

IT-Systeme

Prof. Dr. Torsten Edeler

HAW-Hamburg, Medientechnik

20. März 2017

Orga

Rahmen

① Organisatorisches

② Projektrahmen

Orga

Rahmen

① Organisatorisches

② Projektrahmen

Vorraussetzung zur Teilnahme am Kurs

Orga

Rahmen

- Sie haben das erste Studienjahr komplett bestanden!

Alternative "Classic mobile reloaded!"(Mobile Systeme)

- Kooperation mit Prof. Kabel (Dept. Design)
- Arbeit in gemischten Teams (Media Systems und Kommunikationsdesign)
- Ziel: Präsentation in der Semesterausstellung (Juli)
- Erstellung einer mobilen Applikation
- Beginn: 23. März um 9:30 im Raum 166
- **http:**
`//vorlesungsverzeichnis.design.haw-hamburg.de/kurs/sose17/classic-mobile-reloaded/`

Auf EMIL finden Sie alle Informationen zu dieser Veranstaltung. Insbesondere

- Skript,
- Aufgabenblätter,
- Forum für Ankündigungen.

Passwort für den Kurs „SS17 IT-Systeme“ lautet
sansibar99

Sie erreichen mich per E-Mail unter
torsten.edeler@haw-hamburg.de oder direkt über EMIL.

Ich antworte auf Ihre Nachrichten ausschließlich per E-Mail und nicht über EMIL.

Informationen zur Veranstaltung

- Projektorientiert
 - Zweier- bis Vierergruppen
 - Meilensteine (Konzept, Generalprobe, Ausstellung mit Poster)
 - Nach der Ausstellung folgt die Abgabe eines Projektberichts
- Ablauf
 - 4-5 Vorlesungen
 - Projekt und Gruppenfindung mit Vorstellung der Idee
 - Ca. 8 Termine für Projektbetreuung im „Digitaltechniklabor“

- Sie vollenden ein Projekt
- Präsentation am Ende der Veranstaltung
- Projektdokumentation
- Fristgerechte Abgabe / Teilnahme ist Voraussetzung zum Bestehen des Kurses (**PFLICHT!**):
 - Konzept (Als PDF per E-Mail)
 - Generalprobe
 - Ausstellung
 - Projektbericht (Als PDF per E-Mail)
- Die jeweiligen Termine bzw. Abgabefristen können Sie dem Zeitplan auf der nächsten Seite entnehmen.

SW	Datum	Bemerkung
1	14.03.17	-
2	21.03.17	Vorlesung MT: Edeler MS: Plaß
3	28.03.17	Vorlesung MT: Plaß MS: Edeler
4	04.04.17	Vorlesung MT: Edeler MS: Plaß
5	11.04.17	Vorlesung MT: Plaß MS: Edeler
6	18.04.17	Vorlesung MT: Edeler MS: Plaß
7	25.04.17	!Projektaufteilung!
8	02.05.17	!Konzeptabgabe!
9	09.05.17	
10	16.05.17	
11	23.05.17	
12	30.05.17	
13	06.06.17	
14	13.06.17	
15	20.06.17	!Generalprobe!
16	27.06.17	
17	04.07.17	
18	11.07.17	
-	13.07.17	!Ausstellung! (Jahresausstellung DMI)
-	20.07.17	!Projektbericht Abgabe!

!: Teilnahme-/Abgabepflicht

- Konzept
 - ca. 2 Din-A4 Seiten
 - Zum Konzept gehören Blockschaltbild, Beschreibung der Funktion etc.
 - Abgabe per Mail im PDF-Format mit Namen und Matrikelnummern aller Mitglieder
- Generalprobe
 - Vorzeigen eines Prototypen (Grundsätzliche Funktion muss erkennbar sein)
- Ausstellung
 - Poster Ihrer Projektidee und Lösung auf einer ansprechenden DIN-A2 Seite.
 - Vorzeigen des funktionsfähigen Gerätes.
- Projektbericht
 - Umfang 5-10 Seiten reiner Text (ohne Quellcode)
 - Darstellung von Idee und Lösung
 - Enthält Bilder, Schaltpläne, Quelltext

Bewertung der Arbeit

- Ihre Mitarbeit und Hausaufgaben sind Pflicht!

Es wird benotet:

- Individuelles Engagement
- Konzept
- Ausstellung des Projektergebnisses
- Poster
- Projektbericht

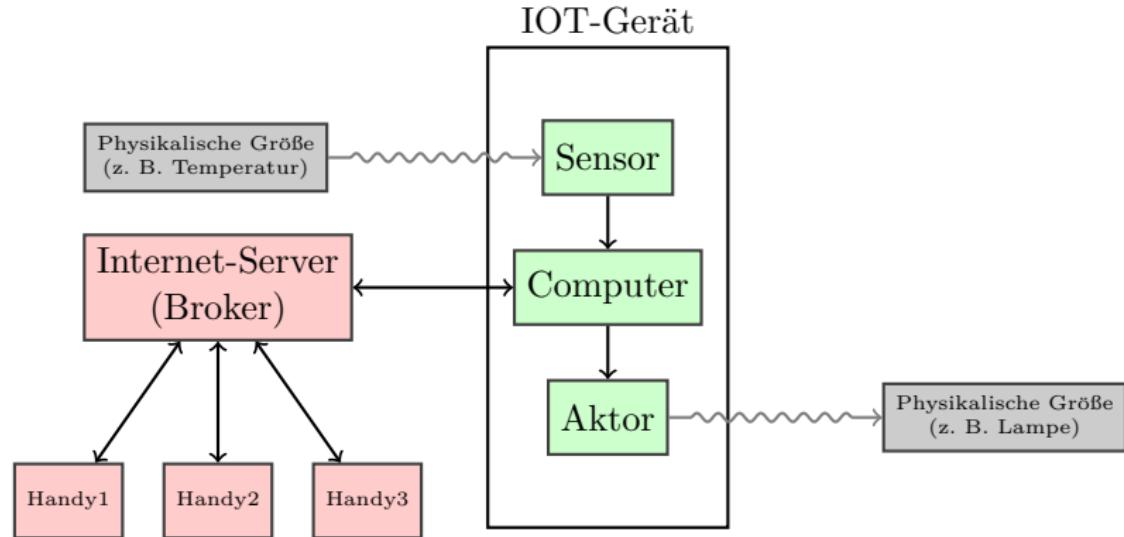
Orga

Rahmen

① Organisatorisches

② Projektrahmen

- Sie entwickeln ein IOT¹-Gerät, dass über Handy bedienbar ist.



Was ist das?

- Buzz-Word (Marketing)
- Computer werden unsichtbar und gehen in 'Dingen' auf.
- Der Terminus 'Internet' bezeichnet dabei jegliche Art der Vernetzung (z. B. Bluetooth, Zigbee...)
- Beispiele:
 - Der berühmte Kühlschrank
 - Philips Hue
 - Amazon Echo

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

③ Raspberry Pi

④ Python

⑤ Raspberry Pi Laborübung

⑥ Anschluss von Peripherie an den RPI

GPIO

SPI

⑦ Benutzerinteraktion

Ausgabe

Eingabe

Raspberry
Pi

Python

Raspberry
Pi

Laborübung

Anschluss
von
Peripherie
an den RPI
Interaktion

3 Raspberry Pi

4 Python

5 Raspberry Pi Laborübung

6 Anschluss von Peripherie an den RPI

7 Benutzerinteraktion

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Kleincomputer Raspberry Pi



- Eine Platine
- Komplett funktionsfähiger PC mit WLAN, USB, Ethernet etc.
- Linux Betriebssystem

Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI

Interaktion

- Einplatinencomputer
- Sehr günstig (ca. 40Euro)
- Betriebssystem Linux
- Einfache elektrische Schnittstelle (GPIO)
- Leistungsbedarf ca. 3Watt
- Sehr viele Sensoren / Aktoren (Zusatzhardware)
- Sehr gute Softwareunterstützung
- Viel Hilfe und Tutorials im Netz
- Sie finden sehr viele „Copy-Paste“-fähige Lösungen, um einzelne Sensoren und Aktoren anzusteuern.

Entwickelt durch britische Stiftung „Raspberry Pi Foundation“

- 2012 Modell B; 700MHz, 512MB Ram
- 2013 Modell A; 700MHz, 256MB Ram
- 2014 Modell A+; 700MHz, 256MB Ram
- 2014 Modell B+; 700MHz, 512MB Ram
- 2015 Rasp. 2 Model B; 900MHz Vierkern-CPU 1GB Ram
- 2015 Raspberry Pi Zero (1GHz Einkern-CPU)
- 2016 **Rasp. 3 Model B; 1,2GHz, Vierkern-CPU, 1GB Ram, 64bit, WLAN, Bluetooth**

Versionen (Bebildert)

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

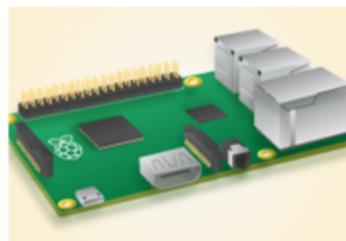
Interaktion



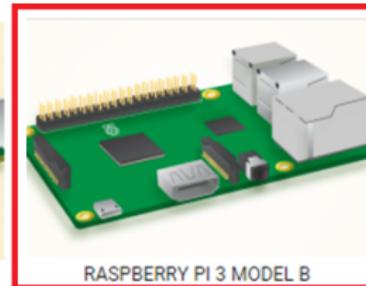
RASPBERRY PI 1 MODEL A+



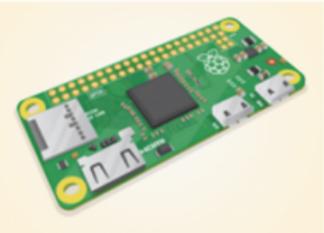
RASPBERRY PI 1 MODEL B+



RASPBERRY PI 2 MODEL B



RASPBERRY PI 3 MODEL B



RASPBERRY PI ZERO

Wo ist die Festplatte?

- Normale PCs haben eine Festplatte
- Der RPI nutzt hierfür eine SD-Karte \geq 8GB empfohlen



- Auf dieser sind das Betriebssystem und alle sonstigen Daten gespeichert.
- Das Betriebssystem samt Anleitung zur Erstellung einer SD-Karte finden Sie unter „<https://www.raspbian.org/>“
- Im Labor ist schon alles installiert

Wo ist die Tastatur und der Bildschirm?

- Normale PCs haben eine Tastatur und Bildschirm
- Grundsätzlich ist das auch beim RPI möglich (→usb, hdmi)
- Üblicher ist allerdings die Verbindung über eine Netzwerkshell (ssh)
- Unter Windows kann mit dem Programm „Putty“ (<http://www.putty.org/>) eine solche ssh-Verbindung aufgebaut werden.
- Weiteres dazu im Abschnitt „Raspberry Pi Laborübung“

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Raspberry
Pi

Python

Raspberry
Pi

Laborübung

Anschluss
von
Peripherie
an den RPI
Interaktion

3 Raspberry Pi

4 Python

5 Raspberry Pi Laborübung

6 Anschluss von Peripherie an den RPI

7 Benutzerinteraktion

Die Programmiersprache Python – Überblick

- Skriptsprache
- Fokus auf einfache Erlernbarkeit
- Große Nutzerbasis (Viel Hilfreiches im Internet)
- Auf dem RPI quasi die standard Skriptsprache

Nachteile:

- Langsamer als C (da interpretiert)

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI
Interaktion

Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI
Interaktion

- Entwicklung seit 1983
- Derzeit sind aktuell die Version 2.7 und 3.5
- Version 2 und 3 sind inkompatibel
 - Die Unterschiede sind marginal, jedoch relevant für die Ausführung.
 - Wir arbeiten mit Version 3

Die Entwicklung in Python basiert stark auf der Nutzung von sog. **Paketen**. Die meisten Pakete sind für Version 2 und 3 verfügbar.

