Hochschule
Augsburg University of
Applied Sciences

RaspberryPi

-

Eine kurze Einführung

Michael Schäferling

Efficient Embedded Systems
University of Applied Sciences Augsburg
Department of Computer Science

Übersicht:

Hochschule Augsburg University of Applied Sciences

Was ist ein “Embedded System”?

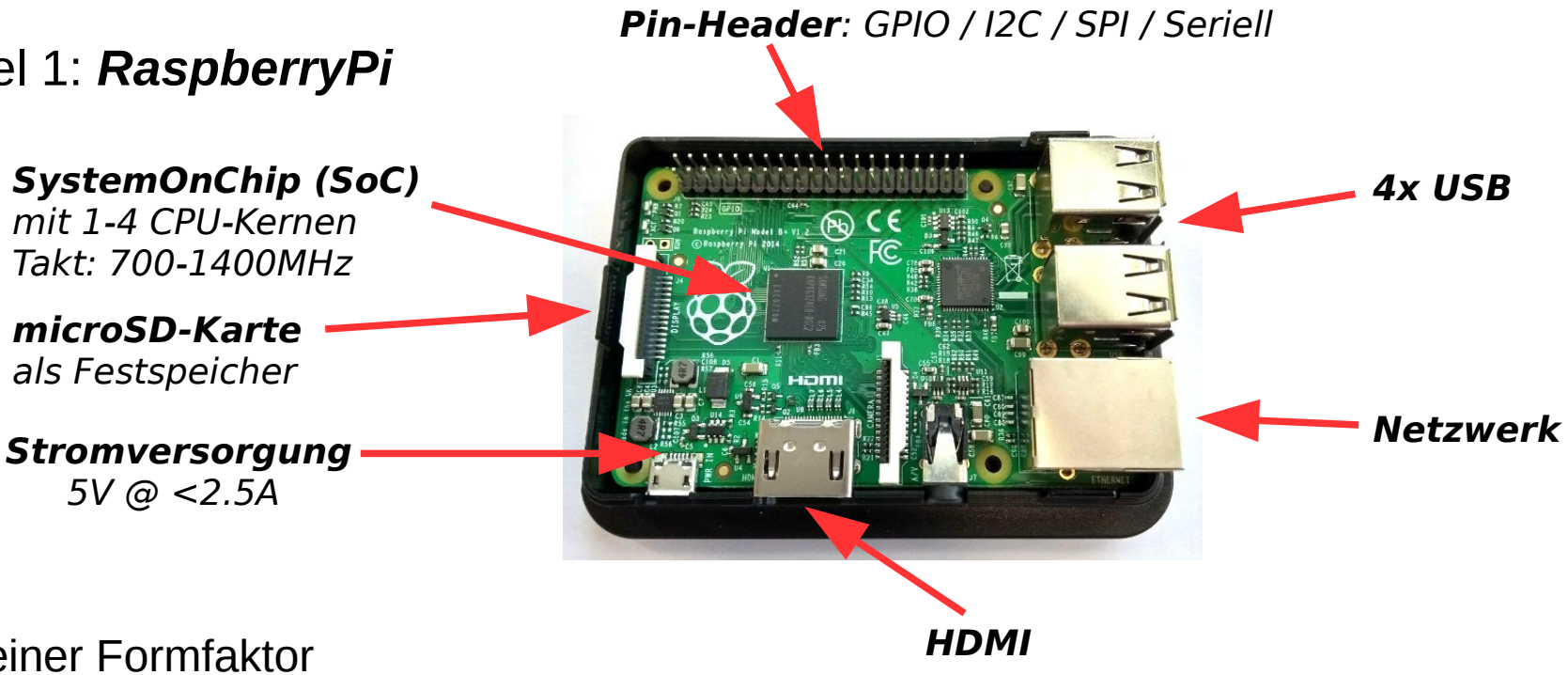
- Grenze zwischen Standard-Rechnerplattformen und Embedded-Systems ist nicht klar definiert

- Wesentliche Merkmale eines “Embedded Systems”:
  - Kleiner Formfaktor
  - Im Allgemeinen weniger leistungsstark als ein aktueller Desktop-PC
  - Geringer Stromverbrauch: < 5Watt
  - Oft keine grafische Benutzeroberfläche

→ Meist auf bestimmten Einsatzzweck hin optimiert

Was ist ein "Embedded System"?

- Beispiel 1: ***RaspberryPi***



  - Kleiner Formfaktor
  - Im Allgemeinen weniger leistungsstark als ein aktueller Desktop-PC
  - Geringer Stromverbrauch: < 10Watt
  - Mit oder ohne grafischer Benutzeroberfläche

Was ist ein "Embedded System"?

- Beispiel 2: ***Arduino***

***Pin-Header**: digital / analog / I2C / seriell*

USB-Anschluss
Programmierung und Spannungsversorgung

Stromversorgung

Mikrocontroller
Takt < 20MHz

- Kleiner Formfaktor
- viel weniger leistungsstark als ein aktueller Desktop-PC
- Sehr geringer Stromverbrauch: << 1Watt
- Keine grafische Benutzeroberfläche

Wie wird Software in einem Embedded System ausgeführt?

