Raspberry Pi

R. Sitter 04.06.2020

www.sitter.de

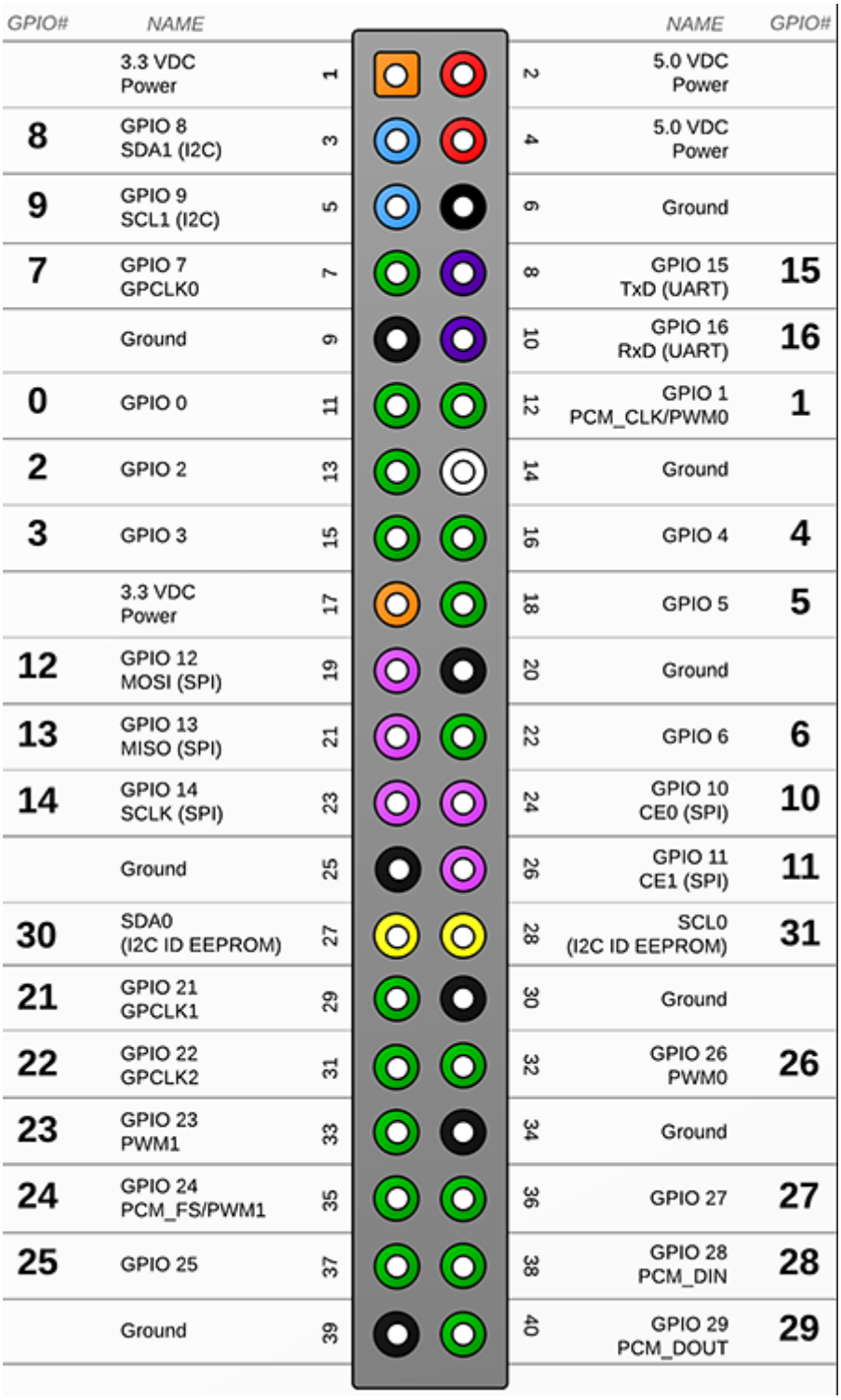
Inhaltsverzeichnis

0. Vorbemerkungen

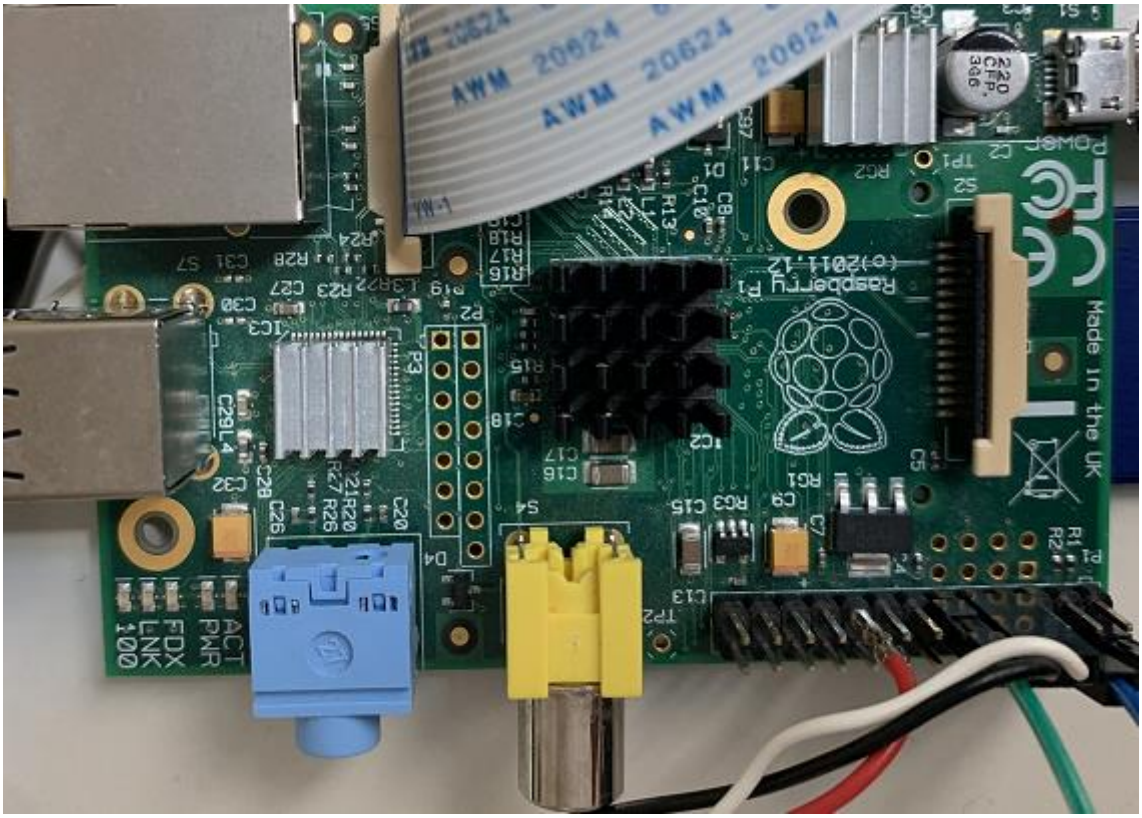
Die Raspberry Pi Mini-Computer (kurz Pi) sind im Vergleich zu den Mikrocontrollern z.B. von ATMEL natürlich reich mit Ressourcen und Peripherie ausgestattet.
Calliope und BBC micro:bit sind dagegen besser für Anfänger geeignet und haben nicht die Möglichkeit eines Computers (Tastatur und Bildschirm).
Auch der ARDUINO nimmt eher eine Zwischenposition ein. Beim ARDUNIO benötigt man mehr Wissen, kann aber auch mehr Peripherie anschließen und bauen.
Das Modul mit dem geringsten Umfang an Leistung ist der ESP8266, ein WLAN – Miniprozessor, der für die Hausautomation ideal geeignet ist und sich mit der Entwicklungsumgebung des ARDUIONO in C gut programmieren lässt.
Mit allen Mini-Computern kann man Hausautomation betreiben, kleine Roboter bauen oder Maschinen betreiben, die computergesteuerte Prozesse benötigen. Vor allem kann man mit den Mini-Computern viel lernen.
In der Regel geht es um Spannungswerte zwischen 3 und 5 Volt. Die Mini-Computer sind diesbezüglich natürlich empfindlich. Die Last an den Pins sollte eingehalten werden. In der Regel schadet eine LED mit Vorwiderstand nicht. Verschiedene Module, die z.B. für den ARDUINO angeboten werden, sind fast bei allen Mini-Computern einsetzbar. Die Steuerung von Servos und Motoren folgt fast immer den Regeln, wie sie bei den Mikrocontrollern gelten (sind dort beschrieben).