Python installieren (Auf dem PC)

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

- Sie können Python zu Hause installieren.
 - Laden Sie sich dafür das Paket Anaconda
<https://www.continuum.io/downloads> herunter.
 - Verfügbar für Linux, MacOS und Windows
 - Achten Sie auf Python Version 3!
 - Videos:
 - Windows:
<https://www.youtube.com/watch?v=LvmpDyFyS7o>
- Pakete installieren:
 - Kommandozeile² `conda install <paketname>`
- Python auf aktuellen Stand bringen:
 - Kommandozeile `conda update anaconda`
- Informationen über das Programm `conda`:
<https://www.youtube.com/watch?v=YJC6ldI3hWk>

²Unter Windows erreichen Sie die Kommandozeile (Dosfenster) durch **Windowstaste → cmd**

Python installieren (Auf dem Raspberry Pi)

Raspberry
Pi
Python
Raspberry
Pi
Laborübung
Anschluss
von
Peripherie
an den RPI
Interaktion

- Auf dem Standardsystem ist Python immer installiert!
- Pythonversionen:
 - Version 2: `python`
 - Version 3: `python3`
- Pakete installieren (für Version 3):
 - `sudo pip3 update` (einmalig)
`sudo pip3 install <packetname>`
 - Derzeit gibt es ca. 100.000 Pakete für Python.
 - Übersicht hier: <https://pypi.python.org/pypi>

Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI

Interaktion

- Bücher

- Raspberry Pi programmieren mit Python
 - Autor: Michael Weigend
 - Verlag: mitp; Auflage: 3. Auflage 2016 (16. Mai 2016)
 - ISBN-13: 978-3958454293



- Internet

- Ein ganz nettes Tutorial finden Sie hier:
http://www.python-kurs.eu/python3_kurs.php
- Noch ein Tutorial für das *Jupyter-Notebook*:
http://nbviewer.jupyter.org/github/ehmatthes/intro_programming/blob/master/notebooks/hello_world.ipynb
- Mit Google erhalten Sie so gut wie immer eine Antwort.
- Youtube-Videos sind teilweise sehr hilfreich.

Übungsthemen Stichwortartig. Bitte machen Sie sich Notizen:

- Starten einer Python-Shell
- Starten Jupyter Notebook
- Ausgabe mit `print()`
- Variablen
- Funktionen
- Datentypen
 - Zahlen (Integer, Float)
 - Zeichenketten (Strings)
 - Sammelobjekte (Listen, Tuple, Dictionary)
- `if`-Abfragen
- `for`-Schleife
- `while`-Schleife

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI
Interaktion

③ Raspberry Pi

④ Python

⑤ Raspberry Pi Laborübung

⑥ Anschluss von Peripherie an den RPI

⑦ Benutzerinteraktion

Verwendung von Putty im Labor (Verbindungsaufbau)

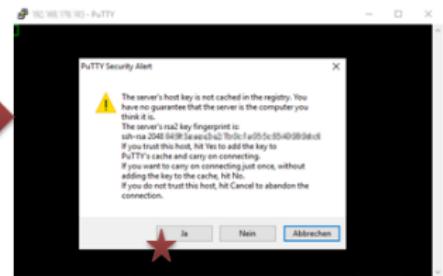
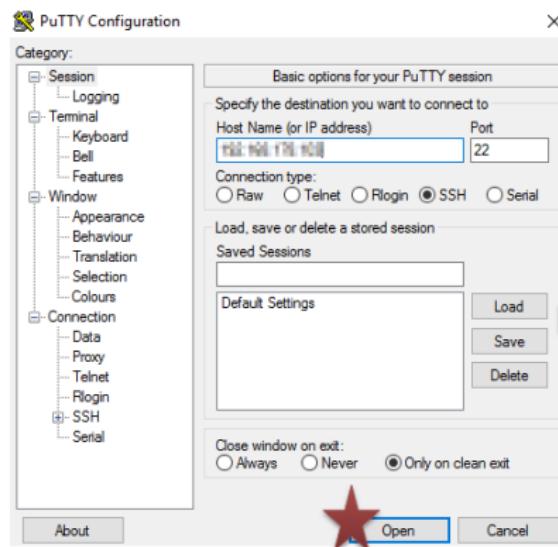
Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

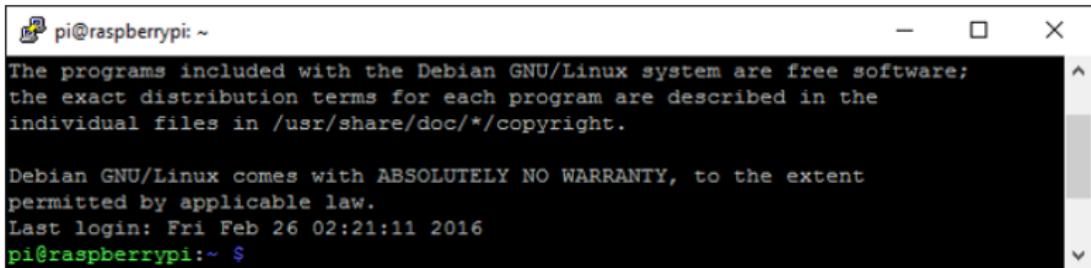


Verwendung von Putty im Labor (Login)

Benutzername: pi

Password: raspberry

Nach dem Login erscheint das Konsolenfenster. Hier können nun Linuxbefehle eingegeben werden



The screenshot shows a terminal window titled "pi@raspberrypi: ~". The window contains the following text:

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb 26 02:21:11 2016
pi@raspberrypi:~ $
```

Schreiben Sie bitte mit!

- SD-Karte (Größe / Belegung)
- Verzeichnis erstellen
- Textdatei erstellen
- Textdatei editieren
- Textdatei anzeigen
- Verzeichnis anzeigen
- Verzeichnis löschen
- Textdatei löschen

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Das erste Pythonprogramm

Erstellen Sie zunächst eine Textdatei mit Namen **test.py**:

```
pi@raspberrypi: ~/ptest
pi@raspberrypi:~ $ mkdir ptest
pi@raspberrypi:~ $ cd ptest
pi@raspberrypi:~/ptest $ nano test.py
```

```
pi@raspberrypi: ~/ptest
GNU nano 2.2.6                               File: test.py                         Modified
print("Hello, World")
```

The terminal shows the command to create a directory 'ptest' and change into it. Then, it uses the 'nano' text editor to create a file named 'test.py' containing the single line of code 'print("Hello, World")'. The status bar at the bottom right indicates the file is 'Modified'.

```
pi@raspberrypi: ~/ptest
GNU nano 2.2.6                               File: test.py                         Modified
print("Hello, World")
```

A blue box highlights the keyboard shortcut 'Ctrl-X' in the menu bar. A blue box also highlights the letter 'Y' in the confirmation dialog box.

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?

Y Yes Ctrl-Y

N No ^G Cancel

Das erste Pythonprogramm ausführen

Führen Sie das Programm aus:



A screenshot of a terminal window titled 'pi@raspberrypi: ~/ptest'. The window shows the command 'python3 test.py' being run, followed by the output 'Hello, World'. The terminal has standard window controls (minimize, maximize, close) and a scroll bar.

```
pi@raspberrypi:~/ptest $ python3 test.py
Hello, World
pi@raspberrypi:~/ptest $
```

- `python3` ist der sog. Interpreter
- Dieses Programm interpretiert (führt aus) jede Zeile in Ihrem Programm
- Das Programm besteht nur aus der einen Anweisung
`print("Hello, World")` → Bildschirmausgabe

Programmausführung über Webbrowser I

Raspberry
Pi

Python

Raspberry
Pi

Laborübung

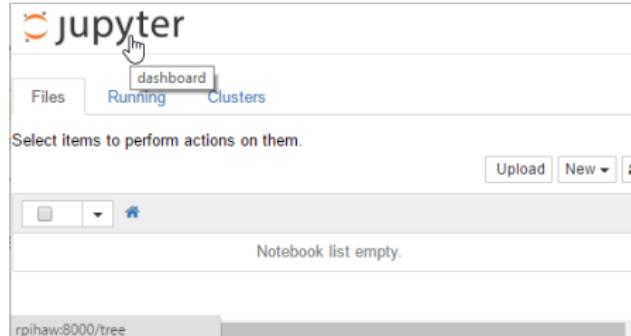
Anschluss
von
Peripherie
an den RPI

Interaktion

Auf den RPIs im Labor ist ein spezieller Webserver installiert, über den Sie ebenfalls Python-Programme ausführen können.

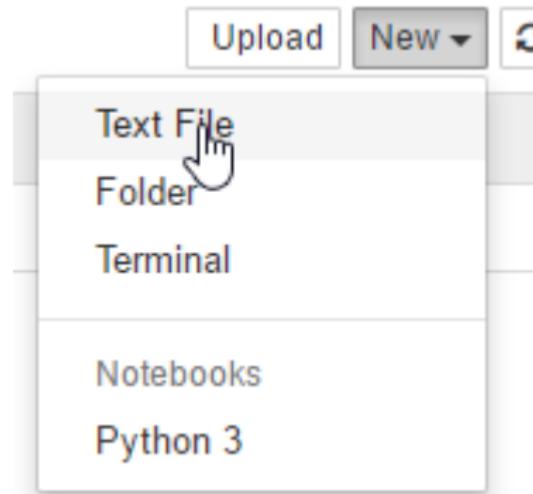
Gehen Sie wie folgt vor:

- Öffnen Sie mit dem Webbrowser die Seite <http://raspi<nr>.local:8000>
 - <nr> ersetzen Sie bitte durch die auf dem Gehäuse aufgedruckte Nummer
 - :8000 bedeutet, dass der Port 8000 angewählt wird



Programmausführung über Webbrowser II

Erstellen Sie eine neue Datei:



Programmausführung über Webbrowser III

Raspberry
Pi

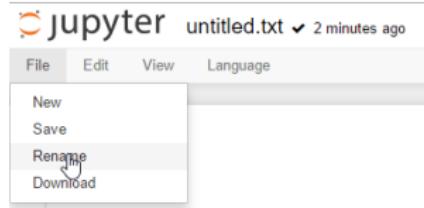
Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

- Benennen Sie die Datei um in hello.py



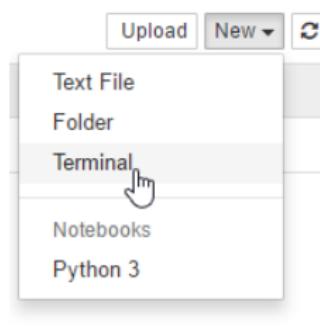
- Schreiben Sie ein einfaches Programm zur Textausgabe
- Speichern Sie mit <strg> + <s> oder im Menü mit File→Save

Programmausführung über Webbrowser IV

- Gehen Sie zurück zur Ausgangsseite (dashboard). Diese erreichen Sie durch ein Klick auf „jupyter“ links oben auf der Seite:



- Wählen Sie anschließend unter dem Punkt „New“ den Punkt „Terminal“:



Programmausführung über Webbrowser V

Raspberry
Pi

Python

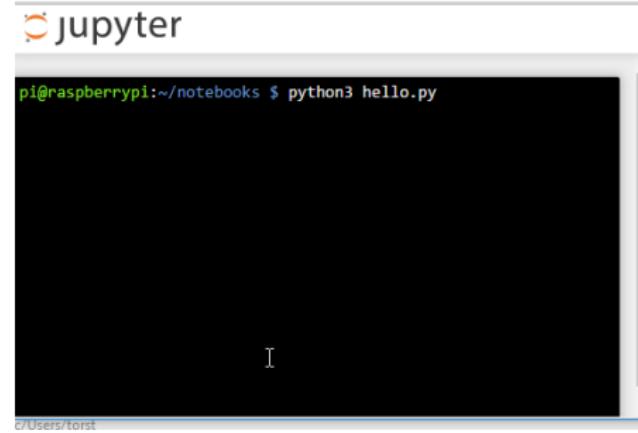
Raspberry
Pi

Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Im nun erscheinenden Terminal-Fenster können Sie das Programm gewohnt ausführen:



The screenshot shows a terminal window titled "jupyter". The title bar has a blue header with the word "jupyter" in white. The main area of the terminal is black, and it displays the following command line text:
pi@raspberrypi:~/notebooks \$ python3 hello.py
I
The cursor is positioned at the end of the line "I". At the bottom of the terminal window, there is a small status bar with the text "C:\Users\torst".

Programmausführung über das Notebook I

Raspberry
Pi

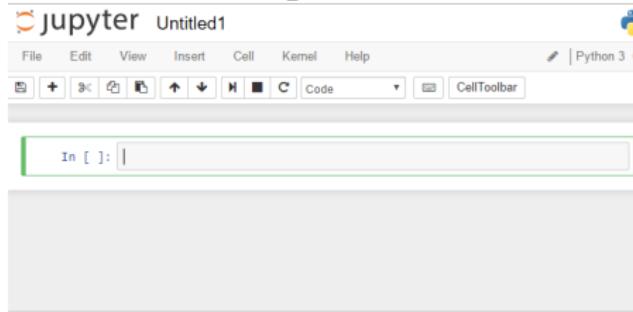
Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Der einfachste Web zur Ausführung von Programmen ist über das Notebook. Sie starten ein Notebook von der Ausgangsseite (dashboard) mit **New → Python3**. Es erscheint folgendes Fenster:



Programmausführung über das Notebook II

- Notebooks sind organisiert in **cells**. Klicken Sie mit der Maus auf eine Zelle erscheint ein grüner Rahmen -

Der Editmode:

```
In [ ]: print("Hallo Notebook")
```

- Im Editmode können Sie die Zelle ausführen mit dem Druck auf **<shift> + <enter>**

```
In [1]: print("Hallo Notebook")
```

Hallo Notebook

```
In [ ]:
```

- Die Ausgabe wird angezeigt und eine neue Zelle im Kommandomodus (blauer Rahmen) erzeugt. Drücken Sie **<enter>**, um in den Editmode zu gelangen

```
In [1]: print("Hallo Notebook")
```

Hallo Notebook

```
In [ ]:
```

Programmausführung über das Notebook III

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Wichtige Hinweise zum Arbeiten mit dem Notebook:

- Im Hintergrund läuft immer der selbe Python-Interpreter.
- Das bedeutet, dass Variablen etc. aus vorhergehenden Programmabläufen vorhanden bleiben.
- Sie können den Interpreter neu starten, wenn Sie aus dem Menü **Kernel -> Restart** wählen.
- Deswegen: Verwenden Sie das Notebook für kleine Tests und schreiben Sie richtige Programme in eine .py-Datei und führen Sie dieses über das Terminal aus.

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

GPIO
SPI

Interaktion

3 Raspberry Pi

4 Python

5 Raspberry Pi Laborübung

6 Anschluss von Peripherie an den RPI

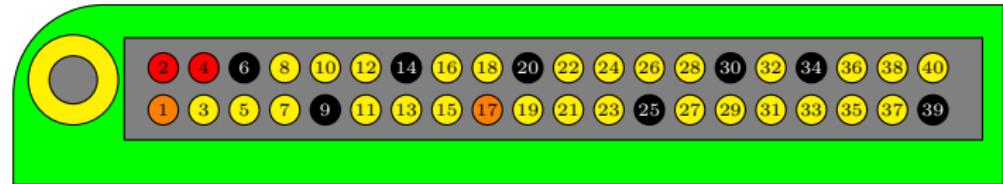
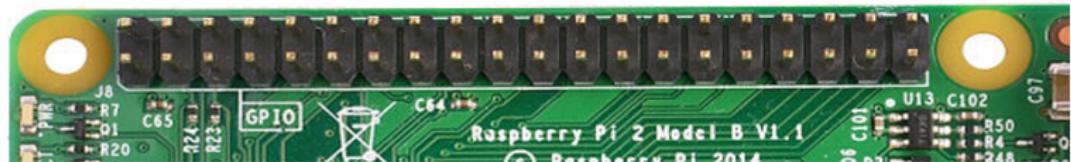
GPIO

SPI

7 Benutzerinteraktion

Anschluss von Peripherie an den RPI

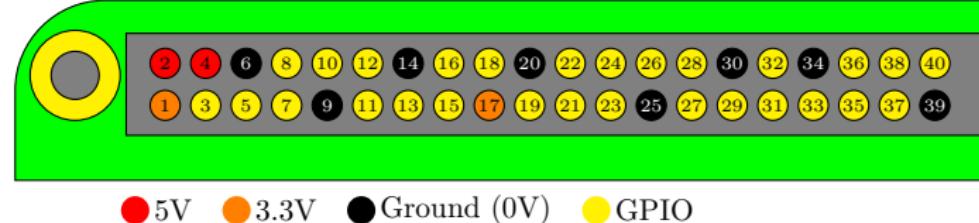
Neben den gängigen Schnittstellen (usb, hdmi, ethernet, etc.) bietet der Raspberry Pi die Möglichkeit gängige Elektronikschnittstellen (SPI, I2C, RS232, GPIO) zu verwenden. Dafür steht ein 40 Pin socket zur Verfügung:



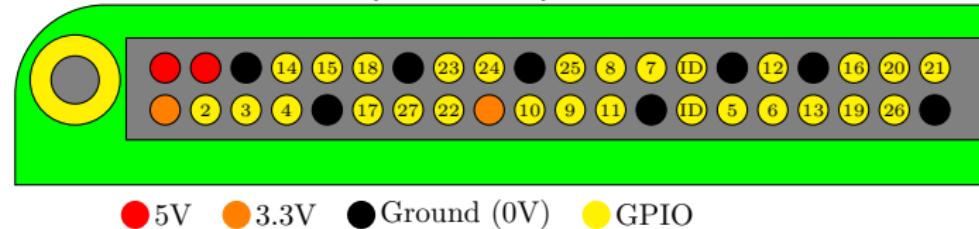
● 5V ● 3.3V ● Ground (0V) ● GPIO

- Auf dem RPI gibt es zwei verschiedene Nummernsysteme für die Programmansteuerung der Stiftleiste:

- BOARD: `GPIO.setmode(GPIO.BOARD)`



- BCM: `GPIO.setmode(GPIO.BCM)`



Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI
GPIO
SPI

Interaktion

- GPIO
 - Ermöglicht pro Pin zwei Zustände (HIGH / LOW).
 - Wird genutzt für einfachste Steueraufgaben (Lampe, Relais AN/AUS etc.)
- SPI
 - Bidirektionale Datenübertragung über drei Leitungen (CLK, MISO, MOSI)
 - Verwendung zur Kommunikation mit integrierten Bausteinen (z. B. AD-Wandler)
 - Nur ein Baustein ansprechbar (Mehr möglich über ChipSelect-Leitungen)
- I2C
 - Bidirektionale Datenübertragung über zwei Leitungen (SDC, SDA)
 - Verwendung zur Kommunikation mit integrierten Bausteinen (z. B. AD-Wandler)
 - Bis zu 127 Bausteine ansprechbar

- Diese Schnittstelle ist eine Möglichkeit mit der Außenwelt Kontakt aufzunehmen.
- 27 Inputs / Outputs
 - Die Richtung wird in Software festgelegt
- Spannungspegel für Ausgänge (Outputs)
 - LOW : 0V
 - HIGH : 3.3V
- Spannungspegel für Eingänge (Inputs)
 - Erlaubter Spannungsbereich (!0...3.3V!)
 - LOW < 0.8V
 - HIGH > 2.0V

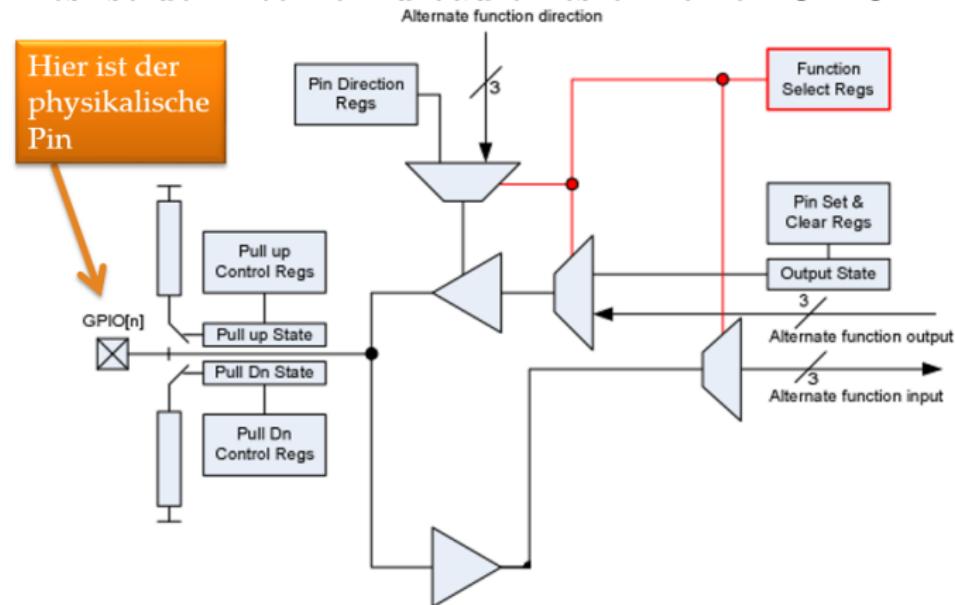
Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPIGPIO
SPI

Interaktion

Dies ist der interne Aufbau eines einzelnen GPIO-Pins:



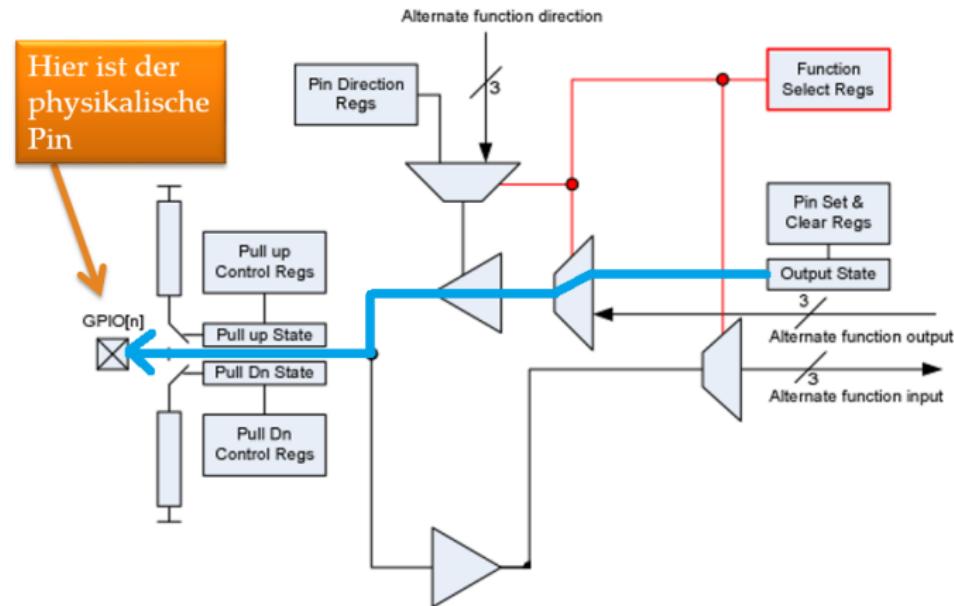
Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPIGPIO
SPI

Interaktion

Beispiel für die Konfiguration als Ausgangspin:



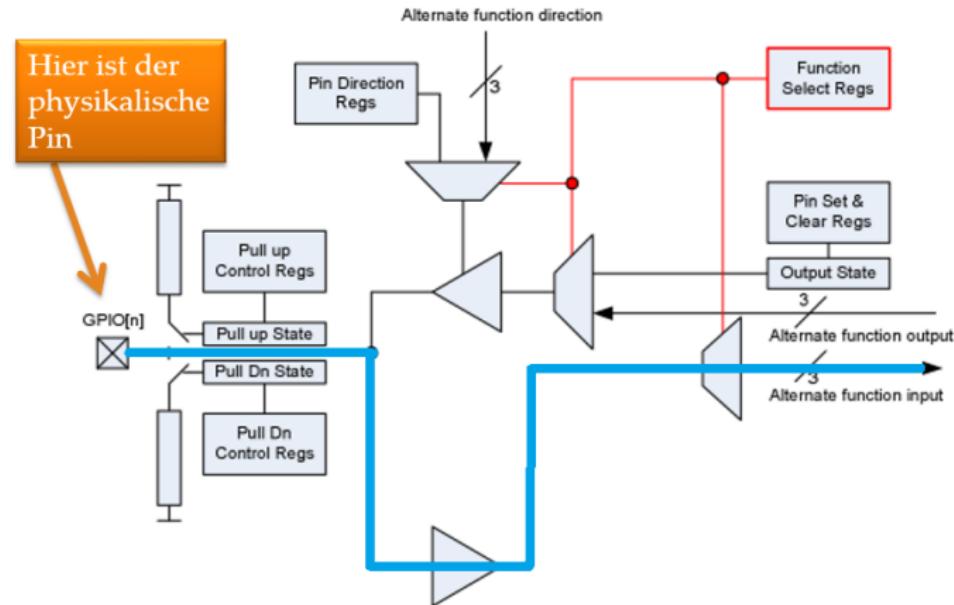
Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPIGPIO
SPI

Interaktion

Beispiel für die Konfiguration als Eingangspin:



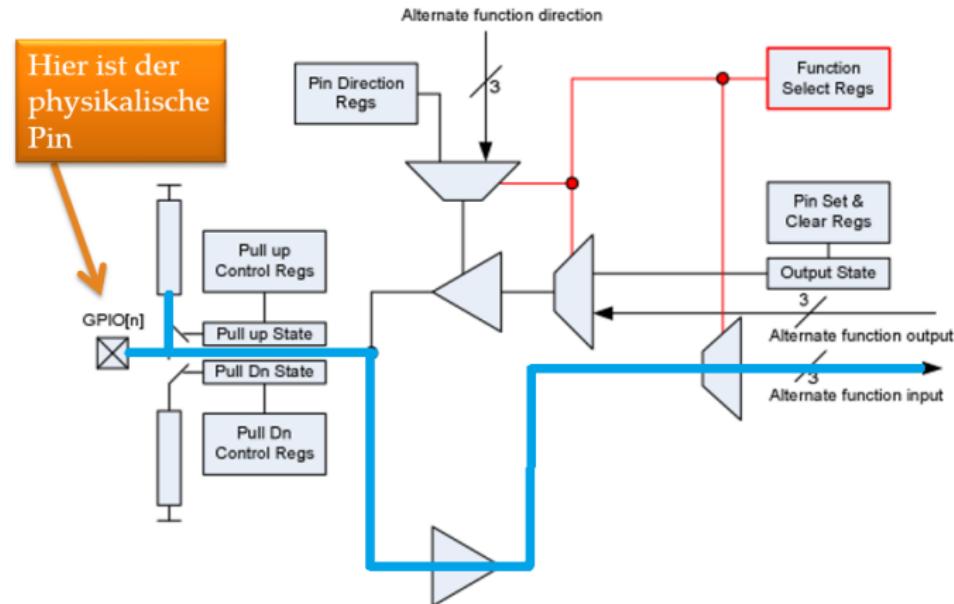
Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPIGPIO
SPI

Interaktion

Beispiel für die Konfiguration als Eingangspin mit Pullup:



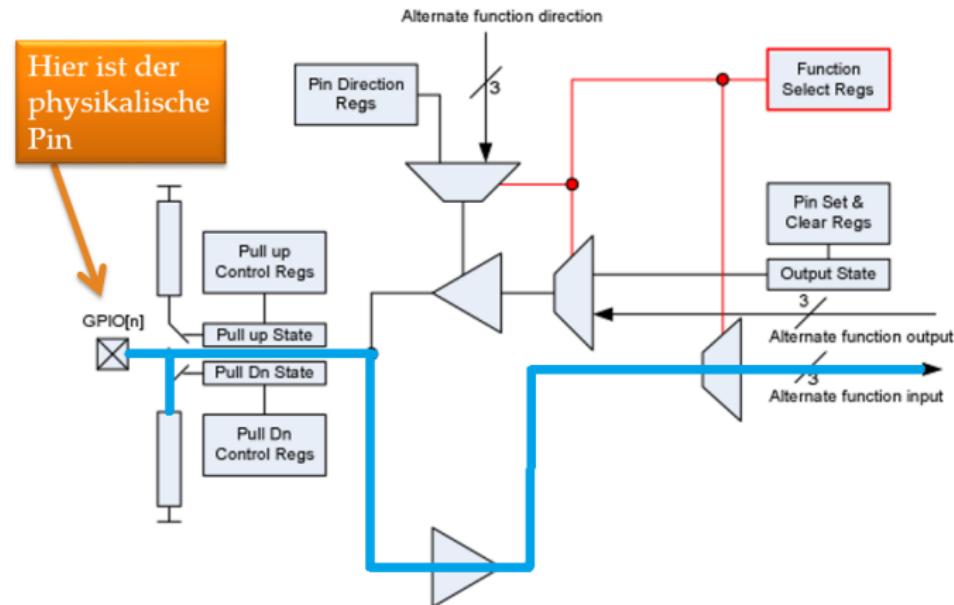
Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPIGPIO
SPI

Interaktion

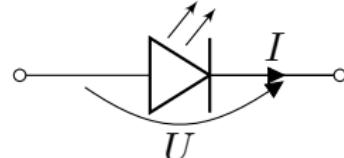
Beispiel für die Konfiguration als Eingangspin mit Pulldown:



Anschluss einer einfachen LED I

Aufgabe: Es soll eine LED mit Hilfe von Softwareanweisungen an und ausgeschaltet werden.

Frage: Wann leuchtet eine LED?



Typische Werte:

rot: 20mA@2.1V
grün: 20mA@2.1V
blau: 20mA@2.9V

Überschreiten Sie NIE die Werte für die Spannung!!!

Verwenden Sie immer einen Vorwiderstand

Anschluss einer einfachen LED (II)

Raspberry
Pi

Python

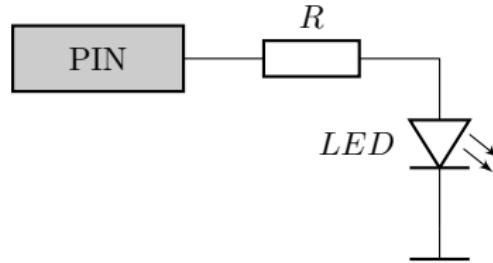
Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

GPIO
SPI

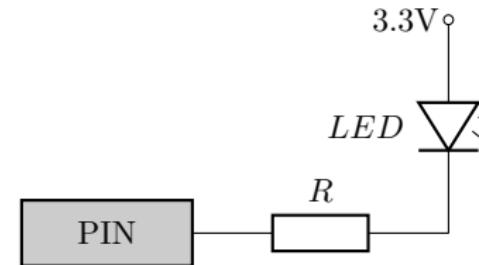
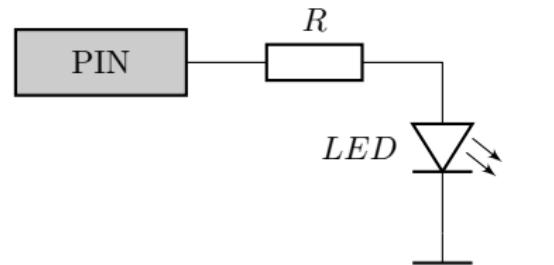
Interaktion

- Beim Anschluss an einen GPIO-Pin des RPIs können zwei Zustände (LOW=0V und HIGH=3.3V) eingestellt werden.
- Wann leuchtet die LED?
- Berechnen Sie den Vorwiderstand für eine LED mit (20mA@2.1V)
- Berechnen Sie den Vorwiderstand für eine LED mit (2mA@2.6V)



Anschluss einer einfachen LED (IV)

- Worin unterscheiden sich die beiden Schaltungen?



Raspberry
Pi

Python

Raspberry
Pi
Laborübung

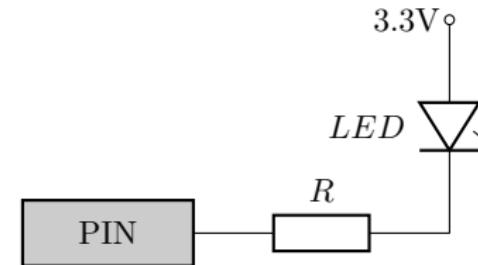
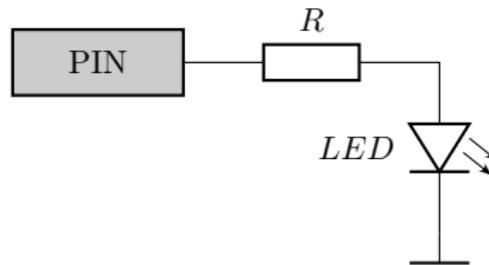
Anschluss
von
Peripherie
an den RPI

GPIO
SPI

Interaktion

Anschluss einer einfachen LED (IV)

- Worin unterscheiden sich die beiden Schaltungen?



Active HIGH

Active LOW

Anschluss einer einfachen LED (V)

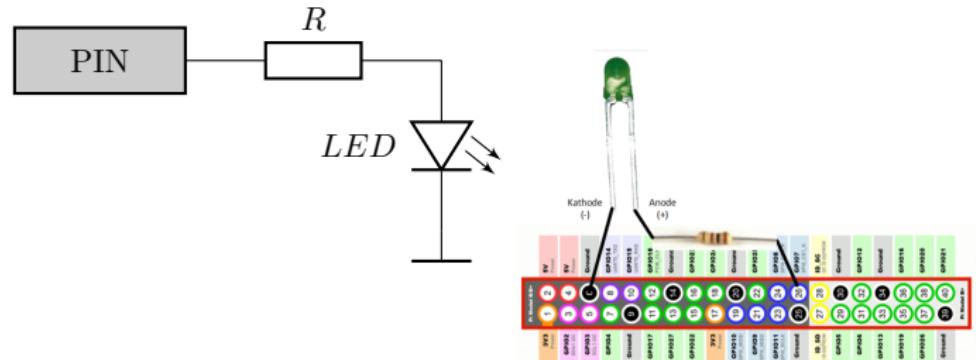
Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI
GPIO
SPI

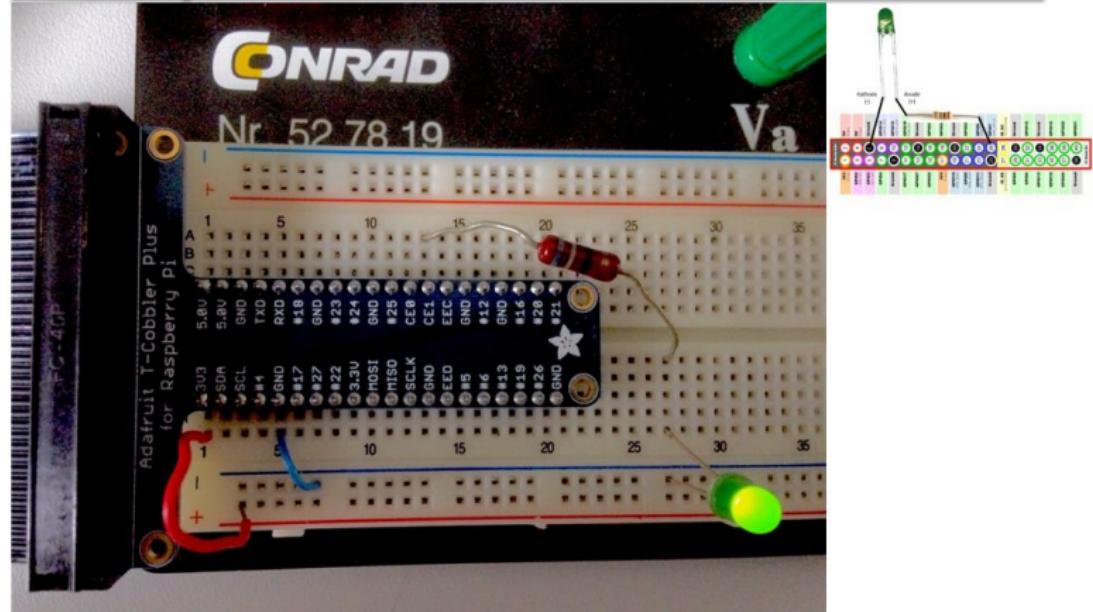
Interaktion