- mit Betriebssystem (OS), z.B.:
  - Linux (z.B. Debian, Ubuntu, …)
  - Windows (Windows Embedded)
  - Echtzeit-Betriebssystem (z.B. FreeRTOS, VxWorks)
- ohne Betriebssystem: “Bare Metal” (bzw. in gekapselter Umgebung)
  - Nur ein Programm wird ausgeführt
  - I/O-Ressourcen, Speicher etc. müssen selbst verwaltet werden
  - Keine störenden Einflüsse (durch OS oder andere Programme)

Übersicht:

Hochschule Augsburg University of Applied Sciences

Woher bekommt man das Betriebssystem?

- Selbst “bauen”: oft nur für Spezialanwendungen benötigt, sehr aufwändig, nur möglich wenn Quellcode verfügbar

- Ein fertiges Abbild (“Image”) beschaffen:
  - Wird üblicherweise vom Hersteller des Systems/Boards angeboten
  - Das Betriebssystem ist speziell auf das Board und Komponenten angepasst

    → Deshalb meist nur für ein spezielles Board geeignet

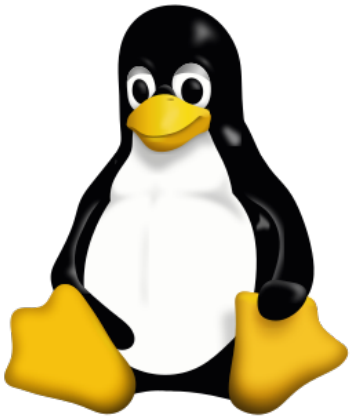
  - Download-Webseite für den RaspberryPi:

    https://www.raspberrypi.org/downloads/

→ *Die “Qual der Wahl”: Welches Linux ist das beste?*

Was ist "das Betriebssystem Linux"?

- "Linux" ist eigentlich nur der so genannte ***Kernel***, dieser kümmert sich um die grundlegenden Aufgaben wie die Ressourcen-Verwaltung des Systems, z.B.:
  - Zuweisung von CPU-Zeit an Programme
  - Zugriff auf Peripherie (über Treiber)
  - Bereitstellung von Arbeitsspeicher
  - Zugriff auf die Festplatte/Dateisystem



"Tux"

- Alles andere, wie die Benutzeroberfläche, Programme (z.B. "LibreOffice", Bildbearbeitung, …), ist das so genannte "***User-Land***"

- Eine ***Distribution*** (Debian, Ubuntu, Raspbian, etc.) stellt den *Kernel* und das *User-Land* bereit. Zudem ist meist ein ***Paket-Management*** integriert (einfaches Installieren, Updaten und Entfernen von Programm-Paketen)

Übersicht:

Hochschule Augsburg University of Applied Sciences

Inbetriebnahme des Systems

Hochschule Augsburg University of Applied Sciences

1. Vorbereiten des Betriebssystem-Mediums (Flashen eines Images)

- RaspberryPi und viele ähnliche Boards:
  - zentrales Speichermedium ist eine (micro-)SD-Karte
  - Diese muss mindestens so groß sein wie das entpackte Image (.img)
  - **Achtung: Beim Flashen droht Datenverlust wenn das falsche Ziel-Gerät angegeben wird!!!**
- Unter Linux:
  - Kopieren des Abbilds mit dem Programm “dd” (als ‘root’):

```
unzip 2018-03-13-raspbian-stretch.zip

sudo dd if=2018-03-13-raspbian-stretch.img of=/dev/mmcblk0
```

- Unter Windows:
  - Kopieren des Abbilds mit “Win32DiskImager”
- Weitere Informationen:
  https://www.raspberrypi.org/documentation/installation/installing-images/README.md

Inbetriebnahme des Systems

Hochschule Augsburg University of Applied Sciences

2. Starten des Systems

- Einsetzen der (micro-)SD-Karte ins Board
- Monitor, Tastatur und Maus anschließen
- Strom anschließen

3. Zugriff auf das System:

- Wenn Monitor und Tastatur/Maus vorhanden und angeschlossen
  oder beim 1. System-Start (da hier leider noch kein Netzwerkzugriff möglich):

  → Grafische Oberfläche

- Wenn das System (RaspberryPi) z.B. in einem Gerät verbaut ist,
  HDMI-/USB-Ports deshalb nicht benutzt werden können
  oder gerade kein Monitor mit Tastatur/Maus verfügbar ist:

  → Zugriff über Netzwerk *(SSH muss aktiviert sein!)*

  → Zugriff über serielle Schnittstelle

4. Konfiguration des neuen Systems:

```
sudo raspi-config
```

- Passwort ändern! *(Change User Password)*
- Hostname anpassen *(Network Options → Hostname)*
- Wenn WLAN vorhanden: konfigurieren *(Network Options → Wi-fi)*
- Wohin soll gebootet werden (Desktop/Konsole)? *(Boot Options → Desktop / CLI)*
- Systemsprache, Tastaturlayout, etc. wählen *(Localisation Options → *)*
  - *Locale = “de_DE.UTF-8 UTF-8”; Timezone=Europe→Berlin; Wifi-Country=DE*
- Für Netzwerk-Zugriff “SSH” aktivieren *(Interfacing Options → SSH)*

5.1 Zugriff auf das System mittels Netzwerk:

- Anschließen des RaspberryPi-Boards an einen Router (z.B. Fritzbox)
- Der Router vergibt dem Board eine IP-Adresse mittels DHCP
  (welche Adresse hat das Board bekommen? → im Router nachsehen)

- Falls kein Router zur Hand ist: Root-FS auf der SD-Karte in Linux-Rechner bearbeiten und IP-Adresse fest vergeben (Datei "/etc/network/interfaces")

- Zugriff unter Linux mit "ssh" ("Secure Shell"):

  `ssh -l pi 192.168.0.102` (←IP-Adresse ggfs. Anpassen)

- Zugriff unter Windows z.B. mit "putty" *(https://www.chiark.greenend.org.uk/~sgtatham/putty/)*

  "Session" → IP address: 192.168.0.102 / Port: 22 / Connection type: SSH

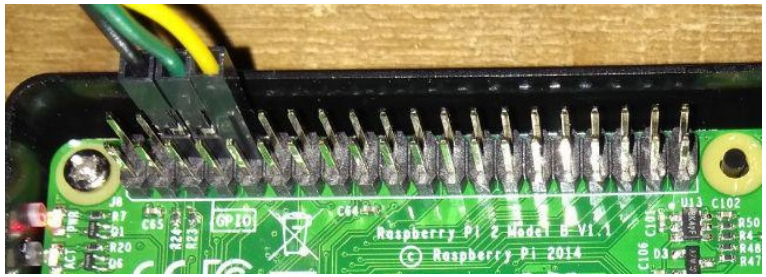
5.2 Zugriff auf das System mittels serieller Schnittstelle (UART) - Teil 1:

- Zugang mit Hilfe eines Terminal-Programms und USB-Seriell-Adapter:
  - Linux: z.B. “picocom”, “minicom”, “cutecom”, …

    ```
    picocom -b115200 /dev/ttyUSB0
    ```

  - Windows: z.B. “putty”

- Bis RaspberryPi 2:
  - TxD und RxD auf den Pins 8 (grün) und 10 (gelb)
  - Anschluss eines geeigneten USB-Seriell-Adapters (auf 3,3V-Pegel achten, **kein RS232**!)



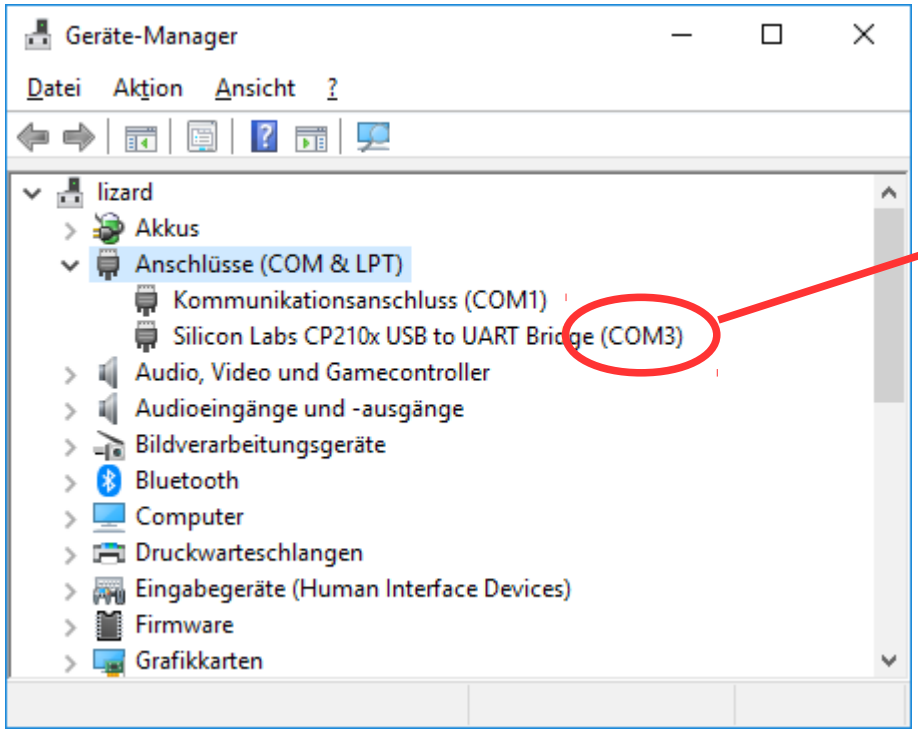
- RaspberryPi 3: UART ist durch Bluetooth-Modul belegt