1. Raspberry Pi Hardware-Versionen

Die Größe der Platine variiert je nach Version von 3 cm mal 6,5 cm bis ca. 6 cm mal 9 cm.
Das Herz des Raspberry Pi ist ein ARM-Prozessor. Man unterscheidet die Versionen A (ohne LAN-Anschluss) und B (mit LAN).
Mittlerweile ist man bei Version 4. Die Entwicklung geht in Richtung mehr RAM, höhere Taktfrequenzen und bessere Peripherie bereits auf der Platine.
Eine spezielle Version ist der Raspberri Pi Zero, eine Minimalversion und den Zero W mit WLAN.
Die 1. Versionen haben alle noch die CPU ARM11 ab Version 2 dann der ARM Cortex-A. Innerhalb der Versionsnummern gibt es die Bezeichnungen A und
B (mit LAN-Anschluss) und Versionen mit + oder anderen Erweiterungen im Namen, welche in der Regel Verbesserungen oder Erweiterungen auf der
Platine bedeuten.
Je nach Ausführung hat der Pi (in normaler Größe oder Minimalausführung) eine oder mehrere USB-Schnittstellen , einen SD-Card Slot für das
Betriebssystem (SD oder Micro-SD), einen HDMI-Anschluss für einen Monitor, einen USB Mini Anschluss für die Stromversorgung mit 5 V, einen Kameraanschluss (CSI),
einen Composite Video Ausgang, Audio über Klinkenbuchse, einen internen Displayanschluss über Flachband und natürlich viele PINs (26 oder 40) für den Anschluss
von Sensoren und Aktoren. Der Pi hat keine Analoganschlüsse, wenn man von Pulse Width Modulation (PWM) absieht.
Der Pinout des Pi ist leider mit zwei Methoden belegt, die man nicht verwechseln darf (GPIO Board und GPIO BCM - Broadcom Anschlussnummer). GPIO steht für General Purpose Input Output.



Der Pi 1 hatte schon 2 USB-Buchsen, Audio, Video, HDMI, SD-Card-Slot und LAN-Buchse an Board. Dazu kommen zwei Steckleisten für Kamera und Display in Flex-Kabel-Ausführung. Die SD-Card hat den Slot auf der Rückseite der Platine.



2. Betriebssystem und Softwarekonfiguration

Als Software für den Raspberry Pi ist Raspberry Pi OS (früher Raspbian) die eindeutige Empfehlung. Dieses Betriebssystem kann man auf der Seite

https://www.raspberrypi.org

als Image-File herunterladen und auf eine bootfähige SD-Card bringen. Dabei wird eine Minimalvariante angeboten, die auch für den Pi 1 auf eine 2 GByte – SD – Card passt. Wenn man noch eigene Daten wie Bilder, Messwerte usw. hinzufügen möchte, wären trotzdem 4 GByte angebracht. Für die grafischen Betriebsysteme von Raspberry Pi OS sind mindestens 8 GByte erforderlich, besser sind hier 16 GByte.

Auch bei dem Betriebssystem selbst gibt es natürlich viele Alternativen (PiNet, RISC OS usw.). Diese kann ich zurzeit nicht beschreiben.

Raspberrypi.org bietet auch einen Imager an, also ein Programm, welches das Image auf eine SD-Card schreibt. Bei mir hat dies nicht auf Anhieb geklappt, deshalb habe ich ein Programm von Rufus genommen, was ich auch für andere Images benutzt habe.

Nun muss man nur noch die SD-Card in den Raspberry Pi stecken sowie Monitor (Fernseher), Tastatur und Maus anschließen und dann das Netzteil anstecken und schon sollte auf dem Bildschirm etwas zu sehen sein. Das linuxbasierte Raspberry Pi OS startet.

2.1. Arbeit mit der Konsole, erstes Linux

Wenn eine Tastatur und ein Monitor angeschlossen sind, kann man sofort beginnen.
Auf dem Bildschirm erscheint etwa so etwas:

pi@raspberrypi:~ $ gefolgt vom Cursor mit der Eingabeaufforderung

raspberrypi ist der Rechnername, pi der Benutzer und während der Installation des Betriebssystems hat man dem pi auch ein Passwort zugeteilt.

Für die Arbeit mit einem Terminprogramm, also des Fernzugriffs über Tablet, Smartphone usw. empfiehlt es sich, diesen Fernzugriff über eine Secure Shell einzurichten.

Auf der Konsole sind Linux-Befehle gültig. Diese sind beim Raspberry Pi ergänzt um Funktionen mit der besonderen Peripherie, also den Schnittstellen. Die Basis ist aber immer Linux. Man sollte ich also damit etwas vertraut machen. Im Anhang sind wichtige Linux-Befehle aufgelistet, sie alle aufzuführen, würde allerdings den Rahmen sprengen.

Wichtig ist z.B. der Befehl „ls“, zum Auflisten der Dateien eines Verzeichnisses bzw. „ls –l“ zum Auflisten der Dateien mit Parametern wie Datum, Rechte usw. Fast alle Linux-Befehle haben Optionen, die dem Befehl folgen, also im Beispiel „-l“ für „ls“. Die Optionen sind mit Leerzeichen getrennt und können mehrere sein. Eine Hilfe bzw. die Auflistung der möglichen Parameter bekommt man bei „ls“ mit „ls –help“.

Zum Wechseln zwischen Verzeichnissen benutzen Sie den Befehl „cd“ für change directory. gefolgt vom Pfad oder dem Verzeichnis. Mit „mkdir“ gefolgt von einem Verzeichnis-Namen kann ein Verzeichnis erstellt werden.

