

# **RASPBERRY PI**

## **LERNE DIE PROGRAMMIERUNG EINES MIKROCOMPUTERS KENNEN**

Sven Krauß

# TERMINE

- KW04 > 26.01.2022
- KW05 > 02.02.2022
- KW06 > 09.02.2022
- KW07 > 16.02.2022
- KW08 > 23.02.2022
- KW09 > Entfällt wg Ferien
- KW10 > 09.03.2022

# THEMENÜBERSICHT

## 1. Was ist ein Raspberry Pi?

Einführung und den Raspberry Einrichten

## 2. Einführung in Python

`print("Hallo Welt")`

## 3. Arbeiten mit den GPIOs

LEDs und Taster

## 4. Sensoren einlesen

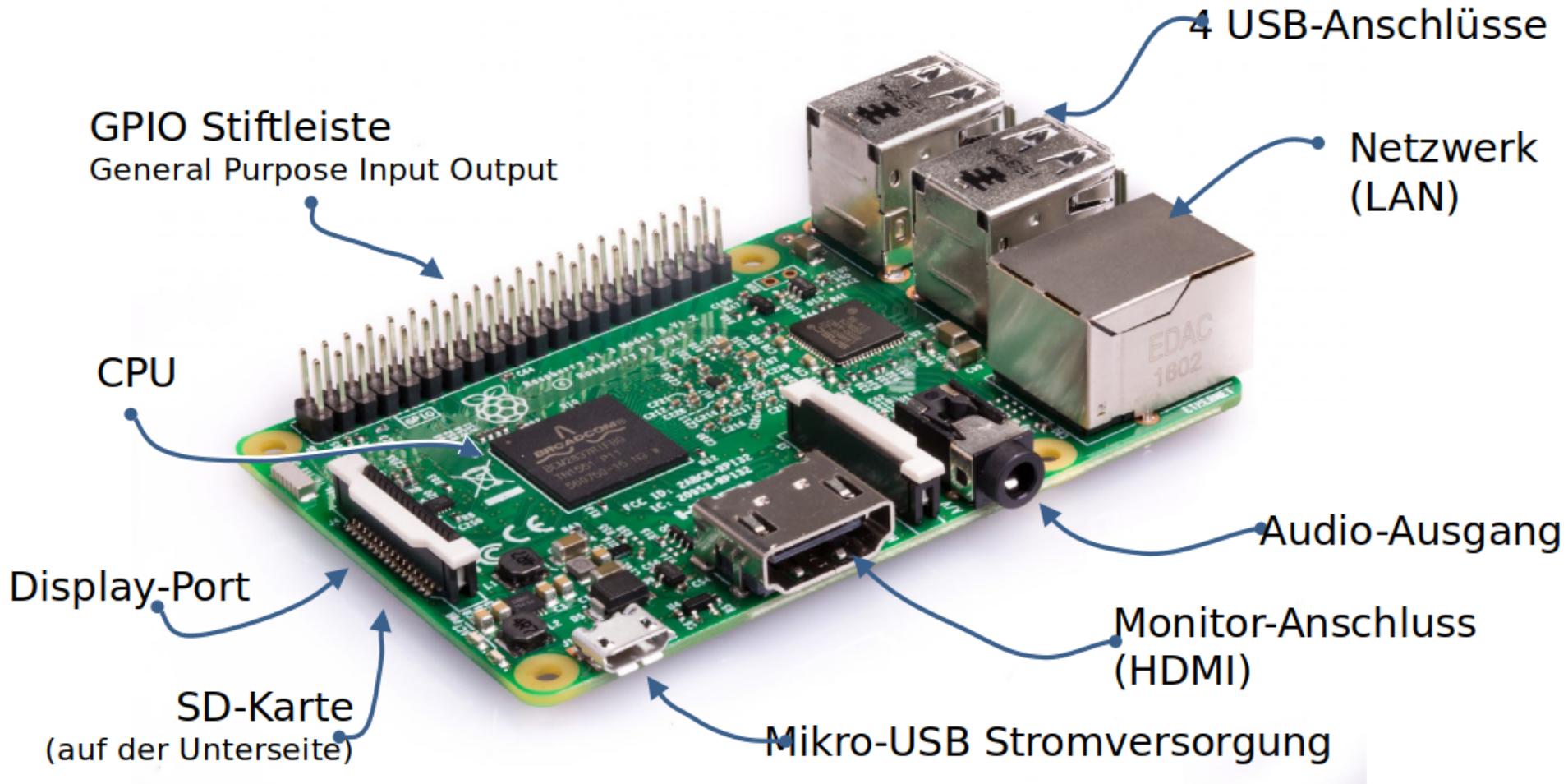
## 5. Weitere Themen

- Raspberry Pi im Netzwerk
- Sonic Pi
- Türklingel
- Eure Vorschläge

# WAS IST DER RASPBERRY PI

- Komponenten des Pi
- Anschlüsse
- Umgang mit dem Pi - Was ist ESD?
- Betriebssystem

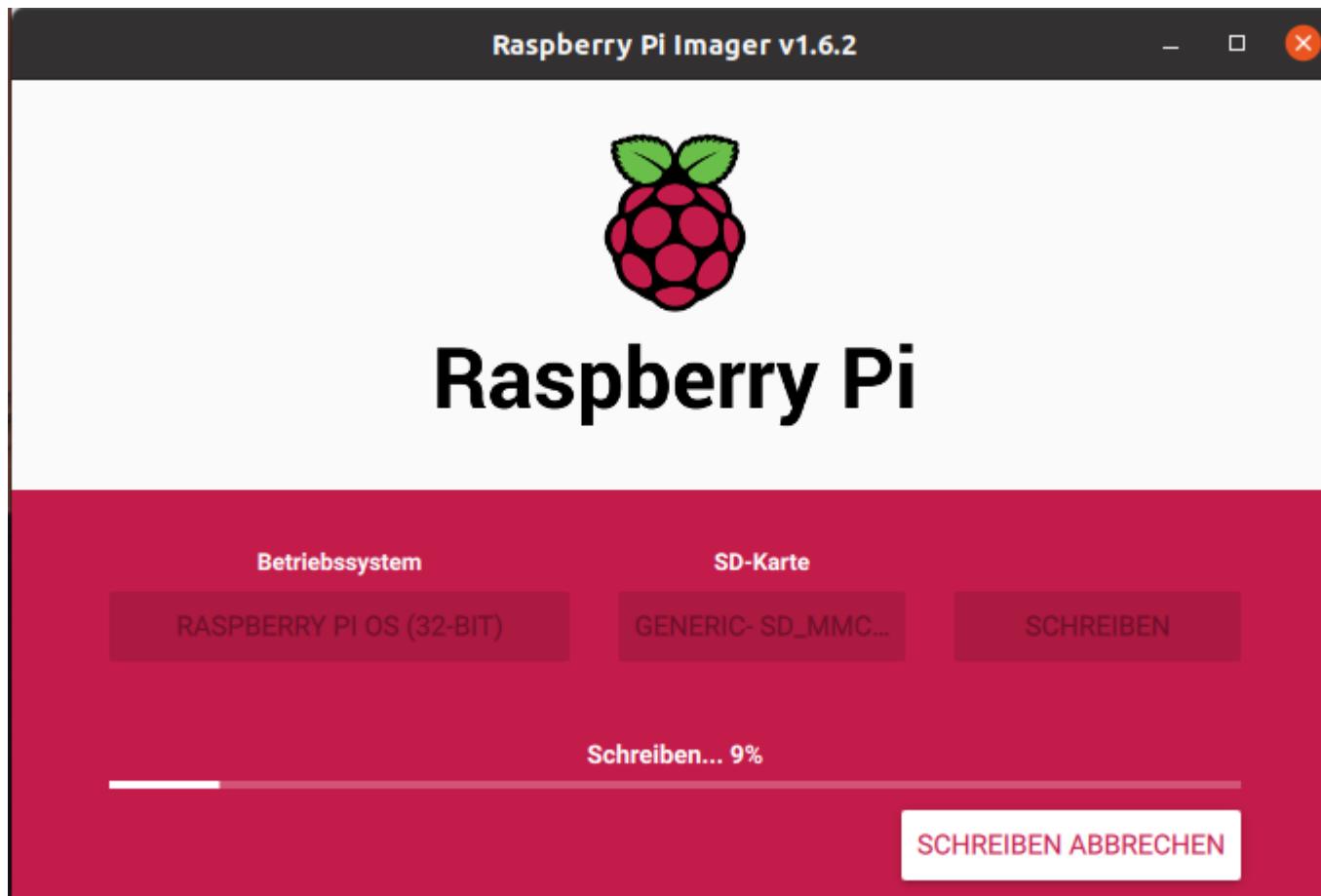
# HARDWARE



# EINIGE ANWENDUNGEN

- Desktop-Computer
- Internetradio
- Überwachungskamera / Webcam
- Robotersteuerung
- IoT / Hausautomation
- Web-Server
- Datenlogger
- Kiosk-Modus
- Retro-Spiele
- ...

