Ditzinger Makerspace

Eine Initiative von
Stadtbibliothek und
Repair Café

Schnittstellen im Raspberry Pi und deren Bedienung

Dieses Dokument beschreibt die sogenannten „General Purpose Input/Output“ (GPIO), digitale Schnittstellen am Raspberry Pi, die per Programm ein- und ausgeschaltet werden können. Verschiedene Möglichkeiten zu deren Bedienung werden vorgestellt.

Inhalt

Harald Orlamünder, 24. Mai 2017

2. Bezeichnung der Schnittstellen

Der Raspberry Pi hat ab Modell B+ eine 40-polige Stiftleiste, über die er mit der Umgebung kommunizieren kann. Auf dieser Leiste liegen Stromversorgungs-Pins und Eingabe/Ausgabe-Pins. Leider gibt es für letztere drei verschiedene Nummernschemata:

- Pin-Nummer direkt am Prozessor (oft mit „BCM“ oder „GPIO“ abgekürzt, BCM steht für „Broadcom, den Hersteller des Chips, GPIO für „General Purpose Input/Output“, also allgemein verwendbarer Ein- bzw. Ausgang)
- Pin-Nummer auf der Stiftleiste (also direkter Hardware-Bezug, HW)
- Nummernschema des Programms „WiringPi“, auf das später eingegangen wird.

Die folgende Tabelle listet in der Mitte die Pins der Steckerleiste (HW), links und rechts davon die Pin-Nummer des Prozessors (BCM), mit GPIOx bezeichnet sowie die alternativen Funktionen der Schnittstellen (siehe nächstes Kapitel). Außen ist die Nummerierung des Programms „WiringPi“ gelistet.

Wiring-Pi	Raspberry Pi BCM	HW Pin	HW Pin	Raspberry Pi BCM	Wiring-Pi
	+ 3,3 V	1	2	**+ 5 V**	
8	(SDA1) GPIO 2	3	4	**+ 5 V**	
9	(SCL1) GPIO 3	5	6	**GND**	
7	(GPCLK0) GPIO 4	7	8	GPIO 14 (Seriell_TxD)	15
	GND	9	10	GPIO 15 (Seriell_RxD)	16
0	GPIO 17	11	12	GPIO 18 (PCM_CLK)	1
2	GPIO 27	13	14	**GND**	
3	GPIO 22	15	16	GPIO 23	4
	+ 3,3 V	17	18	GPIO 24	5
12	(SPI_MOSI) GPIO 10	19	20	**GND**	
13	(SPI_MISO) GPIO 9	21	22	GPIO 25	6
14	(SPI_SLCK) GPIO 11	23	24	GPIO 8 (SPI_CE0)	10
	GND	25	26	GPIO 7 (SPI_CE1)	11
30	(nur für I2C) ID_SD	27	28	ID_SC (nur für I2C)	31
21	GPIO 5	29	30	**GND**	
22	GPIO 6	31	32	GPIO 12	26
23	GPIO 13	33	34	**GND**	
24	GPIO 19	35	36	GPIO 16	27
25	GPIO 26	37	38	GPIO 20	28
	GND	39	40	GPIO 21	29

Daher muss man vor der Benutzung klären, nach welchem Nummernschema gearbeitet wird, bzw. das richtige Nummernschema einstellen. Standardmäßig wird das Nummernschema des Prozessors verwendet (BCM, GPIO).

3. Funktionen der Schnittstellen

Nach dem Einschalten sind die Schnittstellen inaktiv, elektrisch hochohmig, also weder Eingang noch Ausgang noch eine Sonderfunktion. Vor der Benutzung muss also jede Schnittstelle initialisiert werden. Alle Schnittstellen können als allgemein benutzbare Eingänge oder Ausgänge initialisiert werden (also wie der Name sagt: „GPIO“), einige Anschlüsse können aber auch für besondere Schnittstellenaufgaben initialisiert werden. Das sind:

- **SPI** (Serial Peripheral Interface, zur Verbindung entsprechender Bausteine)
- **I2C** (Inter IC Communication, zur Verbindung entsprechender Bausteine)
- **OneWire** (Kommunikation über einen Draht, zur Verbindung entsprechender Bausteine)
- **Serielle Schnittstelle** (wie RS232, aber nicht mit Normpegel sondern 0V / 3,3V)
- **PWM** (Pulse Width Modulation, liefert über ein nachgeschaltetes Tiefpassfilter eine analoge Spannung)

Auf diese Sonderfunktionen wird hier nicht weiter eingegangen.