```
1 import RPi.GPIO as GPIO      # GPIO Bibliothek importieren
2 import time                  # Modul time importieren
3
4 GPIO.setmode(GPIO.BOARD)      # Verwende Board-Pinnummern
5 GPIO.setup(26, GPIO.OUT)       # Setze Pin 26 (GPIO7) als
                                Ausgang
6 GPIO.output(26, True)         # Lege 3.3V auf Pin 26
7 time.sleep(0.5)              # Warte 500ms
8 GPIO.output(26, False)        # Lege 0V auf Pin 26
9 GPIO.cleanup()                # Aufräumen
```



Anschluss einer einfachen LED (VI)

Und so sieht es im Labor aus:



Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI
GPIO
SPI

Interaktion

3 Raspberry Pi

4 Python

5 Raspberry Pi Laborübung

6 Anschluss von Peripherie an den RPI

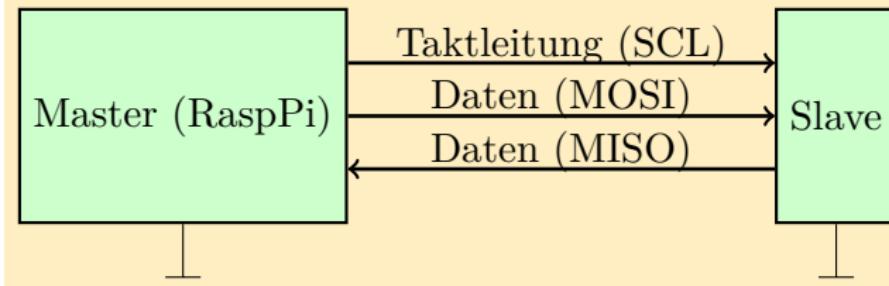
GPIO

SPI

7 Benutzerinteraktion

Grundidee

- Datenübertragung zwischen einfachen elektronischen Komponenten (AD-Wandler / DA-Wandler / Displays / etc.)
- Bidirektional (Send-/Empfangsleitung)
- Seriell (Ein Bit zur Zeit)



Raspberry
Pi

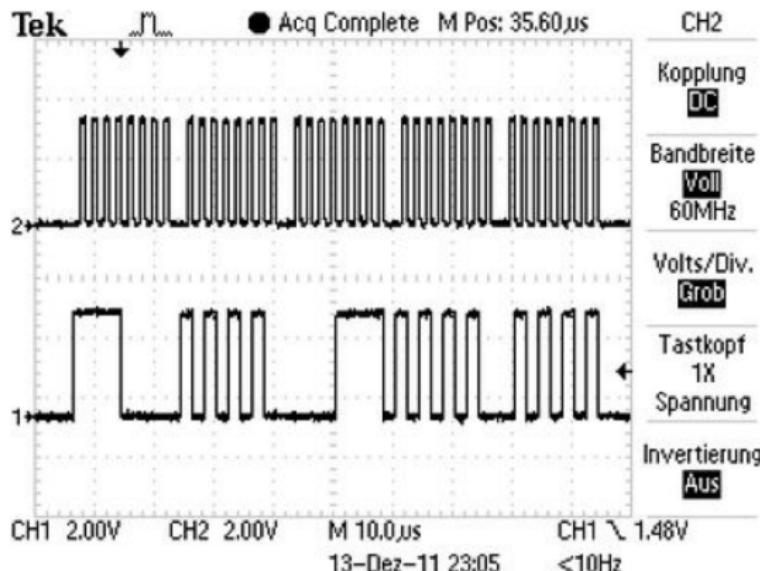
Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI
GPIO
SPI

Interaktion

- Entwickelt von der Firma Motorola
- Kein wirklicher Standard verfügbar
- Es hat sich ein de facto Standard durchgesetzt, der (meistens) eingehalten wird → Datenblatt der Slaves
- Wir besprechen wir das Protokoll, wie es am RaspberryPi umgesetzt wurde.
 - Hinweis: Der Raspberry Pi setzt ein sehr universelles (unspezifisches) SPI-Protokoll ein, damit möglichst viele (vermutlich alle) SPI-fähigen Komponenten angesprochen werden können.

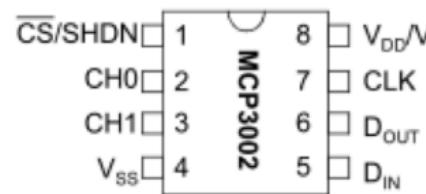
```
1 import spidev
2
3 spi=spidev.SpiDev() #Schnittstellenreferenz
4 spi.open(0, 1)      #Schnittstelle 0 öffnen, Device 1
5 spi.xfer2([0xF0, 0xAA, 0x0F, 0xAA, 0x55]) #Senden
6 spi.close() #Schnittstelle schließen
```



- Senden in 8Bit Paketen
- Dargestellt sind MOSI und SCLK
- Es wird parallel empfangen (Rückgabe von `spi.xfer2`)

Beispiel zum Anschluss eines AD-Wandlers mit SPI-Schnittstelle

- VDD: Versorgungsspannung (2.7V – 5V)
- VSS: Ground (0V)
- CS/SHDN: Chipselect / Shutdown
 - Bei GND → Chipselect
 - Bei VSS → Shutdown
- CLK:SCLK, Dout: MISO, Din: MOSI
- CH0/1: Analoger Eingangskanal 0 oder 1



Raspberry
Pi

Python

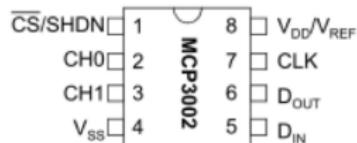
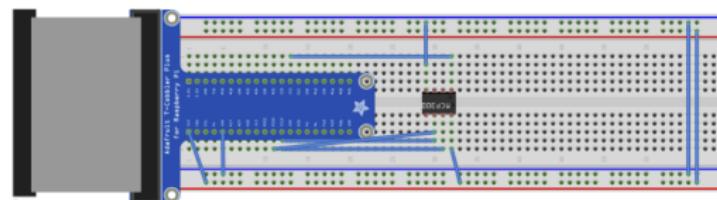
Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI

GPIO

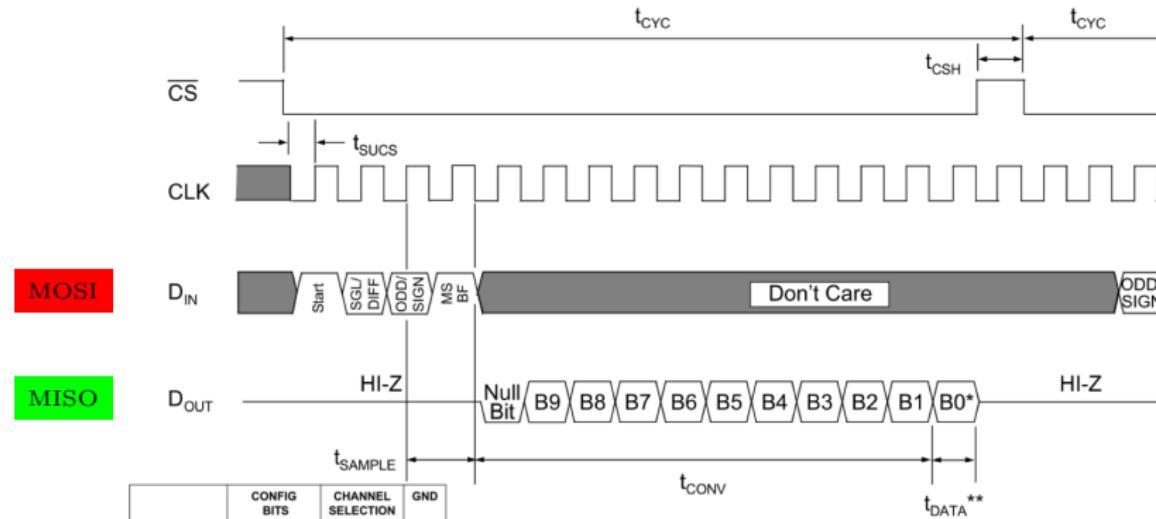
SPI

Interaktion

Schaltung (Erstellt mit Fritzing)

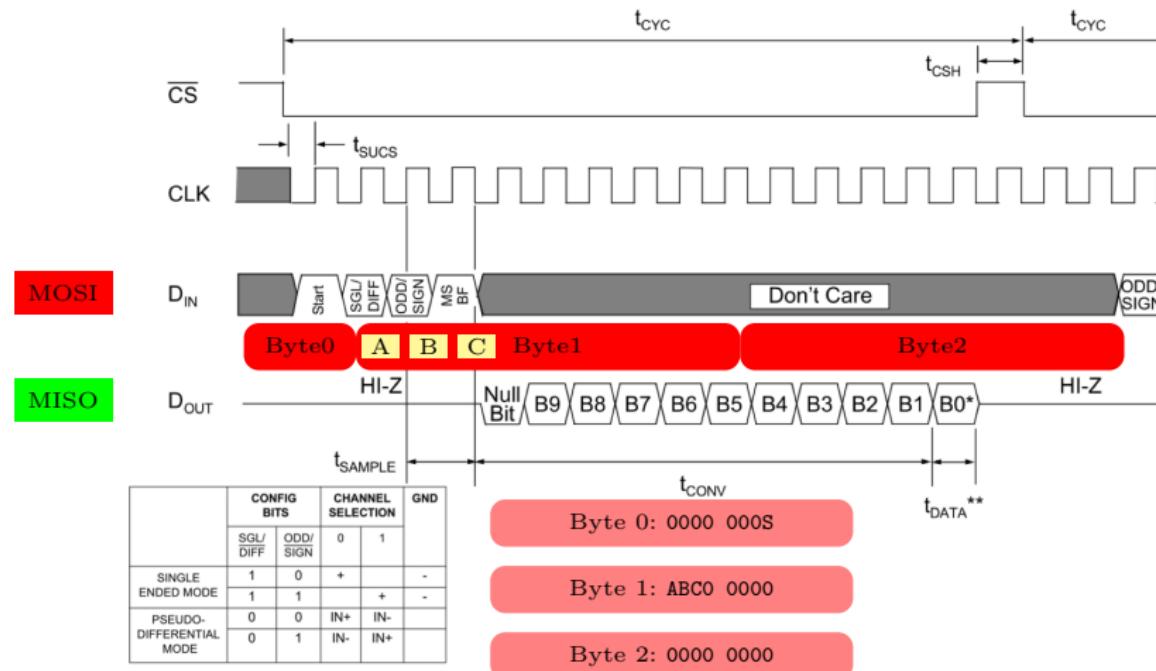


Zeitverlauf laut Datenblatt des MCP3002

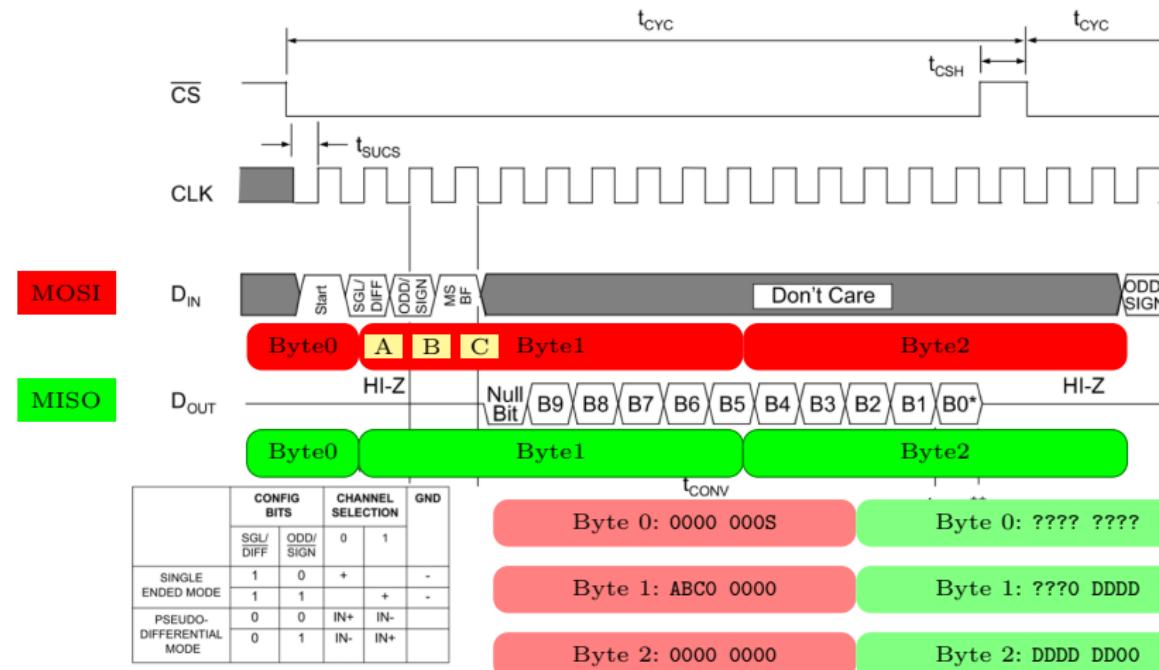


	CONFIG BITS		CHANNEL SELECTION		GND
	SGL/DIFF	ODD/SIGN	0	1	
SINGLE ENDED MODE	1	0	+	-	
	1	1	+	-	
PSEUDO-DIFFERENTIAL MODE	0	0	IN+	IN-	
	0	1	IN-	IN+	

Zeitverlauf laut Datenblatt des MCP3002



Zeitverlauf laut Datenblatt des MCP3002



Software (Schreiben)

Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI
GPIO

SPI

Interaktion