# INSTALLATION



# AUFBAU UND START

*Aufgabe: Systemupdate*

# DIE SPRACHE DER COMPUTER

```
1 0000000 49a1 75c3 7ff0 0ebe 798c e71a d095 a2f3
2 0000010 65bb e654 9fbe 3b33 a1a2 8fe6 91a5 1c61
3 0000020 4487 b7f2 9ba1 9858 2960 d9b6 6a88 af86
4 0000030 85cf 725c b7cb a4ca ddb7 cee7 cbe7 fd69
5 0000040 6988 0dac e990 ecd5 1346 1fe0 ab97 13dd
6 0000050 a0f4 0313 c5da 7fae 3f9b ca04 68f5 71b0
7 0000060 a9eb 3d14 8258 64ea 8836 7e7f d051 fe64
8 0000070 60d7 9712 cbaf ccf7 35ae 072c 084a fffd
9 0000080 5b3e 9d93 cdcc 7365 aeb9 3ead f26c 7e23
10 0000090 38c6 d64d 7625 1daa 5070 6297 9de2 9ea5
```

# DIE SPRACHE DER COMPUTER

```
1 801059c <__adddf3>:  
2 801059c: b530          push {r4, r5, lr}  
3 801059e: ea4f 0441    mov.w r4, r1, lsl #1  
4 80105a2: ea4f 0543    mov.w r5, r3, lsl #1  
5 80105a6: ea94 0f05    teq r4, r5  
6 80105aa: bf08          it eq  
7 80105ac: ea90 0f02    teqeq r0, r2  
8 80105b0: bf1f          itttt ne  
9 80105b2: ea54 0c00    orrsne.w ip, r4, r0
```

# EINFÜHRUNG IN PYTHON

```
1 print("Hallo Welt")
2 ida = 5
3 ida
4 paul = 42
5 paul
6 print("Ida ist", ida)
7 print("Paul ist", paul)
8 print("Zusammen sind sie", (paul + ida))
9 ida=paul
10 print("Zusammen sind sie", (paul + ida))
```

# **ENTWICKLUNGSUMGEBUNG**

Demo

# QUADRIERER

```
1 zahl = input("Gib eine Zahl ein: ")  
2 quadrat = zahl*zahl  
3 print("Das Quadrat ist:", quadrat)
```

# DATENTYPEN

(Im Interpreter)

```
1 var = -5
2 type(var)
3
4 var = 5.0
5 type(var)
6
7 var = "Hallo Welt"
8 type(var)
9
10 var = "55"
11 type(var)
12
13 var = int(var)
14 type(var)
```

# VERZWEIGUNGEN

```
1 if [Bedingung]:  
2     [Anweisung]  
3     ...  
4 elif:  
5     [Anweisung]  
6     ...  
7 else:  
8     [Anweisung]  
9     ...
```

Bedingungen können sein:

- Kleiner:  $a < b$
- Größer:  $a > b$
- Kleiner gleich:  $a \leq b$
- Größer gleich:  $a \geq b$
- Gleich  $a == b$

# VERZWEIGUNGEN - BEISPIEL

```
1 alter = int(input("Wie alt bist du? "))
2 if alter < 10:
3     print("Kind")
4 elif alter < 18:
5     print("Jugendlich")
6 else:
7     print("Erwachsen")
```

# WHILE-SCHLEIFEN

```
1 while Bedingung:  
2     Anweisung  
3     ...  
4     break
```

## Zahlenraten

```
1 from random import randint  
2  
3 print("Rate eine Zahl zwischen 1 und 100")  
4 zufallszahl = randint(1,100)  
5 zahl = int(input("Dein Vorschlag: "))  
6  
7 while zahl != zufallszahl:  
8     if zahl < zufallszahl:  
9         print("Zu klein")  
10    else:  
11        print("Zu groß")  
12    zahl = int(input("Dein Vorschlag: "))  
13  
14 print("Richtig!")
```

# FOR-SCHLEIFEN

```
1 for x in I:  
2     Anweisung  
3     ...  
4     break
```

## Beispiel Fakultät

```
1 fak = 1  
2 for i in range(1,10):  
3     fak = fak * i  
4 print("Fakultät von:", i, "ist", fak)
```

# BIBLIOTHEKEN EINBINDEN

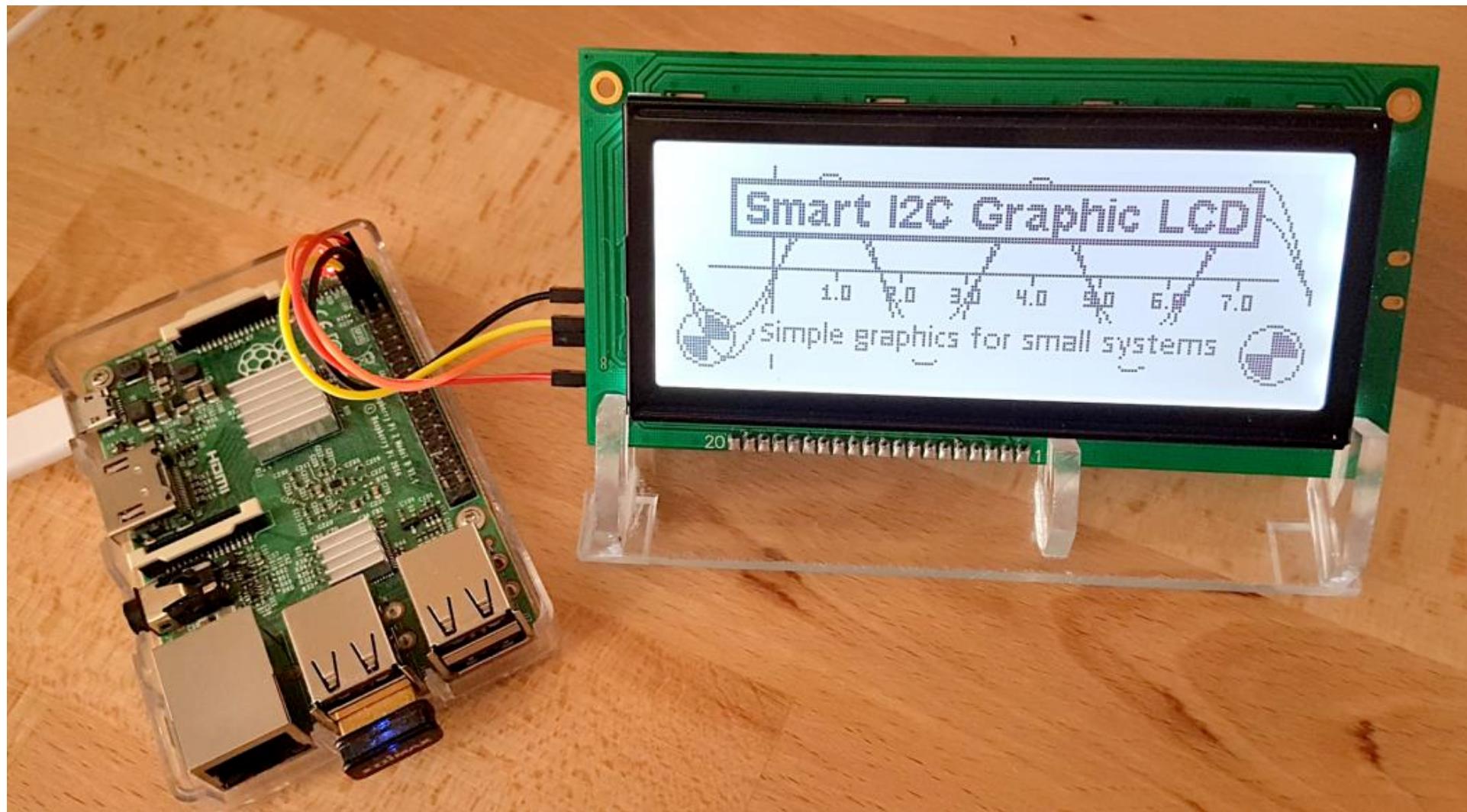
- Die meisten Probleme in der Softwareentwicklung wurden schon von anderen bearbeitet.
- Teile der Problemlösung können in Bibliotheken bereitgestellt werden.
- Das Rad muss nicht neu erfunden werden.

```
1 import random
2 import math
3
4 zufallszahl = random.randint(1,100)
5 s = math.sin(0.5)
```