Es gibt verschieden Editoren für Dateien. Wir verwenden hier nur „nano“. Wenn Sie den Befehl „nano test.txt“ eingeben, öffnet sich der Editor, sie können etwas in die Datei schreiben und mit Ctrl X wieder den Editor verlassen und die Datei speichern. Mit „ls –l“ sollte die Datei dann auch im Verzeichnis zu sehen sein. Erstellen Sie eine Datei mit „sudo nano test1.txt“, so sehen Sie in der Auflistung, dass der Eigentümer der Datei root ist und nicht der Benutzer pi.

2.2. SSH-Zugang einrichten

Nutzen Sie dafür den Befehl „sudo raspi-config“, um den Raspberry Pi Konfigurator zu öffnen.
Je nach Version ist der Menüpunkt „Interfacing Options“ auszuwählen.
Navigieren Sie weiter bis zum SSH-Eintrag und bestätigen Sie die Eingabe mit „Enable“.

Bei älteren Versionen von Raspbian müssen Sie den Server manuell installieren.
Nutzen Sie den Befehl „sudo apt-get install ssh“. Starten Sie nun den Server mit „sudo /etc/init.d/ssh start“.
Einen automatischen Start aktivieren Sie mit „sudo update-rc.d ssh defaults“.

Wie verbinden Sie sich nun zu Ihrem Raspberry Pi? Zuerst müssen Sie Ihre IP-Adresse herausfinden. Nutzen Sie dazu den Befehl „ifconfig“. Diese finden Sie nun hinter dem Eintrag „inet addr:“ Öffnen Sie nun erneut das Terminal und geben Sie folgenden Befehl ein: „ssh pi@ [Ihre IP-Adresse]“.
Nun müssen Sie sich mit Ihrem Passwort einloggen. Nutzen Sie den Befehl „passwd [Passwort]“.
Sie können außerdem über SSH Dateien auf Ihrem Raspberry Pi verwalten. Dazu brauchen Sie lediglich ein Programm wie etwa „WinSCP“, um Dateien auf Ihrem Pi zu verschieben.

Nun könnte Tastatur und Maus auch vom Raspberry Pi abgebaut werden. Das kommt auf den Einsatz des Mini-Computers an.

3. Installation eines Webservers, mySQL und PHP auf dem Pi

3.1. Apache

Den Anfang dazu bildet die Installtion des Apache 2 HTTP Servers. Als aller erstes müssen die Pakete natürlich auf dem neusten Stand sein.

```
sudo apt-get update
sudo apt-get install apache2
```

Und schon ist die Installation fertig. Um zu prüfen, ob alles geklappt hat,
geben wir im Browser entweder http://raspberrypi/ (Achtung: keine Domain-Endung)
oder die interne IP-Adresse (192.168.y.xxx) ein.

3.2. PHP

Um nicht nur reines HTML anzeigen und den Content dynamisch gestalten zu können, benötigen wir PHP. Dazu richten wir die benötigten Pakete ein und testen die Installation auf Funktionstüchtigkeit.

```
sudo apt-get install php5
```

Wir wechseln also in das Verzeichnis: `cd /var/www/`
Hier erstellen wir eine neue Datei phpinfo.php
`sudo nano "phpinfo.php"`
und schreiben:

```
<?php
    phpinfo();
?>
```

Gespeichert wird wie immer mit STRG + O und mit STRG + X der Editor geschlossen.
Nun wird die Adresse http://raspberrypi/phpinfo.php aufgerufen und man sieht die Einstellungen zu PHP auf dem Webserver.

3.3. mySQL

Was wäre ein Webserver nur ohne eine Datenbank? Auf wenige Funktionalitäten beschränkt.
Daher beschreibe ich in diesem Teil die Installation von MySQL (MariaDB), was wohl am verbreitetsten ist.
Zuerst einmal laden wir die benötigten Pakete und bestätigen die Installation:
`sudo apt-get -y install mariadb-server-10.0 php7.3-mysql mariadb-client-10.0`
Danach wird eine Aufforderung zum Passwort auswählen kommen. Hier sollte natürlich ein sicheres Passwort gewählt werden, um ungewünschte Gäste raus zu halten. Im nächsten Tutorial werden wir es brauchen.
Danach das Passwort noch einmal bestätigen und weiter geht die Installation.
Nachdem alles eingerichtet wurde, muss der Pi neu gestartet werden.
`sudo reboot`