```

1  #-*-coding:utf-8*-
2
3  import spidev
4
5  spi = spidev.SpiDev()
6  spi.open(0, 1)                      #Schnittstellenreferenz
                                         #Schnittstelle 0 öffnen, Device 1
7
8  S = 1                                #Startbit ist immer 1
9  A = 1                                #SGL/DIFF 1=SingleEndedMode
10 B = 0                                 #Kanal 0
11 C = 1                                #MSB First
12
13 Byte0 = S
14 Byte1 = (A << 7) | (B << 6) | (C << 5)    #Bits setzen
15 Byte2 = 0
16 print('0x%02X 0x%02X 0x%02X' % (Byte0, Byte1, Byte2))
17 data = spi.xfer2([Byte0, Byte1, Byte0])    #Senden und empfangen
18 print(data)                            #Daten ausgeben
19 spi.close()                           #Schnittstelle schließen

```

	CONFIG BITS		CHANNEL SELECTION		GND
	SGL/ DIFF	ODD/ SIGN	0	1	
SINGLE ENDED MODE	1	0	+		-
	1	1		+	-
PSEUDO- DIFFERENTIAL MODE	0	0	IN+	IN-	
	0	1	IN-	IN+	

A B

Byte 0: 0000 000S

Byte 1: ABC0 0000

Byte 0: 0000 0000

Software (Lesen)

Raspberry
Pi

Python

Raspberry
Pi

Laborübung

Anschluss
von
Peripherie
an den RPI
GPIO
SPI

Interaktion

Hier werden die
empfangenen
Daten dekodiert.

```
1  #-*-coding:utf-8-*-  
2  
3  import spidev  
4  
5  spi = spidev.SpiDev()  
6  spi.open(0, 1)                      #Schnittstellenreferenz  
                                         #Schnittstelle 0 öffnen, Device 1  
7  
8  S = 1                                #Startbit ist immer 1  
9  A = 1                                #SGL/DIFF 1=SingleEndedMode  
10 B = 0                                 #Kanal 0  
11 C = 1                                #MSB First  
12  
13 Byte0 = S  
14 Byte1 = (A << 7) | (B << 6) | (C << 5)    #Bits setzen  
15 Byte2 = 0  
16 print('Gesendet: 0x%02X 0x%02X 0x%02X' % (Byte0, Byte1, Byte2))  
17 data = spi.xfer2([Byte0, Byte1, Byte0])    #Senden und empfangen  
                                         #Verwende bits0-3 von Byte1  
wert = data[1] & 0x0F  
wert = wert << 6                         #Schiebe diese 6 bits nach rechts  
wert = wert + data[2]>>2                  #Verwende die Bits 2-8 von Byte2  
                                         #Schiebe 2 bits nach rechts und addiere  
                                         #Schnittstelle schließen  
21  
22 spi.close()  
23  
24 print('Empfangen: %d' % wert)
```

Byte 0: UUUU UUUU

Byte 1: UUU0 DDDD

Byte 0: DDDD DD00

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe
Eingabe

③ Raspberry Pi

④ Python

⑤ Raspberry Pi Laborübung

⑥ Anschluss von Peripherie an den RPI

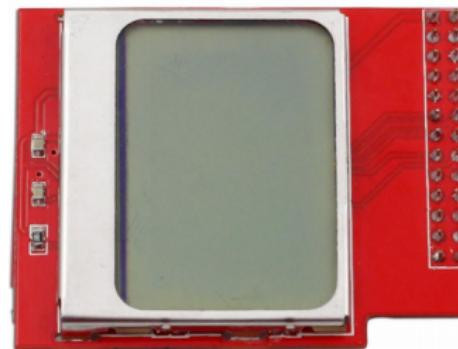
⑦ Benutzerinteraktion

Ausgabe

Eingabe

Anschluss eines Displays

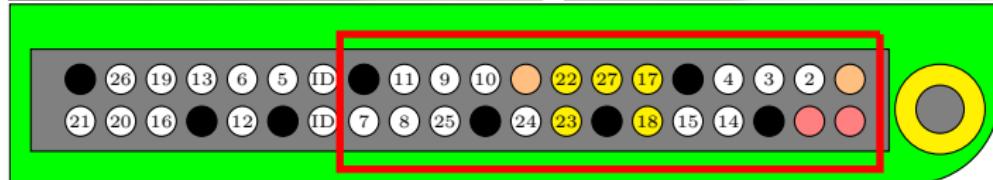
- Benutzerinteraktion benötigt Anzeigeelemente
- Im Labor können Displays verwendet werden
- Diese sitzen auf Huckepack-Boards und verwenden alte Nokia-Displays.



CLK	---	pin11	---	GPIO0 (17)
Din	---	pin12	---	GPIO1 (18)
D/C	---	pin13	---	GPIO2 (27)
CS	---	pin15	---	GPIO3 (22)
RST	---	pin16	---	GPIO4 (23)

Display am Raspberry Pi

- Die Displays sind aufsteckbar (Huckepack)



Es werden die GPIO-Pins 17,18,27,22,23 verwendet (BCM-Modus).

Raspberry
Pi

Python

Raspberry
Pi

Laborübung

6

Anschluss

von

Peripherie

an den RPI

Interaktion

Ausgabe

Eingabe

```
1 import time
2 from PIL import Image
3 from PIL import ImageFont
4 from PIL import ImageDraw
5 import Adafruit_Nokia_LCD as LCD
6
7 SCLK = 17; DIN = 18; DC = 27
8 RST = 23; CS = 22
9
10 def placetext(image, text):
11     font = ImageFont.load_default()
12     draw = ImageDraw.Draw(image)
13     draw.rectangle((0, 0, 83, 47), outline=255, fill
14 =255)
15
16     for i, c in enumerate(text):
17         draw.text((10 + i * 8, 10), c, font=font, fill
18 =0)
19
20     disp.image(image)
21     disp.display()
```