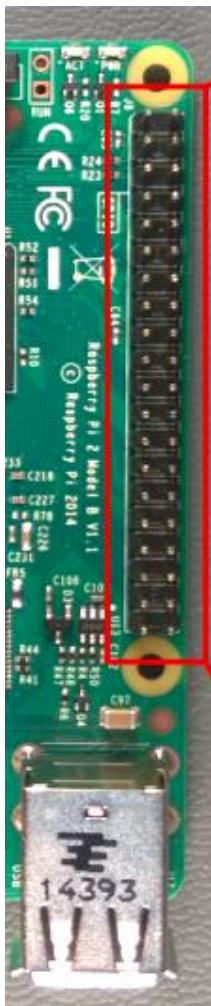
Python Standard Bibliothek

Python Package Index

# **GENERAL PURPOSE INPUT/OUTPUT**



# GPIO PINS (LAYOUT)



Alternate Function	
3.3V PWR	1
I2C1 SDA	GPIO 2
I2C1 SCL	GPIO 3
GPIO 4	5
GND	9
GPIO 17	11
GPIO 27	13
GPIO 22	15
3.3V PWR	17
SPI0 MOSI	GPIO 10
SPI0 MISO	GPIO 9
SPI0 SCLK	GPIO 11
GND	25
Reserved	27
GPIO 5	29
GPIO 6	31
GPIO 13	33
SPI1 MISO	GPIO 19
GPIO 26	35
GND	37
Alternate Function	
2	5V PWR
4	5V PWR
6	GND
8	UART0 TX
10	UART0 RX
12	GPIO 18
14	GND
16	GPIO 23
18	GPIO 24
20	GND
22	GPIO 25
24	GPIO 8
26	GPIO 7
28	Reserved
30	GND
32	GPIO 12
34	GND
36	GPIO 16
38	GPIO 20
40	GPIO 21
SPI0 CS0	
SPI0 CS1	
SPI1 CS0	
SPI1 MOSI	
SPI1 SCLK	

# GPIO MIT PYTHON ANSTEUERN

- Es gibt zwei verschiedene Modi für die Identifizierung
  1. Die BOARD-Methode verwendet die Nummern 1 bis 40 entsprechend dem Layout.
  2. Die BCM-Methode bezeichnet die Pins entsprechend ihres Interface-Kanals (channel).
- Einige GPIO-Anschlüsse können mehrere Funktionen übernehmen (SPI, I2C, USB...)
- Achtung: Ein GPIO-Ausgang darf mit maxima  
le 16mA belastet werden.
- Alles GPIOs funktionieren mit 3.3V

# GPIO MIT PYTHON ANSTEUERN

- Es gibt zwei verschiedene Modi für die Identifizierung
  1. Die BOARD-Methode verwendet die Nummern 1 bis 40 entsprechend dem Layout.
  2. Die BCM-Methode bezeichnet die Pins entsprechend ihres Interface-Kanals (channel).
- Einige GPIO-Anschlüsse können mehrere Funktionen übernehmen (SPI, I2C, USB...)
- Achtung: Ein GPIO-Ausgang darf mit maxima  
le 16mA belastet werden.
- Alles GPIOs funktionieren mit 3.3V

# GPIO MIT PYTHON ANSTEUERN

- Es gibt zwei verschiedene Modi für die Identifizierung
  1. Die BOARD-Methode verwendet die Nummern 1 bis 40 entsprechend dem Layout.
  2. Die BCM-Methode bezeichnet die Pins entsprechend ihres Interface-Kanals (channel).
- Einige GPIO-Anschlüsse können mehrere Funktionen übernehmen (SPI, I2C, USB...)
- Achtung: Ein GPIO-Ausgang darf mit maxima  
le 16mA belastet werden.
- Alles GPIOs funktionieren mit 3.3V

# GPIO MIT PYTHON ANSTEUERN

- Es gibt zwei verschiedene Modi für die Identifizierung
  1. Die BOARD-Methode verwendet die Nummern 1 bis 40 entsprechend dem Layout.
  2. Die BCM-Methode bezeichnet die Pins entsprechend ihres Interface-Kanals (channel).
- Einige GPIO-Anschlüsse können mehrere Funktionen übernehmen (SPI, I2C, USB...)
- Achtung: Ein GPIO-Ausgang darf mit maxima 16mA belastet werden.
- Alles GPIOs funktionieren mit 3.3V

# GPIO MIT PYTHON ANSTEUERN

- Es gibt zwei verschiedene Modi für die Identifizierung
  1. Die BOARD-Methode verwendet die Nummern 1 bis 40 entsprechend dem Layout.
  2. Die BCM-Methode bezeichnet die Pins entsprechend ihres Interface-Kanals (channel).
- Einige GPIO-Anschlüsse können mehrere Funktionen übernehmen (SPI, I2C, USB...)
- Achtung: Ein GPIO-Ausgang darf mit maximale 16mA belastet werden.
- Alles GPIOs funktionieren mit 3.3V