Danach werden wir zur besseren Übersicht und Verwaltung noch **phpMyAdmin** installieren
Nach der Installation von Apache, PHP und MySQL ist phpMyAdmin an der Reihe. Dieser Schritt ist nicht unbedingt nötig, aber für das einfachere Auslesen der Daten im Browser sehr nützlich.
Es bietet eine einfache Benutzeroberfläche, wobei in diesem Teil die Installation, Einrichtung und Benutzung gezeigt wird.
Zwar würde der Webserver auch ohne phpMyAdmin auskommen, aber zum Anzeigen der Datensätze ist dieser ganz praktisch.
Los geht es wieder mit dem herunter laden und installieren der Pakete
`sudo apt-get install php5-mysql libapache2-mod-auth-mysql phpmyadmin`
Nach der Bestätigung wählen wir apache2 aus (mittels Leertaste und Enter) und setzen es fort.
Daraufhin wird gefragt, ob einige Datenbanken, die phpMyAdmin benötigt erstellt werden sollen. Mit bestätigen und weiter.
Als nächstes wird nach dem Passwort, welches für den root-User angelegt wurde gefragt:
phpMyAdmin Passwort festlegen
Jetzt wird noch nach einem Passwort zum Einloggen in phpMyAdmin gefordert.
Nach der Eingabe und Bestätigung muss Apache mit phpMyAdmin noch verknüpft werden. Dazu bearbeiten wir die Datei /etc/php5/apache2/php.ini
`sudo nano /etc/php5/apache2/php.ini`
Ganz am Ende der Datei fügen wir *extension=mysql*.so ein.
Dann nur noch mit STRG + O und STRG + X speichern und beenden.
Das war es bereits. Ab jetzt kann man sich unter http://raspberrypi/phpmyadmin/ (bzw. 192.168.y.xxx) mit den zuvor eingegebenen Passwort für den phpMyAdmin einloggen.
Der Standardbenutzername ist root.
Fertig. Jetzt können auch über das Interface Datenbanken erstellt und verwaltet werden.

3.4. Webserver über das Internet verfügbar machen

Eigentlich ist der Webserver verfügbar, sobald im Router die Portweiterleitung erfolgt ist. Nur muss man nun die IP-Adresse des Routers kennen, an der der Raspberry Pi hängt. Diese kann man auch

im Router sehen. Leider wechselt der Provider oft die IP-Adresse. Hier kann man im Router einen DNS-Service-Provider hinterlegen, der eine Domain in die IP-Adresse wandelt.

No-IP ist so ein Provider (kostenlos). Also legen wir einen kostenlosen Account unter https://www.noip.com/sign-up an. Der einzige „Nachteil“ der Free Version ist, dass jeden Monat eine Mail kommt, in der man aufgefordert wird den Account zu bestätigen.
Nach dem Login klicken wir auf Add a Host und wählen einen Hostnamen und als Domain eine der weiter unten gelisteten No-IP Free Domains. Als Host Type nehmen wir DNS Host (A).

4. nützliche Eistellungen am Pi

4.1. Autostart

Immer wieder passiert es, dass man Programme installiert, aber diese nicht automatisch beim Hochfahren starten. Um den Raspberry Pi Autostart nutzen zu können, braucht man lediglich die Informationen in der /etc/rc.local Datei im Linux System zu hinterlegen.
Als erstes muss im Verzeichnis /etc/init.d/ ein Skript erstellt werden, mittels welchem das Programm gestartet wird, daher erstellen wir ein Skript (es muss nicht unbedingt eine Datei-Endung haben)
`sudo nano /etc/init.d/NameDesSkripts`
mit folgendem Inhalt:

```
#! /bin/sh
### BEGIN INIT INFO
# Provides: noip
# Required-Start: $syslog
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: noip server
# Description:
### END INIT INFO

case "$1" in
  start)
    echo "noip wird gestartet"
    # Starte Programm
    /usr/local/bin/noip2
    ;;
    stop)
    echo "noip wird beendet"
    # Beende Programm
    killall noip2
    ;;  *)
    echo "Benutzt: /etc/init.d/noip {start|stop}"
    exit 1
```

Anstelle von noip2 kann hier natürlich auch jedes andere installierte Programm stehen, aber sei vorsichtig, dass auf keine Benutzerinteraktion gewartet wird (wie das Bestätigen bei apt-get), da es im schlimmsten Fall dazu kommt, dass beim Booten auf die Eingabe gewartet wird und der Pi nicht startet.
Als nächstes weisen wir die benötigten Rechte zu (Lesen & Schreiben)
`sudo chmod 755 /etc/init.d/NameDesSkripts`
und testen das Skript indem wir es starten
`sudo /etc/init.d/NameDesSkripts start`
und gleich wieder stoppen:
`sudo /etc/init.d/NameDesSkripts stop`
Damit das Skript beim Booten auch aufgerufen wird, führen wir folgendes aus:
`sudo update-rc.d NameDesSkripts defaults`
Nun sollte das Programm bei starten auch ausgeführt werden. Sollte eines Tages das Script nicht mehr benötigt werden, kann es aus dem Autostart entfernt werden mit:
`sudo update-rc.d -f NameDesSkripts remove`