```
21 disp = LCD.PCD8544(DC, RST, SCLK, DIN, CS)
22 disp.begin(contrast=60)
23 image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))
24
25 for i in range(50):
26     placetext(image, 'hallo %d' % i)
27     time.sleep(0.1)
```

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe

Eingabe

Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe
Eingabe

3 Raspberry Pi

4 Python

5 Raspberry Pi Laborübung

6 Anschluss von Peripherie an den RPI

7 Benutzerinteraktion

Ausgabe

Eingabe

Raspberry
Pi

Python

Raspberry
Pi

Laborübung

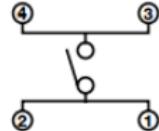
Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe

Eingabe

- Computersysteme benötigen immer eine Dateneingabe
- Dies kann auf vielfältigste Weise geschehen:
 - Netzwerkdaten, Tastatureingaben, Mausaktionen
 - Dateien, etc.
- Für ein eigenständiges Gerät bieten sich an:
 - Touchdisplay
 - Taster, Drehencoder etc.



Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe

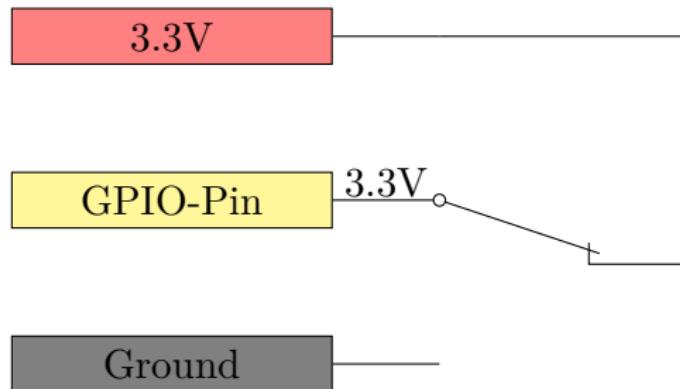
Eingabe

- Taster besitzen zwei Zustände
- GPIO-Pins können zwei Zustände „lesen“
- Wie bringt man beides zusammen?
- Ein GPIO-Pin im Eingangsmodus erkennt
 - Eine Null bei Spannungen von 0V-0.8V
 - Eine Eins bei Spannungen von 2.3V-3.3V
 - **Spannungen > 3.3V zerstören den RPI!**



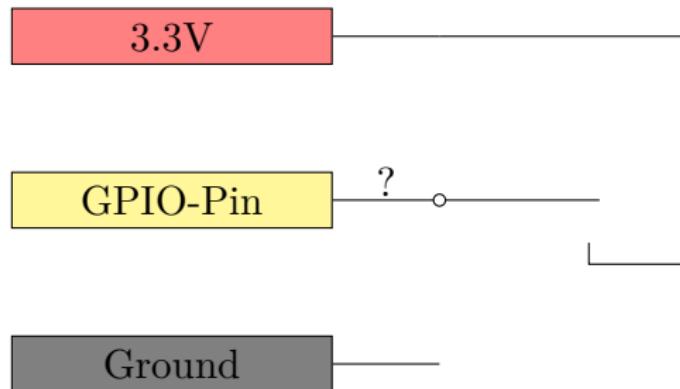
Eingaben durch Taster (II)

- Verwendung eines Tasters
- Direkte Verbindung mit 3.3V → Logisch 1



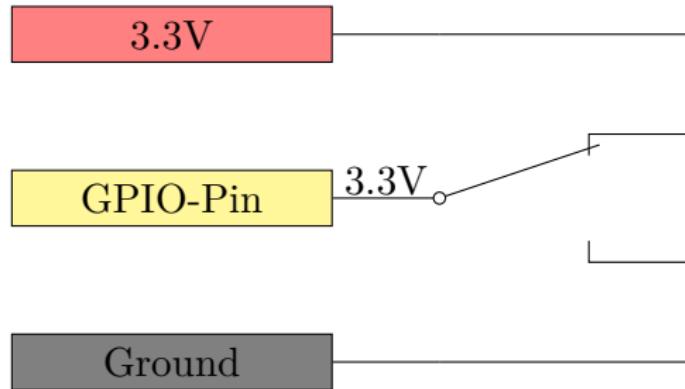
Eingaben durch Taster (II)

- Verwendung eines Tasters
- Offener Eingangsport → undefinierter Zustand



Eingaben durch Taster (III)

- Verwendung eines Wechseltasters
- Direkte Verbindung mit 3.3V → Logisch 1



Raspberry
Pi

Python

Raspberry
Pi
Laborübung

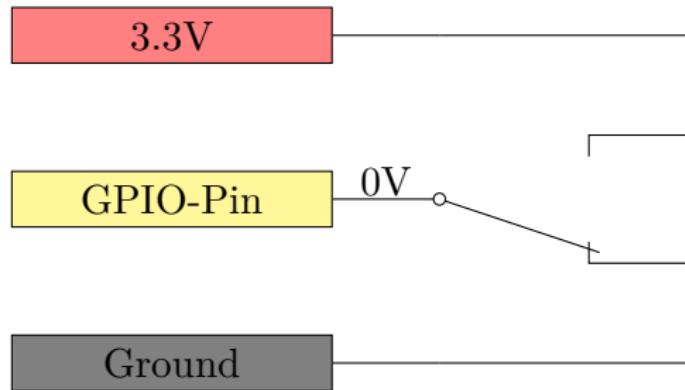
Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe
Eingabe

Eingaben durch Taster (III)

- Verwendung eines Wechseltasters
- Direkte Verbindung mit 0V → Logisch 0



Raspberry
Pi

Python

Raspberry
Pi
Laborübung

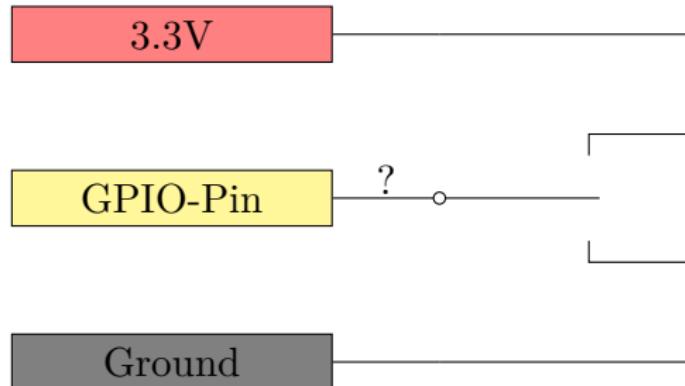
Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe
Eingabe

Eingaben durch Taster (III)

- Verwendung eines Wechseltasters
- Offener Eingangsport → undefinierter Zustand



Raspberry
Pi

Python

Raspberry
Pi
Laborübung

Anschluss
von
Peripherie
an den RPI

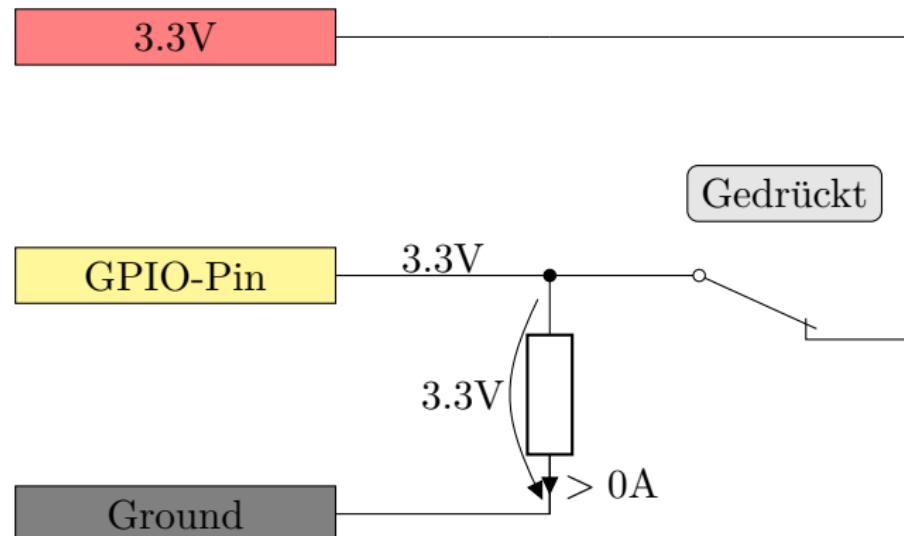
Interaktion

Ausgabe
Eingabe

Eingaben durch Taster (IV)

Raspberry Pi
Python
Raspberry Pi
Laborübung
Anschluss von Peripherie an den RPI
Interaktion
Ausgabe
Eingabe

- Verwendung eines Pulldown-Widerstand
- Direkte Verbindung mit 3.3V → Logisch 1. Durch den Widerstand fließt ein Strom.



Eingaben durch Taster (IV)

Raspberry
Pi

Python

Raspberry
Pi

Laborübung