# GPIO MIT PYTHON ANSTEUERN

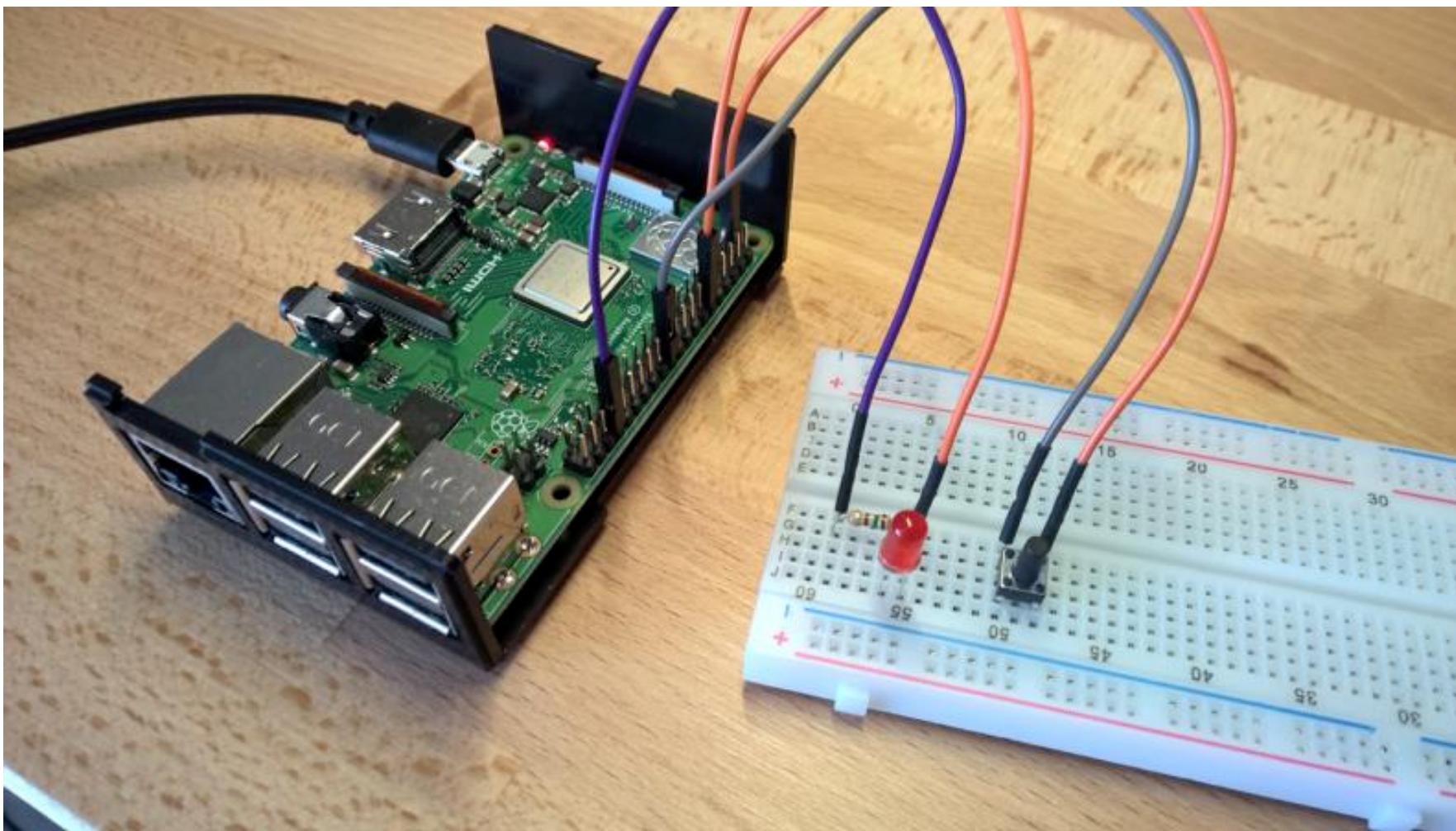
- Es gibt zwei verschiedene Modi für die Identifizierung
  1. Die BOARD-Methode verwendet die Nummern 1 bis 40 entsprechend dem Layout.
  2. Die BCM-Methode bezeichnet die Pins entsprechend ihres Interface-Kanals (channel).
- Einige GPIO-Anschlüsse können mehrere Funktionen übernehmen (SPI, I2C, USB...)
- Achtung: Ein GPIO-Ausgang darf mit maxima  
le 16mA belastet werden.
- Alles GPIOs funktionieren mit 3.3V

# GPIO MIT PYTHON ANSTEUERN 2

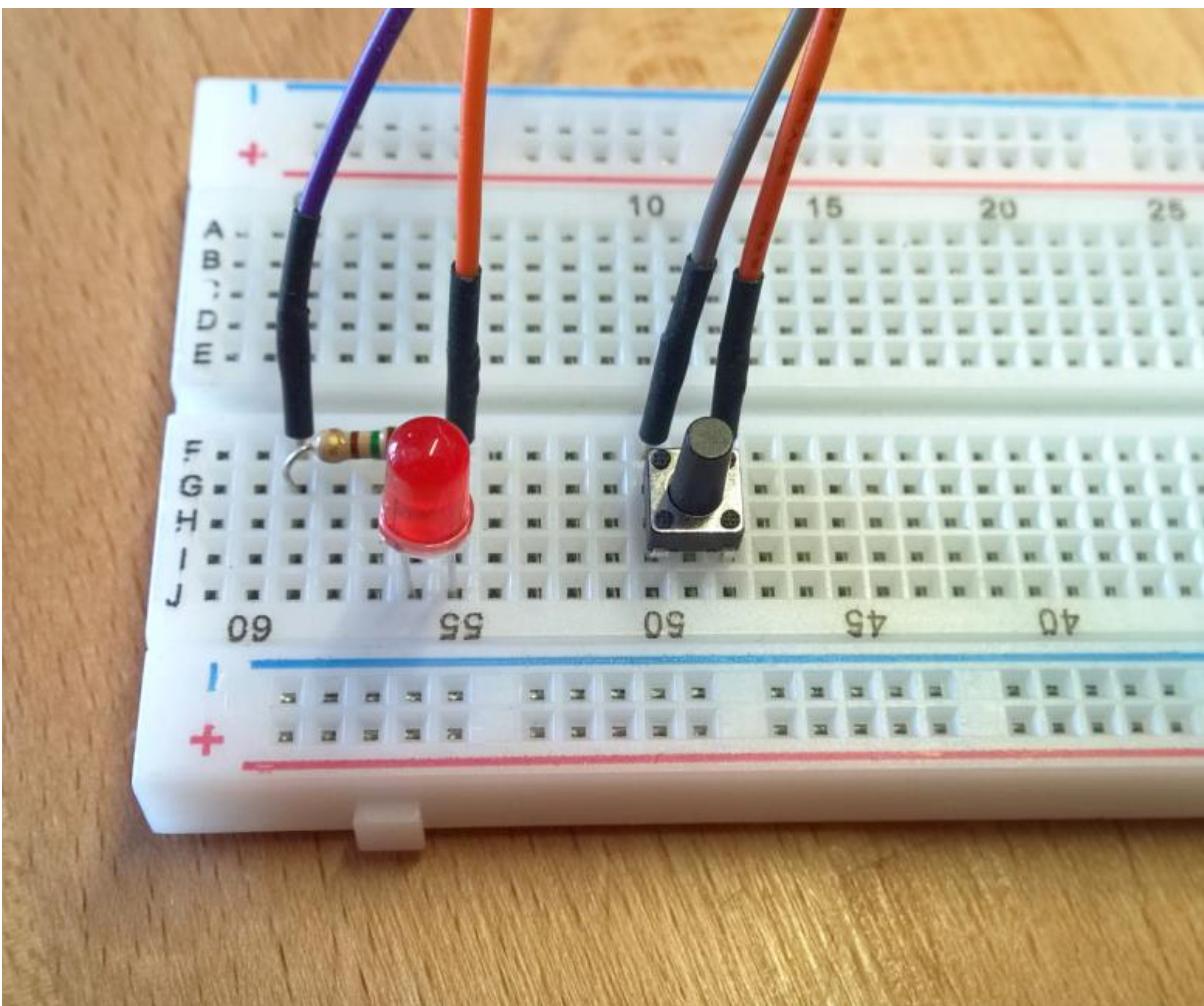
- Die 'RPi'-Bibliothek erlaubt den Ansteuerung der GPIO-Anschlüsse

```
1 import RPi.GPIO as GPIO  
2  
3 GPIO.setmode(GPIO.BCM)
```