4.2. Timetable

Eine andere Option zum Starten eines Skripts oder Programms ist „Cron“. Dadurch ist es möglich einen Befehl (der ein Aufruf eines Programms o.ä. sein kann) zu einem bestimmten Zeitpunkt zu starten. Der Zeitpunkt kann dabei entweder z.B. um die gleiche Uhrzeit am Tag sein oder aber nach dem Hochfahren des Systems. Cron bietet sehr viele Anpassungsmöglichkeiten.

```
crontab -e
#-------------------------------------------------------------
# Shell variable for cron
SHELL=/bin/bash
# PATH variable for cron
PATH=/usr/local/bin:/usr/local/sbin:/sbin:/usr/sbin:/bin:/usr/bin:/usr/bin/X11
#M   S     T M W  Befehl
#-------------------------------------------------------------
5    9-20 * * *   /home/BENUTZERNAME/script/script1.sh > /dev/null
*/10 *    * * *   /usr/bin/script2.sh > /dev/null 2>&1
59   23   * * 0,4 cp QUELLDATEI ZIELDATEI
*  *    * * *   DISPLAY=:0 LANG=de_DE.UTF-8 zenity --info --text "Beispiel für das Starten eines Programms mit GUI"
0    0    * * *   backup
#-------------------------------------------------------------
```

Im ersten Abschnitt stehen die Angaben für die Variablen $PATH und $SHELL. Im unteren Teil dann die einzelnen Cronjobs. In einer Zeile kommen zuerst die fünf Felder für die Zeiten (Minute, Stunde, Tag, Monat, Wochentage) und danach der auszuführende Befehl. In diesem Beispiel werden die Befehle fünf Minuten nach jeder vollen Stunde zwischen 9 und 20 Uhr (also 9:05, 10:05, ..., 20:05) script1.sh ausgeführt.
Aller 10 Minuten wird script2.sh ausgeführt.
Jeden Sonntag und Donnerstag um 23:59 eine Kopie erstellt.
Jede Minute ein Programm mit GUI, das die Display- und die Sprach-Variable benötigt aufgerufen und jeden Tag Punkt Mitternacht 00:00 Uhr ein Backup ausgeführt.
Wichtig ist, dass am Ende der Tabelle ein Kommentar oder eine Leerzeile steht. Ähnlich wie die fstab muss die crontab mit einer Leerzeile enden!

4.3. User einrichten

Dazu wechseln wir erst einmal das Verzeichnis.

```
cd /etc/proftpd/
```

Hier soll der virtuelle Benutzer erstellt werden (z.B. ein FTP-Nutzer). Im folgenden Beispiel erstellen wir den Benutzer tutorials mit dem Homevezeichnis /var/www/
(falls das Rootverzeichnis von Apache geändert wurde, sollte dies natürlich angepasst werden. Gleiches gilt, wenn ein FTP-User auf andere Verzeichnisse Zugriff haben soll).

```
sudo ftpasswd –passwd  --name tutorials --gid 33 --uid 33 --home /var/www/ --shell /bin/false
```

Nun nur noch das Passwort eingeben und bestätigen. Falls das Passwort des Benutzers zu einem späteren Zeitpunkt geändert werden soll, einfach wieder in das Verzeichnis wechseln und den Befehl erneut ausführen.

```
chmod g+s /var/www
chmod 775 /var/www
```

4.4. USB-Stick mounten

Zuerst sollte man den USB-Stick an einen USB-Port stecken und sich dann über das Filesystem informieren:
`sudo lsblk` es folgt das mounten des Mediums (z.B. sda1):

```
sudo mount /dev/sda1 /media/usb
```

und möglicher weise Aktionen wie das Kopieren:

```
cp -r /media/usb /var/www/html
```

5. Python

Python ist eine Script-Sprache ähnlich wie PHP oder Javascript. Ein Python-Interpreter ist für die Ausführung des Codes verantwortlich, der als einfache Textdatei vorliegt. Für Einsteiger ist die Sprache besonders geeignet, weil sie nur wenige Schlüsselwörter umfasst und der Code sich relativ

übersichtlich gestalten lässt. Gleichzeitig zwingt die geforderte Programmstruktur zum sauberen Programmieren, weshalb man als Quereinsteiger gerne über die eine oder andere Besonderheit stolpert.
Die Programmstruktur wird durch Einrückungen mit Leerzeichen oder Tabulatorzeichen gebildet. Andere Sprachen verwenden dazu Klammern oder Schlüsselwörter.