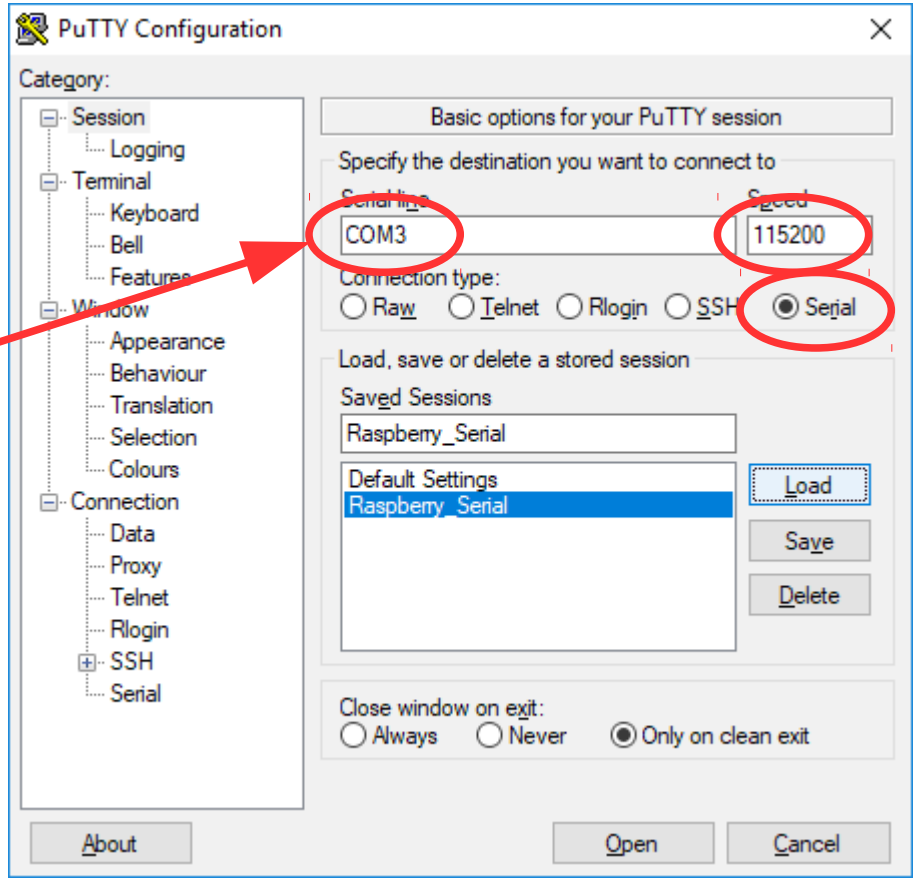
  → Pins 8 und 10 nicht direkt verwendbar, Konfiguration nötig

5.2 Zugriff auf das System mittels serieller Schnittstelle (UART) - Teil 2:

- Beispiel “Putty” (Windows):
  - Welche Schnittstelle (COM1 / COM2 / ...) hat der USB-Seriell-Adapter?

    “Windows” + R → `devmgmt.msc`



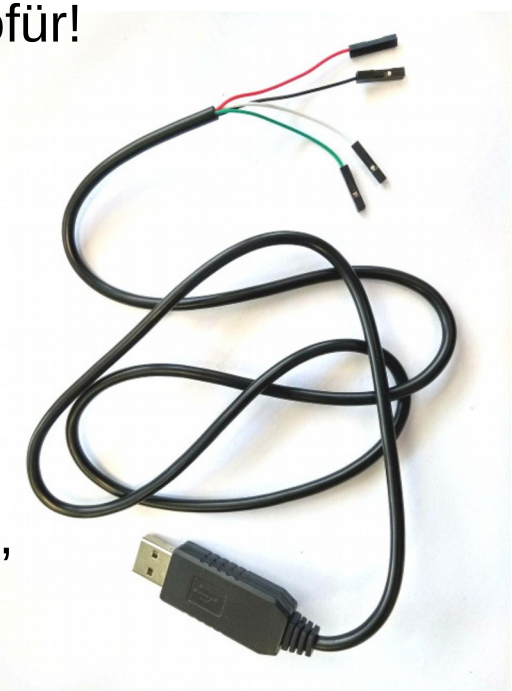


5.2 Zugriff auf das System mittels serieller Schnittstelle - Hinweise

- ***Vorsicht beim Verbinden des Adapters:***

  Für Kommunikation wird nur RX, TX und GND benötigt

  Oft ist auch eine VCC-Leitung verfügbar → Diese führt meist 5V Spannung!
  → Diese Leitung nicht verbinden, ausser man weiß genau wofür!



- ***Vorsicht bei (sehr) günstigen USB-seriell-Adaptern:***

  Hier sind oft **Fälschungen** folgender Chips verbaut:

  - Prolific PL2303
  - FTDI FT232

  → Wird eine Fälschung vom Treiber (unter Windows) erkannt, so wird der Chip unbrauchbar gemacht!