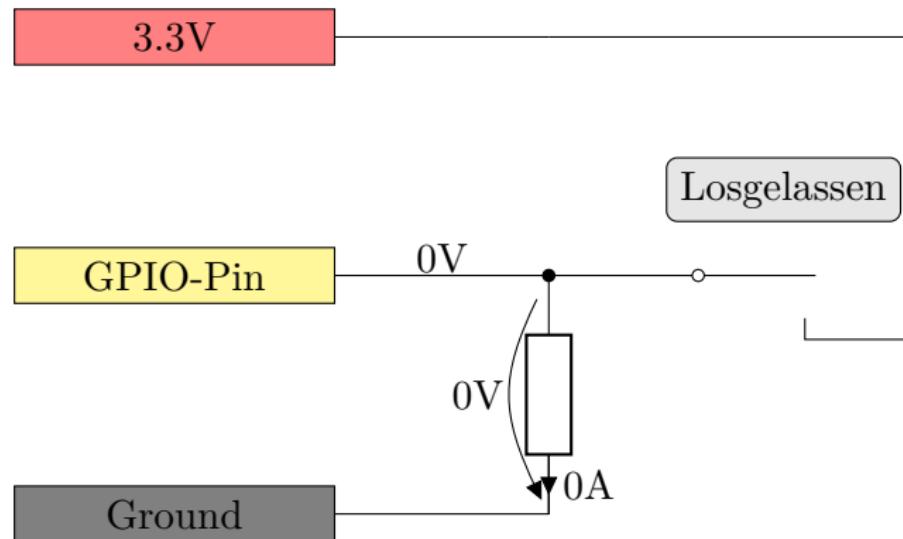
Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe

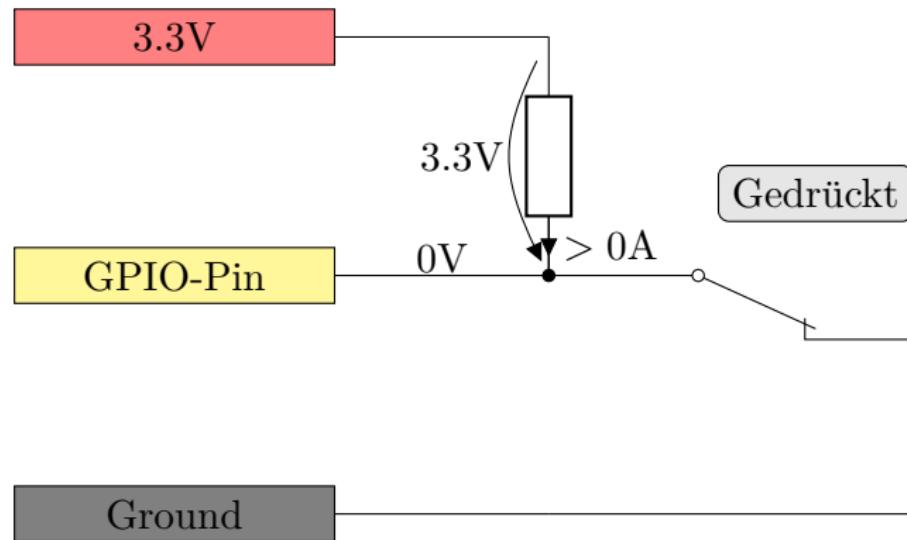
Eingabe

- Verwendung eines Pulldown-Widerstand
- Weiterhin Verbindung über Widerstand zu Ground → Logisch 0. Durch den Widerstand fließt kein Strom.



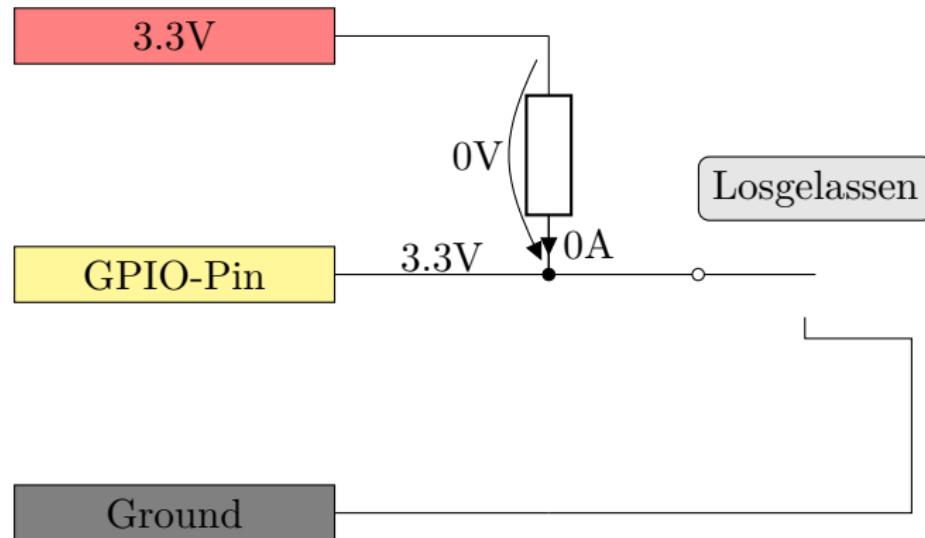
Eingaben durch Taster (V)

- Verwendung eines Pullup-Widerstand
- Direkte Verbindung mit Ground (0V) → Logisch 0.
Durch den Widerstand fließt ein Strom.



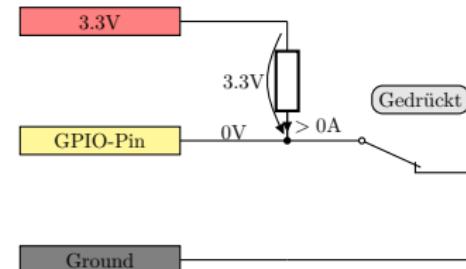
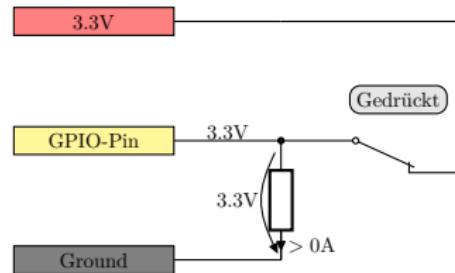
Eingaben durch Taster (V)

- Verwendung eines Pullup-Widerstand
- Weiterhin Verbindung über Widerstand zu 3.3V → Logisch 1. Durch den Widerstand fließt kein Strom.



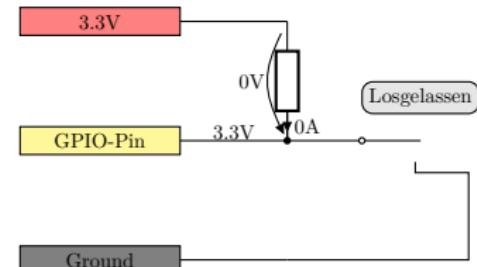
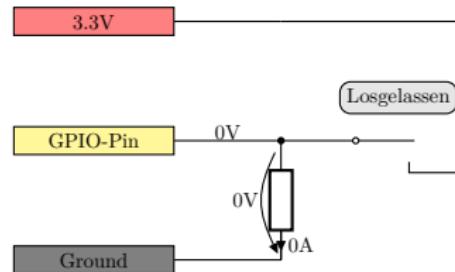
Eingaben durch Taster (VI)

- Verwendung von Pullup- oder Pulldown-Widerstand unterscheiden sich in der Polarität des Auslesens.
- In der Regel liegt der Widerstandswert zwischen $1k\Omega$ und $10k\Omega$.



Eingaben durch Taster (VI)

- Verwendung von Pullup- oder Pulldown-Widerstand unterscheiden sich in der Polarität des Auslesens.
- In der Regel liegt der Widerstandswert zwischen $1k\Omega$ und $10k\Omega$.

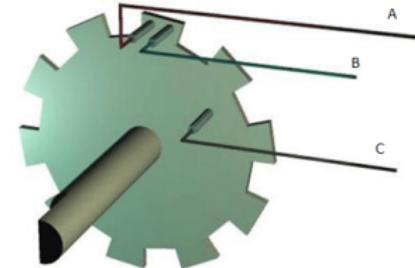
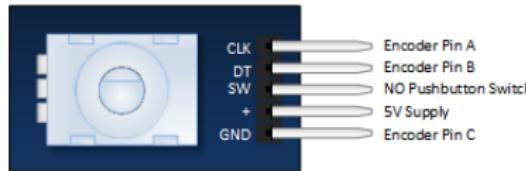


Eingabe durch Drehencoder

- Drehencoder³ nutzt man für Drehrichtungserkennung.



- Platine im Labor und Funktionsprinzip



³Sehr gute Funktionsbeschreibung und Quelle:
<http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/keyes-ky-040-arduino-rotary-encoder-user-manual/>

Eingabe durch Drehencoder (II)

Raspberry
Pi

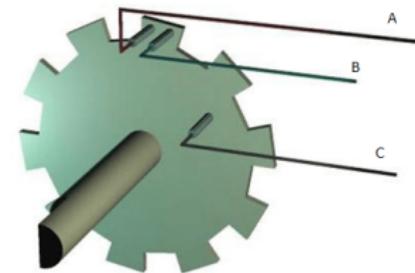
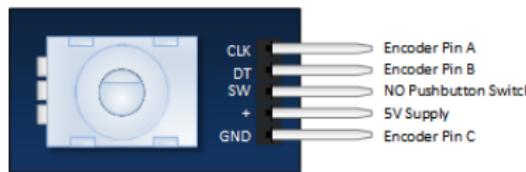
Python

Raspberry
Pi
Laborübung

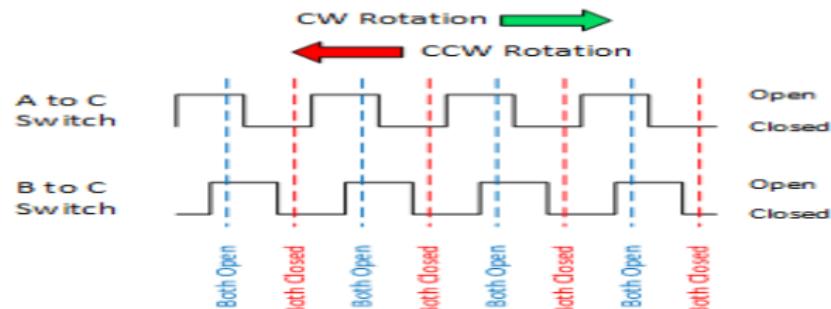
Anschluss
von
Peripherie
an den RPI

Interaktion
Ausgabe
Eingabe

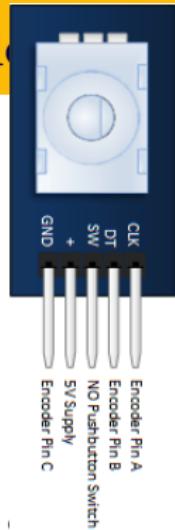
- Platine im Labor und Funktionsprinzip



- Beim Drehen entsteht entsprechend der Drehrichtung ein Muster:



Eingabe durch Drehencoder



- Die Laborplatine hat folgende Anschlüsse:
 - CLK: PinA (Mit GPIO verbinden)
 - DT : PinB (Mit GPIO verbinden)
 - SW : Taster (Mit GPIO verbinden)
 - +: Versorgungsspannung für interne Pullup-Widerstände (Mit 3.3V verbinden)
 - GND: Masse
- Es folgt ein Beispielcode und Anschlussplan für Auslesen des Encoder (PinA → GPIO20 und PinB → GPIO21 (BCM-Mode))

Eingabe durch Drehencoder (IV)

Raspberry
Pi

Py

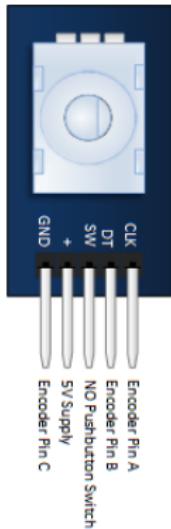
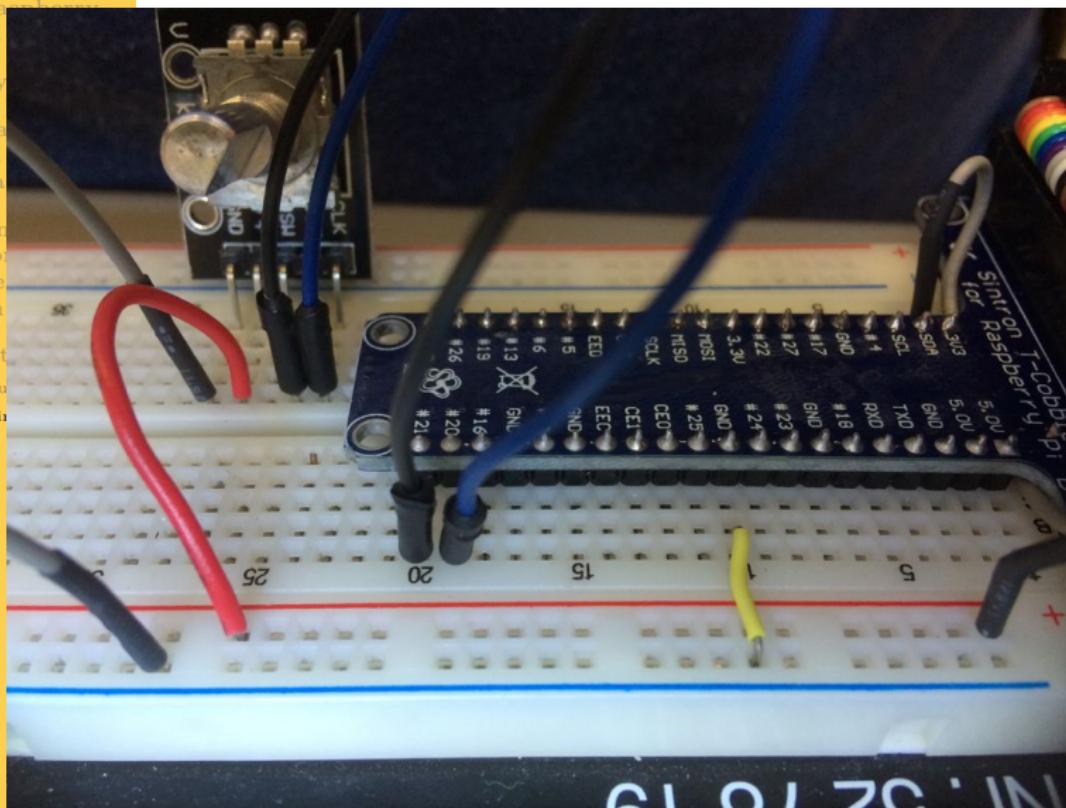
Ra
Pi
La

An
vo
Pe
an

Int

Au

Ein



Eingabe durch Drehencoder – Programmbeispiel

I

Raspberry
Pi

Python

Raspberry
Pi
LaborübungAnschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe

Eingabe