Falls noch nicht geschehen sollten nun die Python-Entwicklungstools installiert werden. Diese brauchen wir später sicher für die Programmierung mit GPIO.

```
apt-get update apt-get install python-dev
```

Im günstigsten Fall sind die schon installiert gewesen, ansonsten bestätigen wir alle Abfragen mit "Enter" also der vorgegebenen Standardantwort.

Da die Python GPIO-Bibliothek in die Installationsquellen von Raspberry Pi OS aufgenommen wurden, können wir einfach mit apt-get installieren:

```
apt-get install python-rpi.gpio
```

Es werden auch evtl. fehlende Abhängigkeiten aufgelöst, deren Installation in einigen Fällen bestätigt werden muss. Jetzt kann die Bibliothek verwendet werden.

Zuerst öffnen wir einen Editor und legen eine Datei mit dem folgenden Inhalt an:

```
nano helloworld.py
#!/usr/bin/python
print ("Hello World")
```

Mit Strg + O, Return und Strg + X die Datei speichern und schließen.
Entweder
`sudo python helloworld.py`
aufrufen oder vor dem Ausführen die Datei ausführbar machen:
`chmod +x helloworld.py`

```
./helloworld.py
```

6. gcc

Im Gegensatz zu Python wird in c++ oder gcc der Code kompiliert und maschinenfähig gemacht:

```
nano hello.c
#include<stdio.h>

int main() {
        printf("Hello World\n");
        return 0;
}
```

Mit Strg + O, Return und Strg + X die Datei speichern und schließen.
Kompilieren:

```
gcc -o hello hello.c
./hello
```

7. externe Hardware anschließen und programmieren

Eine Beschreibung der Schaltungen externer Hardware findet man in mikrokontroller.pdf. Der Raspberry Pi sollte auf 3,3 V berechnet werden. Bei low current leds kann man 1 kOhm als Vorwiederstand zum Testen benutzen, dann passiert nichts. Bitte aber auch den Gesamtstrom im Blick haben!
Als Tipp: jede **LED** hat ein längeres und ein kürzeres Ende. Das längere gehört zum Positiven (3,3V), das kürzere zum Negativen (GND). Als Vorwiderstand bekommt die (normale) LED 470 Ω.

Bevor du die Kabel ans Pi anschließt, überprüfe noch einmal die Schaltung.
Nachdem alles aufgebaut und angeschlossen ist, brauchen wir ein Skript, dass die LED blinken lässt. Wir erstellen ein Skript

```
sudo nano ampel_skript1.py
```

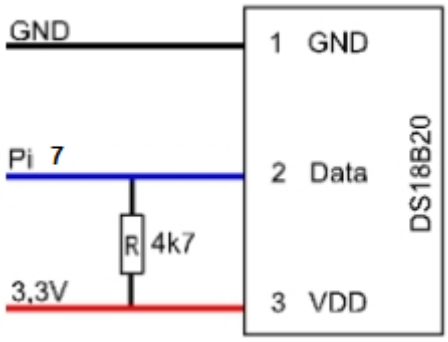
mit folgendem Inhalt:

```
#Bibliotheken einbinden
import RPi.GPIO as GPIO
import time

#GPIO Modus (BOARD / BCM)
GPIO.setmode(GPIO.BOARD)

#Richtung der GPIO-Pins festlegen (IN / OUT)
GPIO.setup(26, GPIO.OUT)

#unendliche Schleife
while True:
    #Pin 26 HIGH Pegel
    GPIO.output(26, True)
    #eine halbe Sekunde warten
    time.sleep(0.5)
    #Pin 26 LOW Pegel
    GPIO.output(26, False)
    #eine halbe Sekunde warten
    time.sleep(0.5)
```

Achtung: Hier ist die Nummer des Pins wichtig und nicht die GPIO Nummer. Die Pin-Nummer ist 26, wobei es GPIO Pin 7 ist. Falls die GPIO Nummer angegeben werden soll, muss in Zeile 6 stehen: GPIO.setmode(GPIO.BCM)
Wir speichern und beenden den Editor (STRG + O, STRG + X) und führen das Skript aus.

```
sudo python ampel_skript1.py
```

Den **Temperatursensor DS18B20** ist ein 1-Wire Sensor, der nur einen Anschluss benötigt und darüber die Adresse und die Daten geschickt werden.