Exkurs: Zeit in Embedded Systemen

- Embedded Systeme (auch RaspberryPi) haben keinen Zeitgeber an Board:

  → Es gibt keine Zeitquelle:

  - beim Herunterfahren wird die aktuelle Zeit in eine Datei gespeichert
  - beim Booten wird diese Zeit wieder gelesen und verwendet

    → die Zeit dazwischen geht “verloren”

- Lösung: Einsatz von Hilfsmitteln
  - Batterie-gepufferte Real-Time-Clock (RTC) als Zusatzbaustein
  - Netzwerk (NTP)
  - DCF-77 (“Funkuhr”)
  - GPS

Übersicht:

Hochschule Augsburg University of Applied Sciences

1. Einleitung: Embedded Systems
2. Das Betriebssystem
3. Inbetriebnahme des Systems
4. **Arbeiten auf der Konsole**
5. Ansprechen externer Komponenten
6. Zusammenfassung

Arbeiten auf der Konsole

Hochschule Augsburg University of Applied Sciences

Arbeiten auf der “Konsole”:

- Paketverwaltung
  - Aktualisieren der installierten Pakete: `sudo apt update` (Paket-Datenbank aktualisieren)
    `sudo apt upgrade` (Pakete aktualisieren)
  - Installieren von neuen Paketen (z.B. “htop”) `sudo apt install htop`
  - Entfernen von Paketen (z.B. “htop”) `sudo apt remove htop`
- Dateisystem-Operationen:
  - z.B. Verzeichnis-Inhalt anzeigen: `ls` ; Verzeichnis wechseln: `cd meine_projekte`
- Editieren von Dateien
  - Nano: `nano <Dateiname>`
  - Vim: `vim <Dateiname>`
- Herunterfahren des Systems
  - `sudo shutdown -h now`
- “Was man sonst so braucht”: *https://files.fosswire.com/2007/08/fwunixref.pdf*

Ausführen von Programmen:

- Programme, die vom System bereit gestellt werden (installiert bzw. im Suchpfad):

  ```
  htop
  ```

- Programme, die im aktuellen Verzeichnis liegen (nicht “installiert”):

  ```
  ./mein_programm
  ```

- Beispiel: Ausführen von Python-Skripten
  - Mittels Interpreter (als Argument): `python mein_programm.py`
  - Als “eigenständiges Programm”: `./mein_programm.py`

    Wichtig: Das Skript muss

    - die “*Magic-Line*” enthalten (1. Zeile): `#!/usr/bin/python`
    - ausführbar sein: `chmod +x mein_programm.py`

Übersicht:

Hochschule Augsburg University of Applied Sciences

Interaktion mit der Umwelt – Auswahl der Komponenten

- Welche Aufgabe soll erfüllt werden, welche Hardware wird angesprochen?
  - Nur einen Taster auswerten bzw. eine LED ansteuern?
  - Soll ein Display angebunden werden?
  - Sollen analoge Messwerte eingelesen werden?
  - Soll Entfernung gemessen werden?
  - Soll ein Motor / Schrittmotor betrieben werden?
  - …

→ Wird ein externer Baustein benötigt?

  - Welche Schnittstelle zur Kommunikation wird angeboten?
  - Wie ist das Protokoll beschaffen?
  - Welche elektrischen Eigenschaften besitzt die Schnittstelle?

Ansprechen externer Komponenten

Hochschule Augsburg University of Applied Sciences

Interaktion mit der Umwelt – Bitte das System nicht “grillen” (1)

- Die IO-Pins sind nur 3,3Volt-tolerant!

  → Höhere Spannungspegel am IO-Pin zerstören diesen, evtl. sogar das komplette SoC des RaspberryPi!

- Die IO-Pins sind nicht für hohe Ströme ausgelegt!

  → 10mA je IO-Pin und 50mA gesamt sind in Ordnung, alles darüber zerstört den IO-Pin bzw. das komplette SoC des RaspberryPi!

- **VOR INBETRIEBNAHME: DIE SCHALTUNG NOCH (MINDESTENS) EINMAL KONTROLLIEREN!**

Interaktion mit der Umwelt – Bitte das System nicht “grillen” (2)

- Die IO-Pins sind nur 3,3Volt-tolerant!

  → Dafür sorgen, dass von Außen nur 3,3Volt-Pegel an den Pins anliegen, oft genügt ein kleiner Spannungsteiler um höhere Pegel anzupassen.

- Die IO-Pins sind nicht für hohe Ströme ausgelegt!