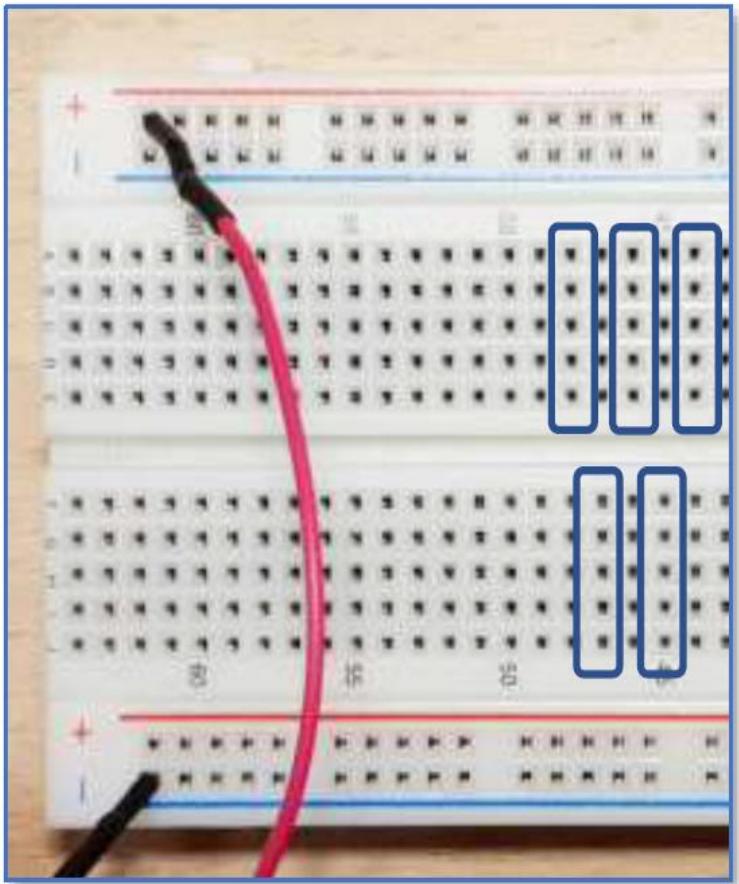
# AUFBAU MIT BREADBOARD



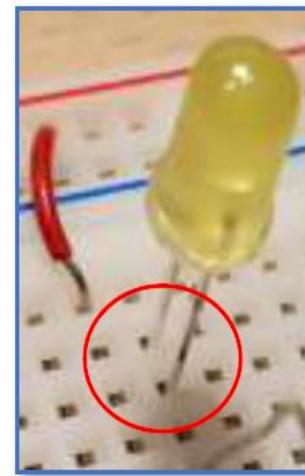
# AUFBAU MIT BREADBOARD



# AUFBAU MIT BREADBOARD



Die 5 Löcher der senkrechten Reihen sind miteinander elektrisch verbunden. Wir nennen die senkrechte Reihe ein „Feld“. **Wichtig:** Die Bauteile müssen so eingesteckt werden, dass ihre Beine nicht in demselben Feld stecken, sondern in benachbarten Feldern.



falsch

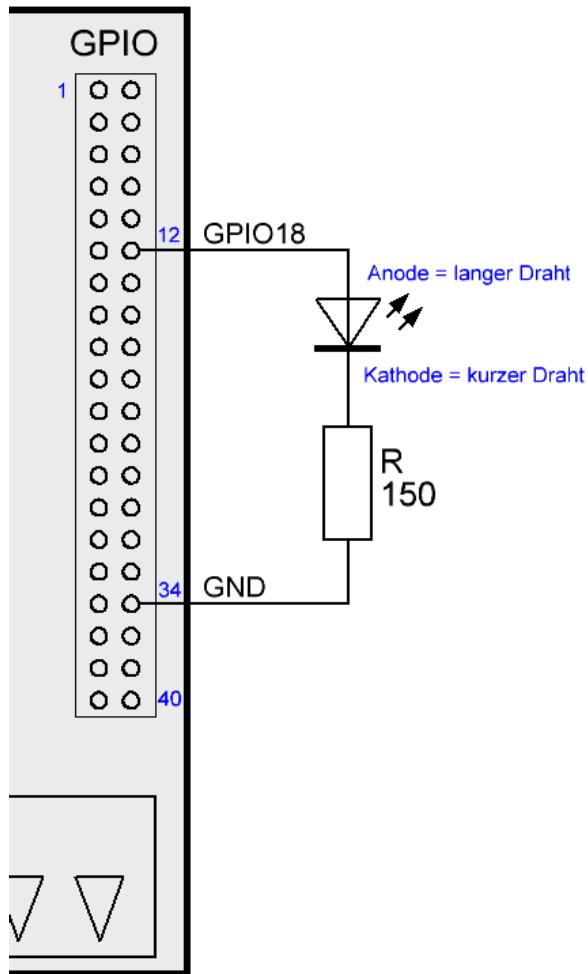


richtig

# BLINKENDE LEUCHTDIODE

- LEDs können nur mit einem Vorwiderstand an den GPIOs angeschlossen werden. In dem Fall 150 Ohm.
- Die Polarität muss beachtet werden. Der kurze Anschluss kennzeichnet die Anode und muss mit der positiven Seite der Spannungsversorgung verbunden werden. Der lange Anschluss ist die Kathode und gehört an die negative Seite (hier Ground GND).

# BLINKENDE LEUCHTDIODE



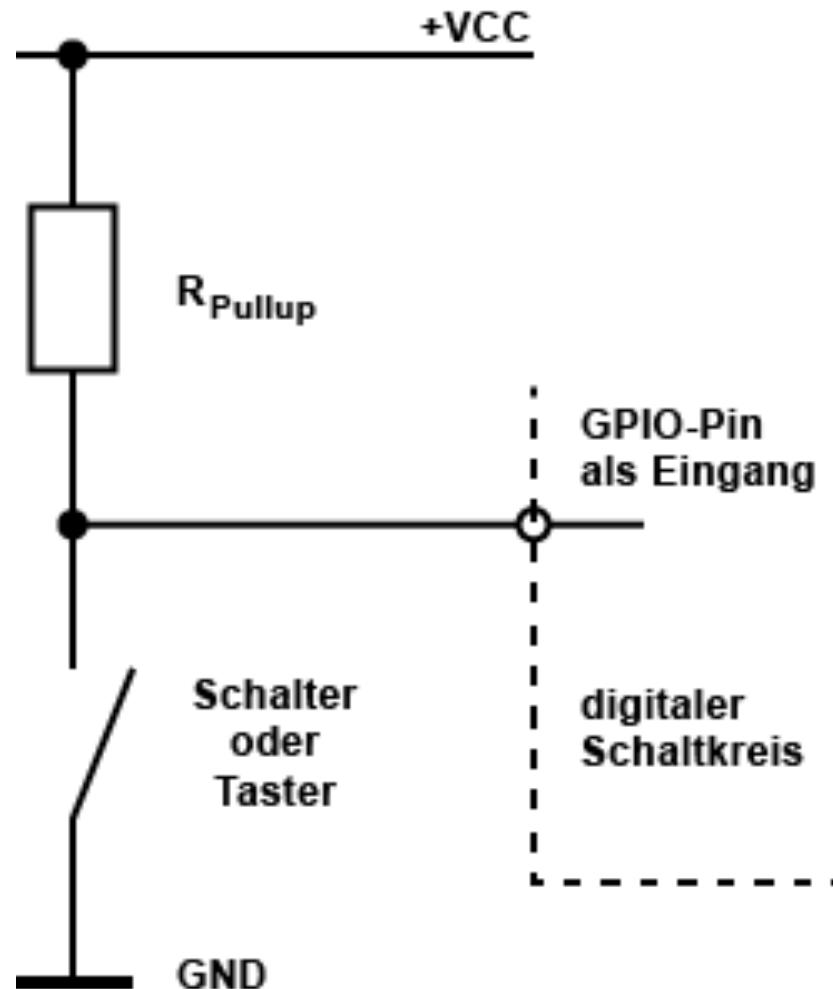
# BLINKENDE LEUCHTDIODE

```
1 import time
2 import RPi.GPIO as GPIO
3
4 led_pin = 18
5
6 GPIO.setmode(GPIO.BCM)
7 GPIO.setup(led_pin, GPIO.OUT)
8
9 try:
10     while True:
11         GPIO.output(led_pin, GPIO.HIGH)
12         time.sleep(0.5)
13         GPIO.output(led_pin, GPIO.LOW)
14         time.sleep(0.5)
15 except KeyboardInterrupt:
```