4. Elektrische Eigenschaften der Schnittstellen

Die Schnittstellen-Pins sind direkt mit dem Prozessor verbunden. Da dieser mit einer Versorgungsspannung von 3,3V arbeitet, haben auch die Schnittstellen Logikpegel von 3,3V und 0V (mit GND für Ground bezeichnet). Das ist besonders wichtig wenn ein Pin als Eingang geschaltet wird - es darf nicht mehr als 3,3V angelegt werden, sonst wird der Prozessor zerstört.

Eine andere Einschränkung betrifft die Belastbarkeit der Prozessor-Pins: maximal 10 mA dürfen fließen. Das gilt für beide Logikpegel. Damit eignen sich gerade Low-Current-LEDs (üblicherweise mit 3 mA betrieben) zum Anschluss über einen Vorwiderstand. Werden höhere Ströme benötigt sind entsprechende Treiber vorzusehen.

5. Software-Bedienung der Schnittstellen

5.1. Shell-Kommandos

Linux behandelt Geräte in der gleichen Weise wie Dateien. Im Raspberry liegen die entsprechenden Einträge unter

```
/sys/class/gpio/
```

Dort befinden sich nach dem Booten zwei Dateien: „export“ und „unexport“. Für den Zugriff auf die Schnittstellen müssen weitere Dateien erzeigt werden. Alle Aktionen müssen als Root ausgeführt werden, also einfach immer „sudo“ vor den Befehl setzen. Die folgenden Abschnitte zeigen, wie man mit Shell-Befehlen (also auf der Kommandozeilen-Ebene) die Schnittstellen behandeln kann.

Will man eine Schnittstelle benutzen, dann muss sie zuerst initialisiert werden, damit überhaupt mit ihr gearbeitet werden kann. Dazu wird in die Datei „export“ die Nummer der Schnittstelle geschrieben. Achtung: es ist standardmäßig die Pin-Nummer des Prozessors. Als Beispiel wird GPIO17 initialisiert (also HW Pin 11):

```
$ sudo echo 17 > /sys/class/gpio/export
```

Jetzt existiert in diesem Directory ein neues Unterverzeichnis namens „gpio17“. Ob die „17“ in Anführungszeichen gesetzt wird oder ohne ist an dieser Stelle ohne Bedeutung.

Als nächstes muss man diesem GPIO eine Richtung geben. Dazu schreibt man in das neue Unterverzeichnis eine Datei mit dem Namen „direction“ mit der gewünschten Richtung, also z.B. als Ausgabe mit „out“ bzw. als Eingabe mit „in“:

```
$ sudo echo out >/sys/class/gpio/gpio17/direction
```

```
$ sudo echo in >/sys/class/gpio/gpio17/direction
```

Und schließlich kann man in eine Datei namens „value“ den Wert, also eine „0“ oder „1“ schreiben:

```
$ sudo echo 1 >/sys/class/gpio/gpio17/value
$ sudo echo 0 >/sys/class/gpio/gpio17/value
```

In der Standardeinstellung entspricht eine logische „0“ dem GND-Potential, die logische „1“ einem Spannungspegel von etwa 3,3 V. Manchmal wäre es sinnvoll, wenn die Zuordnung umgekehrt wäre. Das kann mit einem Kommando erzielt werden, in dem in eine Datei mit Namen „active-low“ eine „1“ geschrieben wird:

```
$ sudo echo 1 /sys/class/gpio/gpio17/active_low
```