  → Den Strom begrenzen (z.B. mittels Widerstand), falls für das externe Bauteil (z.B. Motor) hohe Leistung benötigt wird: z.B. Transistor-Schaltung oder speziellen Treiber-Baustein verwenden

  (verfügbare Leistung pro Pin: 3.3V * 0.01A = 0,033W = 33mW)

Ansprechen externer Komponenten

Interaktion mit der Umwelt – Einfache Beispiele (1)

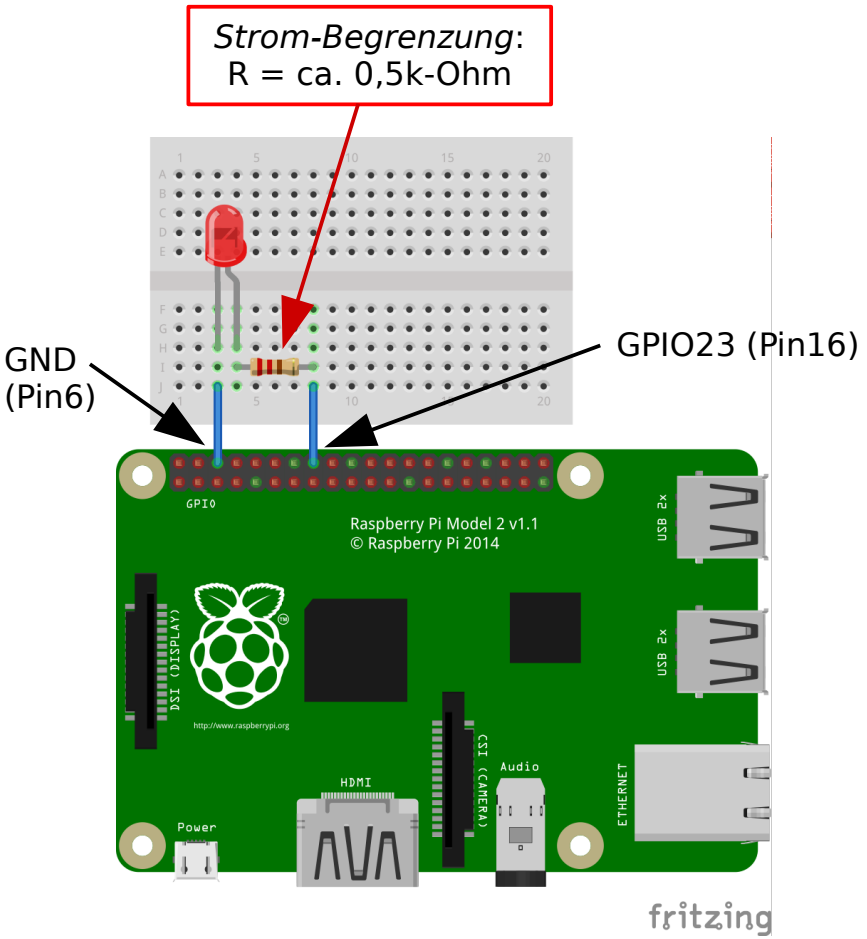
- LED ansprechen *(hier: "active-high")*
  - Spannung: vom RaspberryPi-GPIO-Pin
  - Strom: muss begrenzt werden!

    → bei einer LED: Vorwiderstand

    → bei mehreren LEDs: zus. Transistoren

```
import RPi.GPIO as GPIO
import time

LED_PIN=23
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(LED_PIN,GPIO.OUT)
while True:
  print("LED an")
  GPIO.output(LED_PIN,GPIO.HIGH)
  time.sleep(1)
  print("LED aus")
  GPIO.output(LED_PIN,GPIO.LOW)
  time.sleep(1)
```



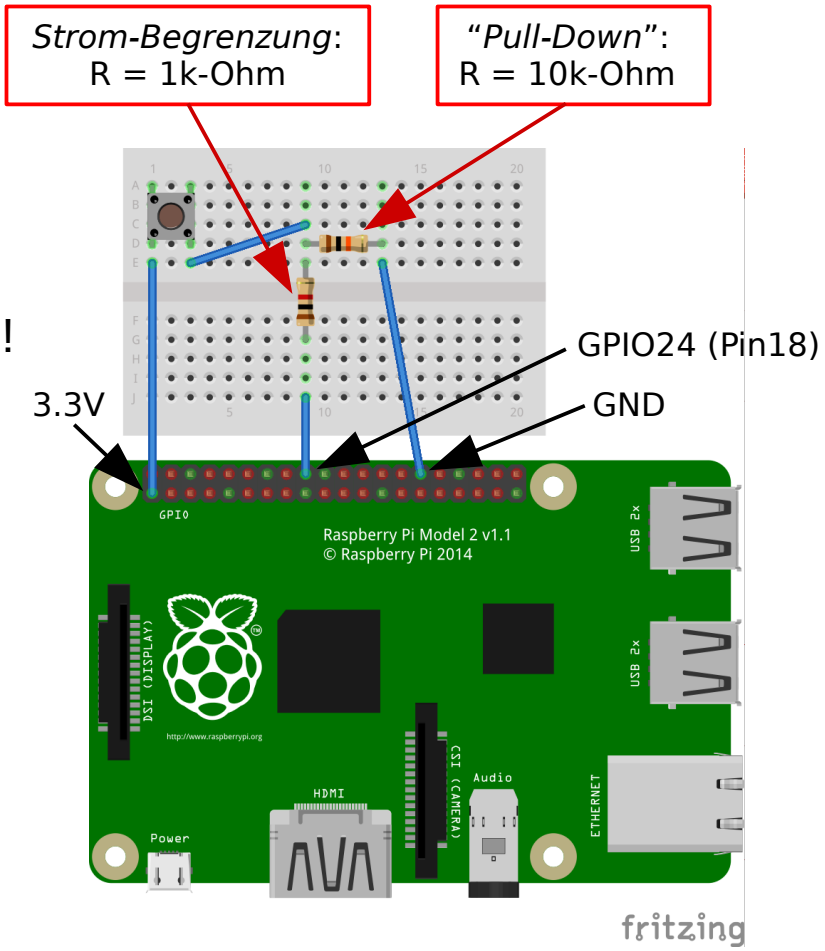
Interaktion mit der Umwelt – Einfache Beispiele (2)