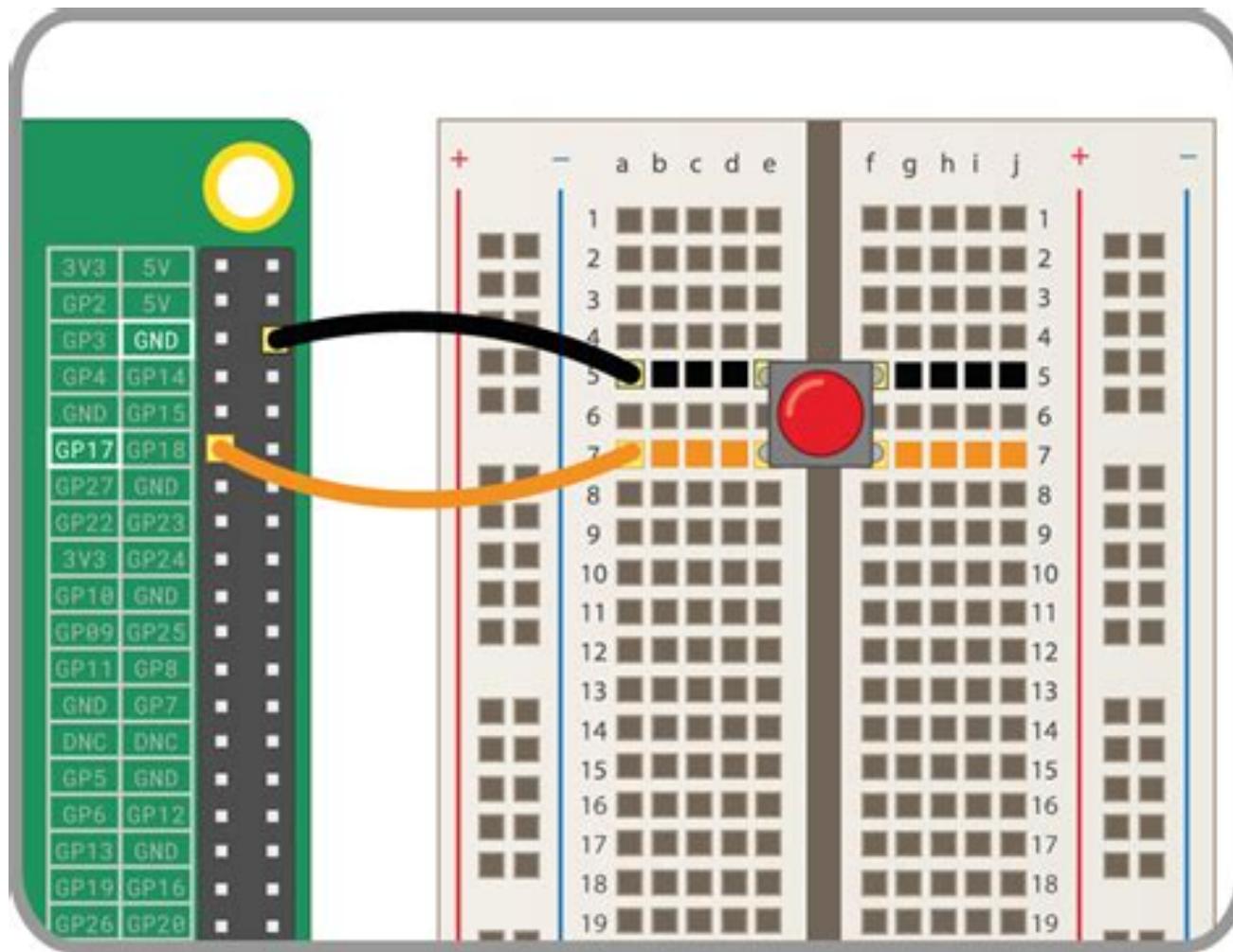
# TASTER ABFRAGEN

- Kann verwendet werden um den Zustand eines elektrischen Kontakts zu erfassen
- z.B. Lichtschranken, Türkontakte, Taster, etc...
- Pull-Up oder Pull-Down beachten ([Klick](#))
- Die Pull-Up / Pull-Down Widerstände sind in der CPU bereits eingebaut und können über die Bibliothek aktiviert werden.

# TASTER ABFRAGEN



# TASTER ABFRAGEN



# TASTER ABFRAGEN

```
1 import time
2 import RPi.GPIO as GPIO
3
4 tast_pin = 17
5
6 GPIO.setmode(GPIO.BCM)
7 GPIO.setup(tast_pin, GPIO.IN, pull_up_down = GPIO.PUD_UP)
8 try:
9     while True:
10         time.sleep(0.5)
11         if GPIO.input(tast_pin) == GPIO.HIGH:
12             print("Taster (nicht) gedrückt")
13         else:
14             print("Taster (nicht) gedrückt")
15
```

# TASTER ABFRAGEN

Aufgabe: Ändere das Programm so ab, dass nur die Ereignisse "Taste gedrückt" und "Taste losgelassen" ausgegeben werden.

# FUNKTIONEN

```
1 # Funktionsdefinition
2 def sekunden_alter(name, alter):
3     print(name, "ist", alter, "Jahre alt!")
4     sekunden = alter * 365 * 24 * 60 * 60
5     print(name, "ist an seinem Geburtstag", sekunden, "Sekunden")
6     print()
7
8 # Funktionsaufrufe
9 sekunden_alter("Paul", 12)
10 sekunden_alter("Jonas", 15)
11 sekunden_alter("Herbert", 42)
12 sekunden_alter("Luise", 66)
```

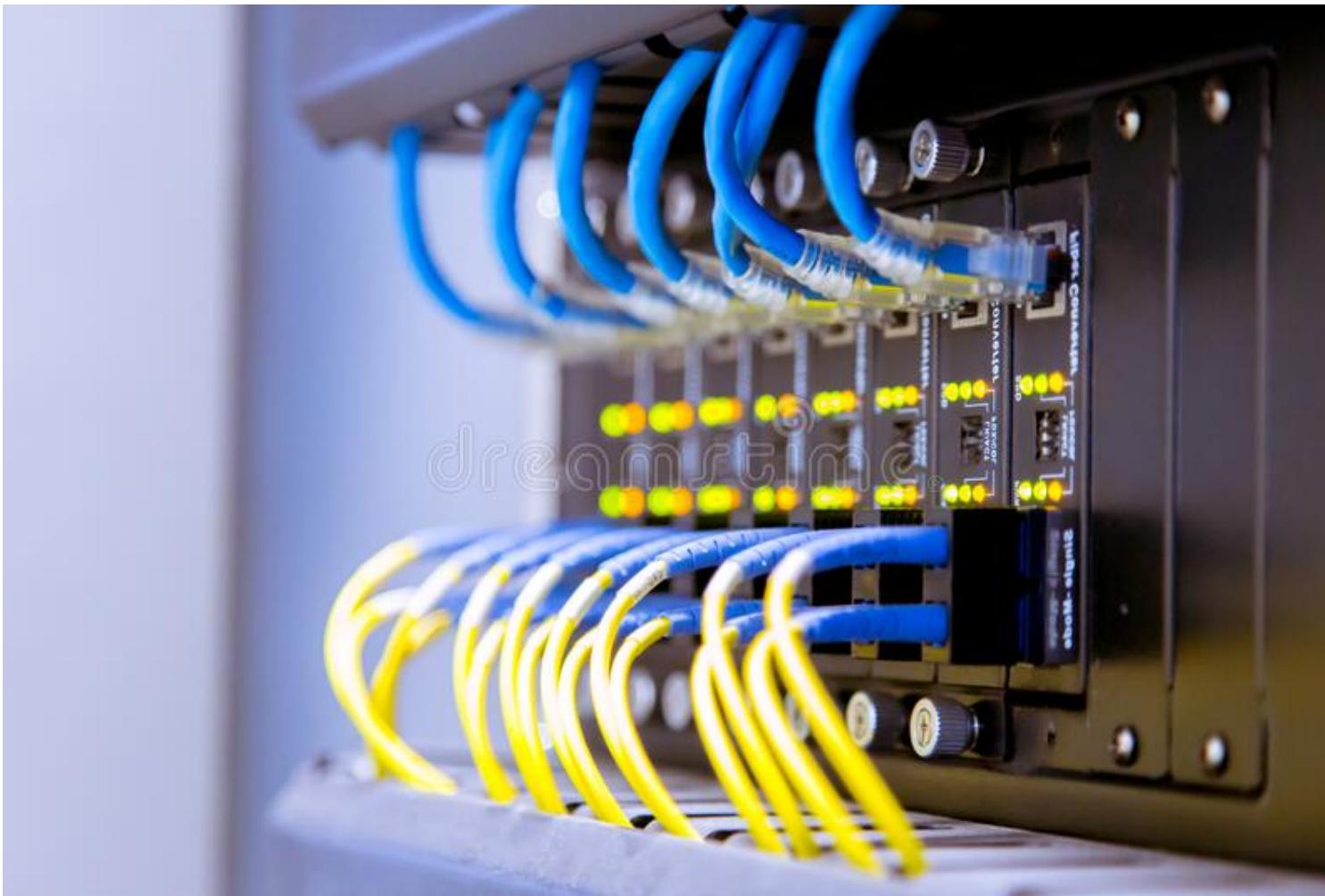
# TASTER ABFRAGEN

```
1 import time
2 import RPi.GPIO as GPIO
3
4 tast_pin = 17
5
6 GPIO.setmode(GPIO.BCM)
7 GPIO.setup(tast_pin, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
8
9 # Callback-Funktion
10 def ereignis(channel):
11     print("Taster gedrückt")
12
13 GPIO.add_event_detect(tast_pin, GPIO.FALLING, callback = er
14
15 try:
```