Oft ist es auch sinnvoll, einem GPIO bei der Initialisierung einen Wert zu geben. Natürlich kann nach dem „direction“- gleich eine „value“-Kommando gegeben werden. Eleganter ist es, den Wert zu initialisieren. Auch dafür gibt es ein Kommando, es ist quasi eine Variante von „direction“:

```
$ sudo echo low >/sys/class/gpio/gpio17/direction
$ sudo echo high >/sys/class/gpio/gpio17/direction
```

Interessant ist hier, dass nicht „0“ und „1“ geschrieben werden, sondern „low“ und „high“. Aber mit dem vorher erwähnten „active-low“ wird das verständlich: Unabhängig von der späteren Zuordnung der Logikpegel werden diese hier unmissverständlich auf einen niedrigen oder hohen Spannungspegel gesetzt.

Da es für das weitere Spielen mit dem GPIO lästig ist, jedes Mal „sudo“ eingeben zu müssen, bietet es sich an, die Zugriffsrechte dieser beiden Dateien so zu ändern, dass der normale Nutzer Zugriff hat. Das geht mit dem Befehl „chmod“:

```
$ sudo chmod 660 /sys/class/gpio/gpio17/direction
$ sudo chmod 660 /sys/class/gpio/gpio17/value
```

Wenn man den Pin nicht mehr braucht sollte er wieder deaktiviert werden. Das geht indem man die GPIO-Nummer nach „unexport“ schreibt:

```
$ sudo echo 17 >/sys/class/gpio/unexport
```

Shell-Befehle können in eine Datei geschrieben und dann per Aufruf ausgeführt werden.

5.2. Benutzung von „WiringPi“

WiringPi ist ein Hilfsprogramm, das zum einen die Bedienung der Schnittstellen vereinfacht, andererseits sich in Programmiersprachen einbinden lässt, da entsprechende Bibliotheken mitgeliefert werden.

WiringPi ist nicht standardmäßig ins Betriebssystem integriert. Daher muss es zuerst installiert werden, dazu sind folgende Befehle notwendig:

```
$ sudo apt-get update
$ sudo apt-get install git-core
$ git clone git://git.drogon.net/wiringPi
$ cd wiringPi
$ ./build
```

Ob auf einem Raspberry WiringPi schon installiert ist kann man mit der Versionsabfrage feststellen:

```
$ gpio -v
```

Wenn keine Fehlermeldung kommt, ist WiringPi installiert. Das Programm liefert übrigens eine Übersicht über die verschiedenen Nummernschemata, ähnlich wie oben dargestellt. Das Kommando lautet:

```
$ gpio readall
```

Der Zugriff auf die Schnittstellen geht jetzt mit dem neuen „gpio“-Kommando viel einfacher als mit der Shell. In den folgenden drei Zeilen wird der GPIO 17 als Ausgang initialisiert, der Wert „1 geschrieben und der Anschluss gelesen mit Ausgabe auf dem Bildschirm:

```
$ gpio export 17 out
$ gpio -g write 17 1
$ gpio -g read 17
```

***Hinweis**: WiringPi benutzt standardmäßig seine eigene Nummerierung, außer beim export-Kommando. Die Option “-g“ sorgt dafür, dass die normale GPIO-Nummerierung benutzt wird, mit der Option -l kann die physikalische Pin-Nummerierung gewählt werden.*

An Ende sollte wie immer aufgeräumt werden:

```
$ gpio unexportall
```

WiringPi ist sehr mächtig und unterstützt auch die alternativen Funktionen der Schnittstellen. Für einen Überblick ist die man-Page hilfreich:

```
$ man gpio
```

5.3. Benutzung der Programmiersprache Python

Python ist eine einfache Programmiersprache, die beim Raspberry mit dem Betriebssystem mit installiert wird. Sie kann entweder über die Kommandozeile aufgerufen werden mit dem „idle“-Kommando:

```
$ idle
```

oder über die graphische Oberfläche: `Menu` - `Entwicklung` - `Python 2 (IDLE)`.