- Taster einlesen *(hier: "active-high")*
  - Spannung: "von extern", muss für ca. 3.3V sein!
  - Auch der Strom muss begrenzt werden!
  - Was passiert, wenn Schalter nicht gedrückt ist?

    → Auch dann muss ein definierter Pegel anliegen!

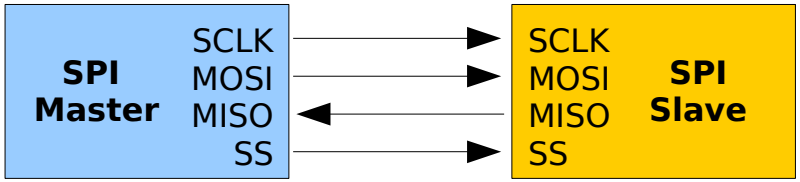
```
import RPi.GPIO as GPIO
import time

TASTER_PIN=24
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(TASTER_PIN,GPIO.IN)
while True:
  pressed = GPIO.input(TASTER_PIN)
  if pressed == 1:
    print("Taster gedrueckt")
  else:
    print("Taster offen")
  time.sleep(0.1)
```

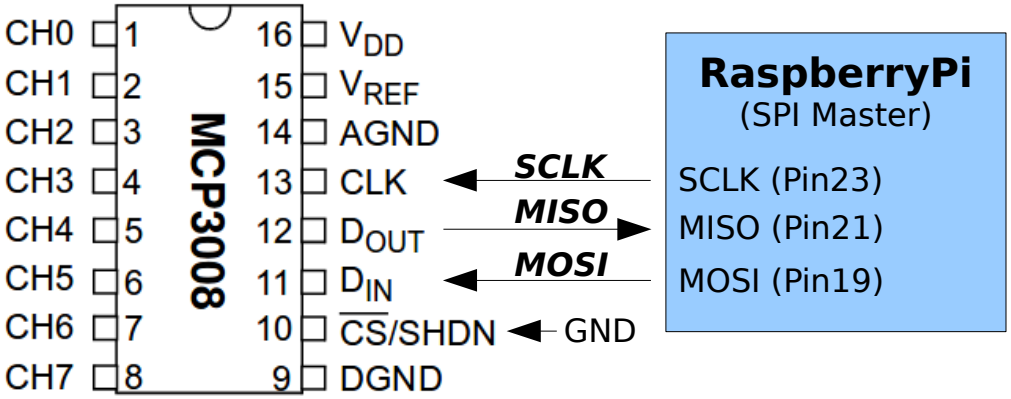


Interaktion mit der Umwelt – Protokolle

- **SPI** (“Serial Peripheral Interface”)
  - MOSI (Daten Master→Slave),
    MISO (Daten Slave→Master),
    SCLK (Takt)
  - über SS (SlaveSelect) auch 1:n-Verbindungen möglich (1 Master, mehrere Slaves)