# TASTER UND LED KOMBINIEREN

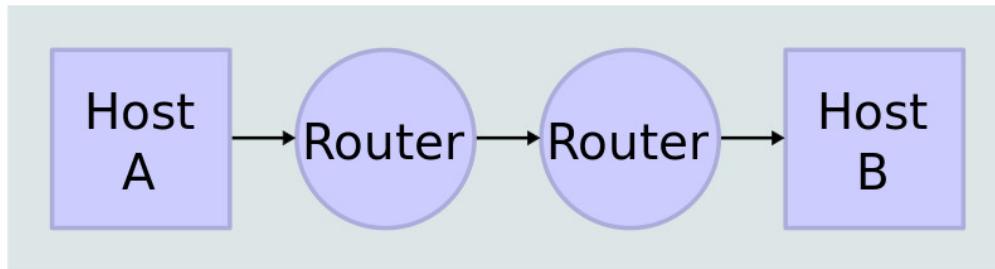
Aufgabe: Ändere das Programm so ab, dass die LED genau 3x blinkt wenn der Taster gedrückt wird.

# RASPBERRY PI IM NETZWERK



# NETZWERKSTRUKTUR

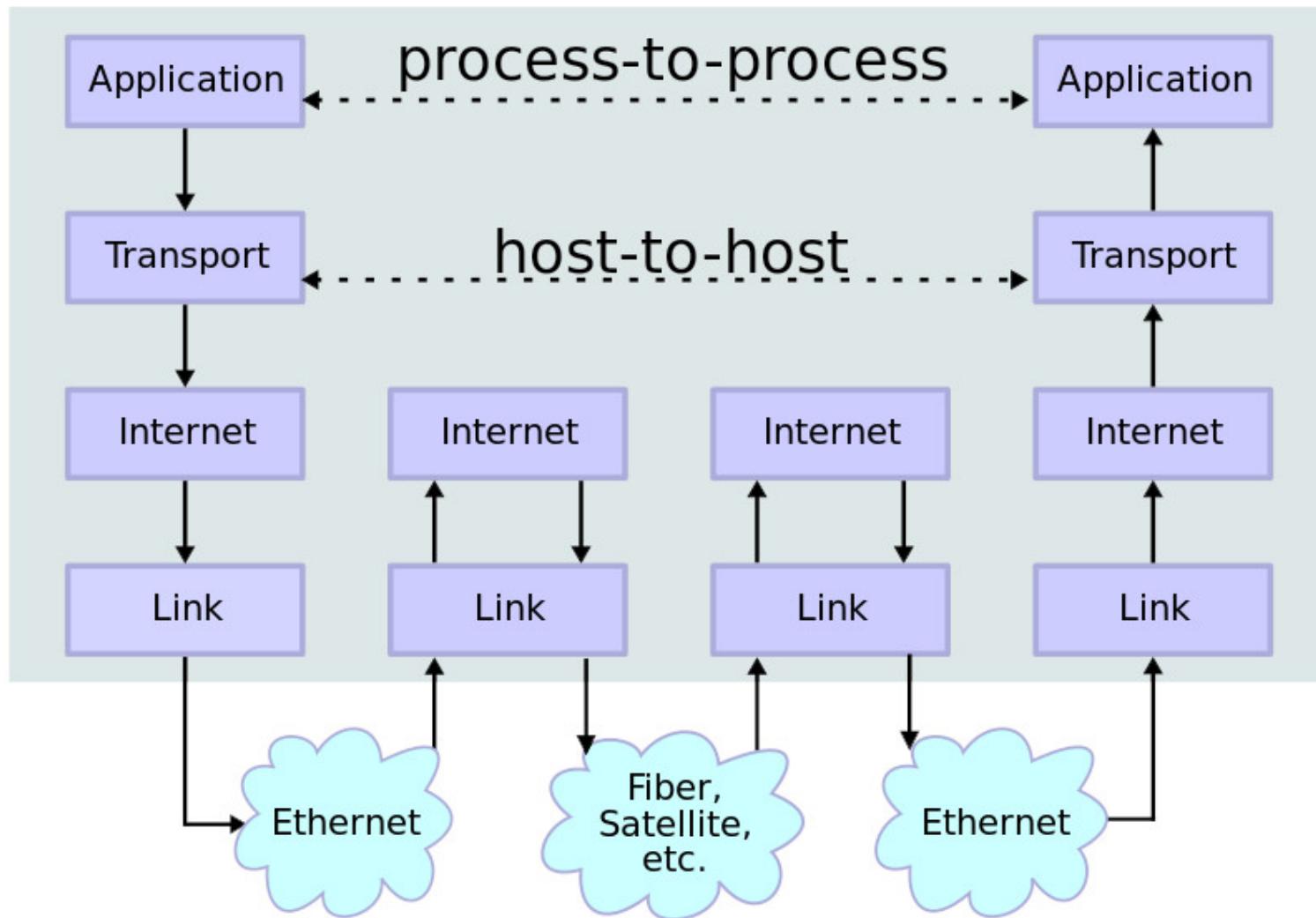
## Network Topology



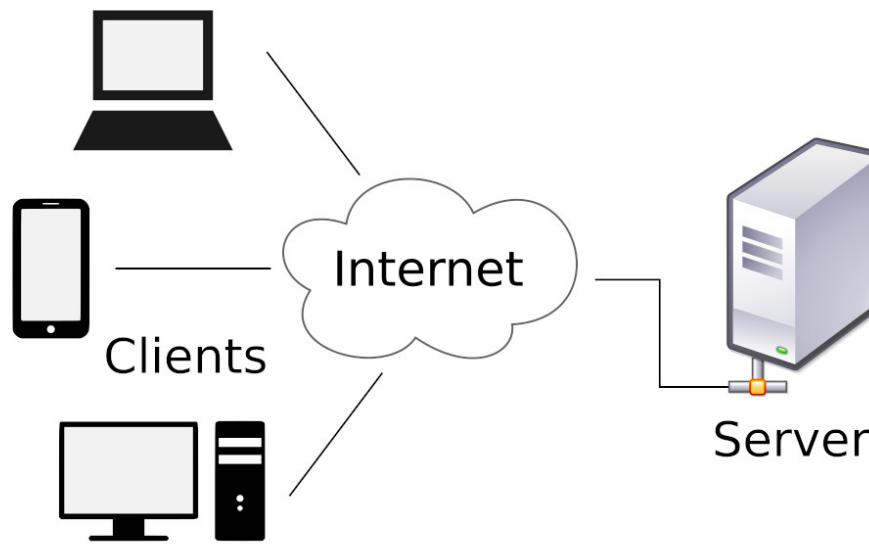
- Ende-zu-Ende-Kommunikation
- Paketorientiert
- Datenkanal zwischen zwei Teilnehmern