Es öffnet sich ein Fenster mit der Python-Shell. Hier können Programme entwickelt und gestartet werden.

Will man nun im Python auf die Schnittstellen-Pins zugreifen, dann muss die spezielle Bibliothek „RPi.GPIO“ geladen werden. Das geht mit folgendem Befehl:

```
import RPi.GPIO as GPIO
```

Jetzt muss festgelegt werden, welches Nummernschema benutzt werden soll (BCM oder BOARD), im folgenden Beispiel wird die Prozessorpin-Nummerierung verwendet. Dann werden die Warnungen ausgeschaltet. Das ist natürlich nicht notwendig, aber beim Spielen mit dem Anschluss sind sie störend. Im nächsten Schritt wird der zu verwendende Anschluss konfiguriert, hier Anschluss 17 als Ausgang. Und schließlich kann der Ausgang z.B. eingeschaltet werden:

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(17, GPIO.OUT)
GPIO.output(17, 1)
```

Natürlich lässt sich jetzt der Anschluss wieder ausschalten:

```
GPIO.output(17, 0)
```

Vorher wurde schon auf die Initialisierung eines Ausgangs hingewiesen. Das geht mit einer Erweiterung des „setup“-Kommandos:

```
GPIO.setup(17, GPIO.OUT, initial=GPIO.LOW)
```

Will man einen Anschluss als Eingang konfigurieren, geht das mit dem „GPIO.setup“-Kommando:

```
GPIO.setup(4, GPIO.IN)
```

Lesen geht dann mit GPIO.Input, wobei man natürlich mit dem Wert etwas machen muss, z.B. mit print auf dem Bildschirm ausgeben:

```
print  ( GPIO.input(4))
```

Am Ende eines Python-Programms sollten alle verwendeten Anschlüsse wieder in den Urzustand versetzt werden, was mit dem „cleanup“-Kommando erreicht wird:

```
GPIO.cleanup()
```

Ein in Python erstelltes Programm kann abgespeichert werden und später über seinen Namen aufgerufen werden

5.4. Benutzung von Hochsprachen

Für viele Hochsprachen wie C oder Pascal existieren entsprechende Bibliotheken für den Raspberry Pi. das sind entweder speziell dafür geschriebene oder bestehende wie „WiringPi“ lassen sich integrieren. Aber auch direkte Shell-Befehle lassen sich oft integrieren. Das folgende Beispiel zeigt die Initialisierung des Anschlusses 17 als Ausgang in einem Pascal-Programm:

```
gReturnCode := fpsystem(‘echo “17“ > /sys/class/gpio/export’);
gReturnCode := fpsystem(‘echo “out“ > /sys/class/gpio/direction’);
```

Das Thema Hochsprachen-Unterstützung soll hier nicht weiter vertieft werden.

6. Quellen

http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_GPIO.html
http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_GPIO_Shell.html
http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_GPIO_C.html
http://raspberrypiguide.de/howtos/raspberry-pi-gpio-how-to/
http://www.raspberry-pi-geek.de/Magazin/2013/05/
Tricks-zum-Programmieren-der-GPIO-Schnittstelle
http://www.elektronik-labor.de/Raspberry/RpiGPIO1.html
https://tutorials-raspberrypi.de/raspberry-pi-gpio-erklaerung-beginner-programmierung-lernen/
https://www.reichelt.de/reicheltpedia/index.php5/Anleitungen_f%C3%BCr_den_Raspberry_Pi
Franzis Raspberry Pi Maker Kit Elektronik
http://wiringpi.com/
https://projects.drogon.net/raspberry-pi/wiringpi/
https://projects.drogon.net/raspberry-pi/wiringpi/the-gpio-utility/
http://www.raspberrypi-tutorials.de/software/gpios-am-raspberry-pi-mit-wiringpi-schalten.html
https://sourceforge.net/p/raspberry-gpio-python/wiki/BasicUsage/
http://wiki.freepascal.org/Lazarus_on_Raspberry_Pi