→ z.B. Anbindung Baustein “MCP3008” zur Messung von Analogwerten

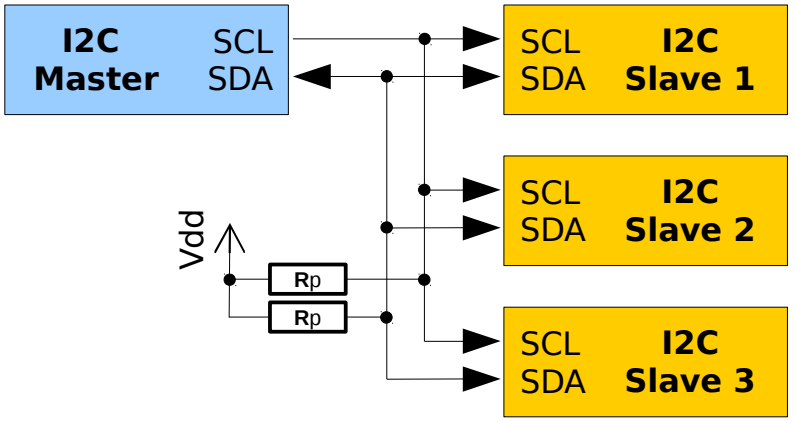


- ***Generell eine gute Idee:*** ***Datenblätter lesen!***

Ansprechen externer Komponenten

Hochschule Augsburg University of Applied Sciences

Interaktion mit der Umwelt – Protokolle



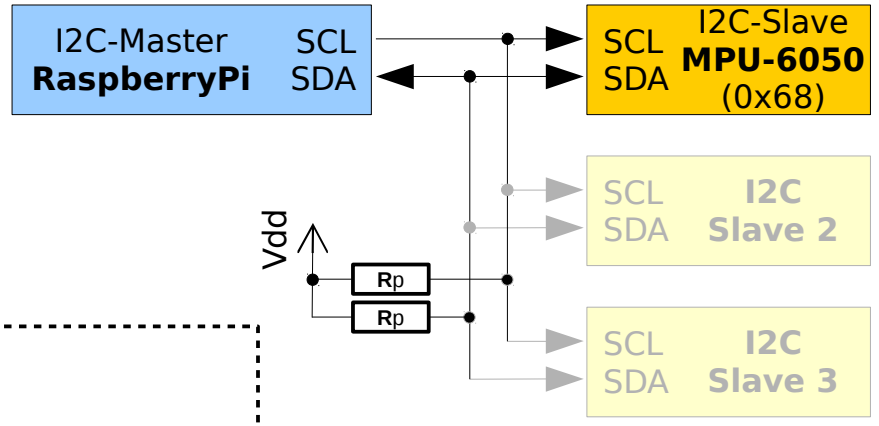
- **I²C** (auch I2C, IIC: “Inter-Integrated Circuit”):
  - SCL (Takt) und SDA (Daten)
  - Pullup-Widerstände nötig (i.d.R.: Rp=4,7kOhm), oft sind diese schon in Modulen integriert

→ z.B. Anbindung Modul zur

- Strom-Messung
- OLED-Display
- Gyroskop / Beschleunigung: z.B. “ST L3GD20H” (Gyro), “MPU-6050” (Gyro+Beschl.)
- RealTime-Clock: z.B. “DS1307”
- …

Ansprechen externer Komponenten

Interaktion mit der Umwelt – Protokolle



- **I²C** – Code-Beispiel
  für Gyroskop/Beschl.-Sensor “MPU6050”:

```
import smbus
import time

# I2C-Adresse des MPU6050 (siehe 'i2cdetect -y 1')
address = 0x68

# Interne Register des MPU6050 (siehe Datenblatt...)
accel_x_reg = 0x3b

# Device erstellen und aktivieren (Power-Mgmt.-Reg.)
MPU6050 = smbus.SMBus(1)
MPU6050.write_byte_data(address, 0x6b, 0)

while True:
  h = MPU6050.read_byte_data(address, accel_x_reg)
  l = MPU6050.read_byte_data(address, accel_x_reg+1)
  value = (h << 8) + l
  print("Accel_x = {}". format(value))
  time.sleep(0.1)
```

Interaktion mit der Umwelt – Sonstiges

- Entfernung messen:
  - Ultraschall:

    z.B. HC-SR04: Abstand wird über Impulslänge an Data-Pin bestimmt

    → Achtung! Modul benötigt 5V Versogungsspannung,

    an Data-Pin wird Spannungsteiler benötigt: 5V → 3.3V

  - Infrarot: z.B. “Sharp GP2Y0A21” → Abstand über Analog-Wert: 3V (nah) – 0.3V (fern)

- Motoren ansteuern über externe Treiber (meist mit MOSFET o.Ä. ausgestattet):
  - Linear-Motor: z.B. L298
  - Schritt-Motor-Steuerung

Übersicht:

Hochschule Augsburg University of Applied Sciences

1. Einleitung: Embedded Systems
2. Das Betriebssystem
3. Inbetriebnahme des Systems
4. Arbeiten auf der Konsole
5. Ansprechen externer Komponenten
6. **Zusammenfassung**

Zusammenfassung

Hochschule Augsburg University of Applied Sciences

Was ist ein “Embedded System”?

- Gewöhnlich klein u. energiesparend, aber leistungsschwächer als “Standard-PC”
- Bietet viele Extras (GPIO, ADC, …), die ein “Standard-PC” nicht hat

Wie nehme ich den RaspberryPi in Betrieb?

- Image auf SD-Karte kopieren
- System passend konfigurieren → auch auf Sicherheitsaspekte achten!

Wie kann mit dem RaspberryPi mit Python programmiert werden?

- Editoren (auf der Konsole oder auch graphisch) zum Programmieren
- Auf der Konsole (Shell) sind Skripte ausführbar

Wie können externe Komponenten angebunden werden?

- Über verschiedene Schnittstellen können vielfältige Module angebunden werden
- Darauf achten, dass nichts zerstört wird (→ Spannungspegel und Stromfluss!!!)