# NETZWERKSTRUKTUR

# Data Flow



# CLIENT-SERVER-ARCHITEKTUR



# WAS SIND PORTS?

- Ein Port ist ein "Teil" einer Adresse und umfasst den Wertebereich 0 bis 65535
- Zuordnung von Diensten zu Port-Nummern.
- Client und Server "binden" sich an jeweils einen Port, wenn eine Verbindung aufgebaut wird.
- Ports 0 bis 1023: System Ports
- Ports 1024 bis 49151: Registered Ports
- Ports 49152 bis 65535: Dynamic Ports

# NETCAT

## Server-Seite

```
netcat -l -p 50000
```

## Client-Seite

```
netcat X.X.X.X 50000
```

# FRAGE-ANTWORT-SPIEL - SERVER

```
1 import sys
2 import socket
3
4 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 sock.bind(("\"", 51000))
6 sock.listen(1)
7
8 while True:
9     print("Warte auf deinen Spieler ")
10    spieler, addr = sock.accept()
11    print("Spieler hat sich verbunden. Seine Adresse ist ", a
12    try:
13        while True:
14            print("Warte auf Frage...")
15            frage = spieler.recv(256)
```

# FRAGE-ANTWORT-SPIEL - CLIENT

```
1 import sys
2 import socket
3 import time
4
5 spieler = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6 print("Verbinde zum Spieler ...")
7 spieler.connect(("10.2.15.154", 51000))
8 print("Verbunden")
9
10 try:
11     while True:
12         frage = input("Tippe deine Frage ein und dann ENTER: ")
13         spieler.sendall(frage.encode())
14         print("Warte auf die Antwort")
15         antwort = spieler.recv(256)
```

# SONIC PI

Installation im Terminal:

```
sudo apt install sonic-pi
```

## Anleitung

[Live Coding \(Youtube\)](#)

[Live Coding von Sam Aaron \(Youtube\)](#)

# ABSCHLUSSPROJEKT

Ziel: Wir bauen ein Sensor, der in einem Winkel von 180 Grad nach dem kleinsten Abstand zum Sensor bestimmt.

Dazu befestigen wir auf einem Servomotor ein Ultraschallsensor. Damit kann der Ultraschallsensor um 180 Grad gedreht werden.

Es soll ein Programm geschrieben werden, dass die Umgebung abgesucht und im Anschluss den Sensor auf das Ziel richtet.

# AUFGABEN / FUNKTIONEN

1. Ansteuerung Servomotor
2. Ansteuerung Ultraschallsensor
3. Algorithmus zum scannen und positionieren

Bildet drei Gruppen, die jeweils ein Thema bearbeiten.  
Am Ende werden wir alles auf einem Raspberry Pi  
zusammenfügen.

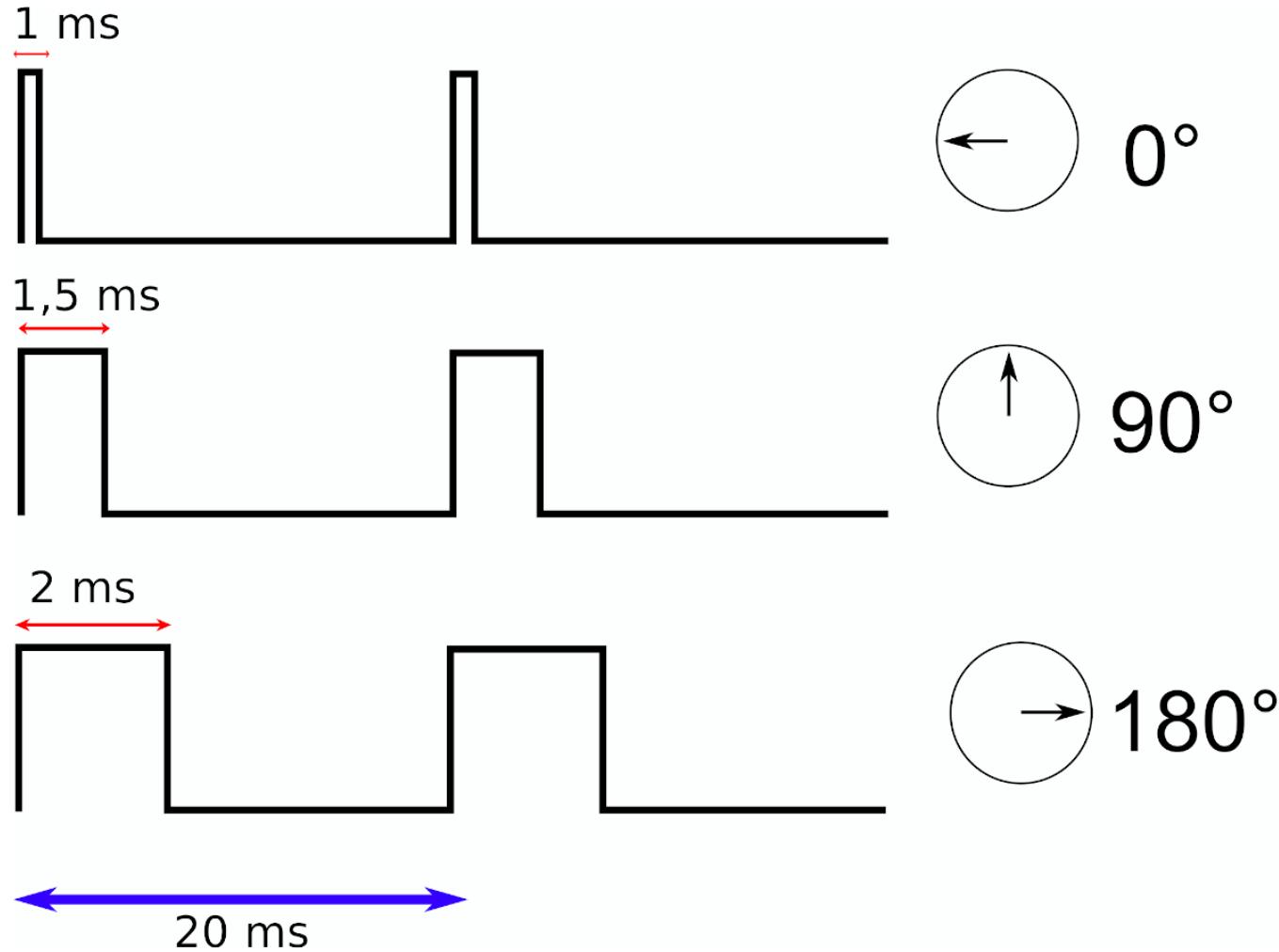
# GROBSTRUKTUR / ZUSAMMENARBEIT

```
1 def setze_position(winkel):
2     print("Neuer Winkel: ", winkel)
3
4 def messe_abstand():
5     return 0.0
6
7 def ermittle_winkel():
8     return -1
9
10 def programm():
11     winkel = ermittle_winkel()
12     if winkel >= 0:
13         print("Kleinster Abstand gefunden bei:", winkel)
14         setze_position(winkel)
15     else:
```

# **(MODELLBAU-)SERVOS ANSTEUERN**







Anleitung Ansteuerung

# ULTRASCHALLSENSOR HC-SR04

Funktionsweise

Anleitung