```
1 import RPi.GPIO as GPIO
2 import time
3
4 class rotary:
5     def __init__(self, pinA, pinB):
6         self.state = "idle" #Zustandsvariable
7         self.pinA = pinA    #GPIO Nummer von PinA
8         self.pinB = pinB    #GPIO Nummer von PinB
9     def update(self):
10        pinstate = self.poll()
11        result = 0
12
13     #Implementierung des Zustandsautomaten
14     #      <CWD> --> <CW2> --> <CW3> ----->|<-->
15     # <idle>                                <-->
16     #      <CCWD> --> <CCW2> --> <CCW3> ->|<-->
17     if self.state == 'idle':
18         transition = { (0,0): 'idle',
19                         (0,1): 'CW1',
20                         (1,0): 'CCW1',
21                         (1,1): 'idle'}
```

Eingabe durch Drehencoder – Programmbeispiel

II

Raspberry
Pi
Python
Raspberry
Pi
Laborübung
Anschluss
von
Peripherie
an den RPI
Interaktion
Ausgabe
Eingabe

```
22         self.state = transition[pinstate]
23 elif self.state == 'CW1':
24     transition = { (0,0): 'CW2',
25                   (0,1): 'CW1',
26                   (1,0): 'idle',
27                   (1,1): 'idle' }
28         self.state = transition[pinstate]
29 elif self.state == 'CW2':
30     transition = { (0,0): 'CW2',
31                   (0,1): 'idle',
32                   (1,0): 'CW3',
33                   (1,1): 'idle' }
34         self.state = transition[pinstate]
35 elif self.state == 'CW3':
36     transition = { (0,0): 'idle',
37                   (0,1): 'idle',
38                   (1,0): 'CW3',
39                   (1,1): 'idle' }
40         self.state = transition[pinstate]
41 if pinstate == (1,1):
42     result = 1
```

Eingabe durch Drehencoder – Programmbeispiel

III

Raspberry
Pi

```
43     elif self.state == 'CCW1':  
44         transition = { (0,0): 'CCW2',  
45                           (0,1): 'idle',  
46                           (1,0): 'CCW1',  
47                           (1,1): 'idle' }  
48         self.state = transition[pinstate]  
49     elif self.state == 'CCW2':  
50         transition = { (0,0): 'CCW2',  
51                           (0,1): 'CCW3',  
52                           (1,0): 'idle',  
53                           (1,1): 'idle' }  
54         self.state = transition[pinstate]  
55     elif self.state == 'CCW3':  
56         transition = { (0,0): 'idle',  
57                           (0,1): 'CCW3',  
58                           (1,0): 'idle',  
59                           (1,1): 'idle' }  
60         self.state = transition[pinstate]  
61     if pinstate == (1,1):  
62         result = -1  
63 return result
```

Python

Raspberry

Pi
LaborübungAnschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe

Eingabe

Eingabe durch Drehencoder – Programmbeispiel

IV

Raspberry
Pi

64

Python

65

Raspberry
Pi

66

Laborübung

67

Anschluss
von
Peripherie
an den RPI

68

Interaktion

69

Ausgabe

70

Eingabe

71

```
#poll() gibt den Zustand beider Pins als Tuple aus
#(pinA, pinB). Dabei wird mehrmals gelesen und
#erst wenn <debounce>-mal der gleiche
#Zustand gelesen wurde, wird dieser zurückgegeben.
def poll(self, debounce=20):
    state = (-1,-1)
    stable_cnt = 0
    while(stable_cnt <= debounce):
        a = int(GPIO.input(self.pinA))
        b = int(GPIO.input(self.pinB))
        newstate = (a,b)
        if newstate != state:
            stable_cnt = 0
            state = newstate
        else:
            stable_cnt += 1
    return state
```

82

83

84

Eingabe durch Drehencoder – Programmbeispiel

V

Raspberry
Pi

```
85 def setup(pinA, pinB):
86     #BCM Nummerierungsmodus
87     GPIO.setmode(GPIO.BCM)
88     GPIO.setwarnings(False)
89     #PinA und PinB als Eingänge parametrieren
90     GPIO.setup(pinA, GPIO.IN)
91     GPIO.setup(pinB, GPIO.IN)

92 # ----- Programmstart -----
93 pinA = 20; pinB = 21

94 setup(pinA, pinB)
95 rot = rotary(pinA, pinB)
96 s = 0
97 while(1):
98     time.sleep(0.001)    #Dies dient nur zur Reduzierung
99                         #der Prozessorlast
100    s += rot.update()  #Rot Update gibt -1,0,+1 zurück
101                         #entspricht (links, still, rechts)
102    print(s)           #Ausgabe der Zählvariable
```

Python

Raspberry
Pi

Laborübung

Anschluss
von
Peripherie
an den RPI

Interaktion

Ausgabe

Eingabe