Wenn alles entsprechend verkabelt ist, können wir das 1-Wire Protokoll damit aktivieren:

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
```

Und wir fügen noch eine Zeile hinzu (Pin7 ist GPIO Pin 4):

```
sudo nano /boot/config.txt
```

Dies wird ans Ende gepackt:

```
dtoverlay=w1-gpio,gpioin=4,pullup=on
```

Speichere mit STRG+O und schließe mit STRG+X.

Ob es geklappt hat, können wir herausfinden, indem wir folgendes eingeben:

```
lsmod
```

Die Module müssten nun aufgelistet sein, falls nicht wird ein anderer GPIO Pin als 4 benutzt oder es trat ein Fehler beim Aktivieren auf.

Damit nicht bei jedem Start die Module geladen werden, tragen wir sie in die Datei /etc/modules ein:

```
sudo nano /etc/modules
```

und fügen als letztes die folgenden zwei Zeilen ein:

```
w1_gpio
w1_therm
```

Für den nächsten Schritt benötigen wir als erstes die ID des Sensors. Falls du vorhast mehrere in Reihe anzuschließen, teste am besten jeden einzeln und notiere dir die ID, damit du sie später nicht verwechselst.

Wir wechseln das Verzeichnis und geben uns die Dateien aus

```
cd /sys/bus/w1/devices/
ls
```

Eine der Dateien heißt *10-000802b4ba0e* (bei dir anders) und ist die ID, mit der wir den Sensor abfragen (bitte ID anpassen):

```
cat /sys/bus/w1/devices/10-000802b4ba0e/w1_slave
```

In der Ausgabe sehen wir als letzte Angabe die Temperatur (in „Milligrad“)

```
31 00 4b 46 ff ff 05 10 1c : crc=1c YES
31 00 4b 46 ff ff 05 10 1c t=24437
```

Durch 1000 geteilt macht das 24.437°C.

Als Python-Programm ist das natürlich komfortabler:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# 1-Wire Slave-Liste lesen
file = open('/sys/devices/w1_bus_master1/w1_master_slaves')
w1_slaves = file.readlines()
file.close()

# Fuer jeden 1-Wire Slave aktuelle Temperatur ausgeben
for line in w1_slaves:
  # 1-wire Slave extrahieren
  w1_slave = line.split("\n")[0]
  # 1-wire Slave Datei lesen
  file = open('/sys/bus/w1/devices/' + str(w1_slave) + '/w1_slave')
  filecontent = file.read()
  file.close()

  # Temperaturwerte auslesen und konvertieren
  stringvalue = filecontent.split("\n")[1].split(" ")[9]
  temperature = float(stringvalue[2:]) / 1000

  # Temperatur ausgeben
  print(str(w1_slave) + ': %6.2f °C' % temperature)
```

Die Ausgabe dazu sieht bei mir so aus:

```
10-000802bf634d:  24.25 °C
10-000802cfb15d:  24.38 °C
```

Da der Pi keine Analogsignale verarbeiten kann, bietet es sich an, eigene Mikrocontroller oder den **ARDUINO** an den Pi anzuschließen. Beim ARDUINO kann man die serielle Schnittstelle über USB benutzen und darüber gleich den ARDUINO mit Strom versorgen. Am ARDUINO kann z.B. die Helligkeit gemessen und seriell an den Pi übertragen werden. Ein Python Programm liest die Daten und bringt sie auf den Bildschirm:

```
#!/usr/bin/env python
import serial
import time

s = serial.Serial('/dev/ttyUSB0',9600)
if (s.isOpen() == True):
```

```
        s.close()
s.open()
time.sleep(5)

s.write("test")
try:
        while True:
                responce = s.readline()
                print(responce)
except KeyboardInterrupt:
        s.close()
```

8. nützliche Linux-Befehle am Pi:

Bild mit der Kamera machen:
raspistill -o /var/www/html/bild.jpg
Prozess im Hintergrund starten
Python Test.py &
top - Prozesse mit q beenden
bg letzten Befehl in den Hindergrund
bf letzten Befehl in den Vordergrund
ps - laufende Prozesse auflisten -f, -ef
kill - Prozess beenden (Prozessnummer angeben) bzw. killall
Befehle auch nach ssh noch aktiv :
nohup
Superuser benutzen:
sudo
Passwort ändern:
passwd
Nutzer hinzufügen:
adduser
Dateisystem anzeigen:
df –h
Verzeichnisgrössen in Sum:
du -s
ausschalten:
sudo shutdown -h now
Neustart:
sudo reboot
Shell-Script aufrufen:
./xxx.sh
Verschiedenes:
alias lsb="ls -l"
cat led1.py | wc -w - zählt Wörter
cat led1.py | less - Zeilenweise anz.
sudo chmod +x test.sh - ausführbar
ifconfig - Netzwerk Daten anzeigen
grep - Zeichenketten in Dateien finden

9. Quellenangaben:

https://www.elektronik-kompendium.de
https://tutorials-raspberrypi.de
https://py-tutorial-de.readthedocs.io/de/python-3.3/introduction.html
https://de.wikipedia.org/wiki/Raspberry_Pi
https://www.berrybase.de
http://www.netzmafia.de