffnet man ein Terminalfenster unter: Menü – Zubehör – Lxterminal. Im Terminalfenster wird das Tkinter-Programm zur Ausführung gebracht:

```
pi@raspberrypi ~ $ sudo python ein_ausgabe.py
```

Es erscheint ein Fenster gemäß Bild 14.

Durch Klicken auf „LED an" bzw. „LED aus" wird die LED geschaltet. Im unteren Textlabel steht der Zustand des Eingabe-Pins. Per Interrupt wird auf Pegeländerungen am Pin reagiert. Der Button „Zustand Taster?" wird nicht gebraucht – er dient im Programm zum Aufzeigen einer Alternative zum Interrupt.

Ausblick

Hier wurden die Basismöglichkeiten der GPIO-Pin-Nutzung dargestellt. Man kann auch relativ schnell I²C, 1-Wire (DS18x20), SPI, UART usw. einbinden. Anleitungen und Beispiele findet man reichlich in den zahlreichen Büchern sowie auf sehr vielen sehr guten Internetseiten.

Im vierten Teil unserer Raspberry-Artikelserie wird der Zugriff auf den Raspberry über das (W)LAN und über das Internet beschrieben. Damit wird es möglich, über das Netz am Raspberry angeschlossene Geräte anzusteuern oder Sensoren usw. abzufragen, und die Basis für eigene Home-Automatisierungs-Projekte ist gelegt. ELV

Elektronikwissen

WiringPi

Mit WiringPi werden die Möglichkeiten der Linux-Shell und einiger Programmiersprachen um viele mächtige Befehle zur I/O-Steuerung erweitert. WiringPi ist ein Projekt von Gordon Henderson (www.wiringpi.com).

Vorteile bei Nutzung von WiringPi:
Man muss sich nicht mehr um unterschiedliche Raspberry-Pi-Versionen kümmern, weil es eine WiringPi-Pin-Nummerierung gibt, und man hat kurze Befehle für Ausgabe, Eingabe, PWM, I²C, SPI usw. zur Verfügung. WiringPi-Befehle lassen sich in der Kommandozeile und auch in Skripten oder Programmiersprachen benutzen.

Falls WiringPi noch nicht installiert ist:
System auf aktuellen Stand bringen:

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
```

Gitcore installieren:

```
pi@raspberrypi ~ $ sudo apt-get install git-core
```

Über Github die WiringPi Library klonen:

```
pi@raspberrypi ~ $ git clone git://drogon.net/wiringPi
```

In WiringPi-Ordner wechseln und alles updaten:

```
pi@raspberrypi ~ $ cd wiringPi
pi@raspberrypi ~ $ git pull origin
```

Installation ausführen:

```
pi@raspberrypi ~ $ ./build
```

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
Raspberry Pi 2 B, Starter-Set	CD-11 93 80	€ 89,95
Raspberry Pi 2 B, 1 GB	CD-11 93 85	€ 37,95
Edimax-USB-WLAN-Adapter	CD-11 18 84	€ 8,95
NOOBS-Betriebssystem für Raspberry Pi 2, auf microSD-Karte	CD-12 01 50	€ 14,95
Netzteil (5 V/2 A)	CD-11 78 49	€ 6,25
hama Multikartenleser Basic, USB 2.0	CD-12 17 52	€ 7,95
Funk-Tastatur mit Touchpad	CD-11 66 75	€ 36,95
Verbindungskabel-Set 25 cm	CD-11 78 53	€ 6,95
Verbindungskabel-Set 50 cm	CD-11 78 54	€ 7,95
Verbindungskabel für Raspberry Pi 2x 20-polig, 30 cm	CD-11 78 52	€ 5,95
Verbindungskabel für Raspberry Pi 2x 20-polig female, 25 cm	CD-11 97 41	€ 5,95
Buch: Raspberry Pi programmieren	CD-11 24 30	€ 30,–
Buch: Raspberry Pi programmieren mit Python	CD-11 57 69	€ 24,99
Buch: Linux mit Raspberry Pi	CD-11 54 45	€ 30,–
Buch: Raspberry Pi: Mach's einfach	CD-11 84 57	€ 30,–

Infos zu den Produkten/Bauteilen finden Sie im Web-Shop. Preisstellung Oktober 2015 – aktuelle Preise im Web-Shop.