



Aus der Vortragsreihe des OV Meschede :
„Arduino und Co.“

Hier : Raspberry Pi

© Heribert, DK2JK und Josef, DL8DBN

RaspberryPi



(c) Michael H. („Laserlicht“), CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=80140656>

26.10.22

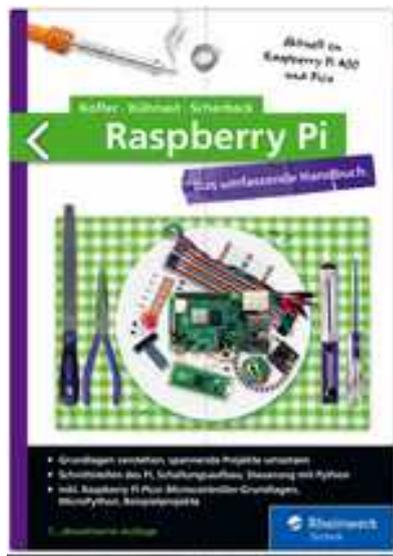
Arduino &Co

2/16

Dieses Board ist praktisch ein kleiner PC; es fehlen nur noch Tastatur und Bildschirm; ansonsten verhält er sich ähnlich wie ein PC. Das Betriebssystem ist Linux. Gegenüber einem PC hat er den Vorteil, dass weitere Pins zum Steuern zur Verfügung stehen, also Ein- und Ausgänge, über die man z.B. Relais oder Leuchtdioden ansteuern kann .

Für den Bastler sind natürlich die digitalen Ein- und Ausgänge von großem Interesse.

Nachschlagewerk



1088 Seiten, 7., aktualisierte Auflage 2021,
gebunden, in Farbe
Rheinwerk Computing, ISBN 978-3-8362-
8351-9

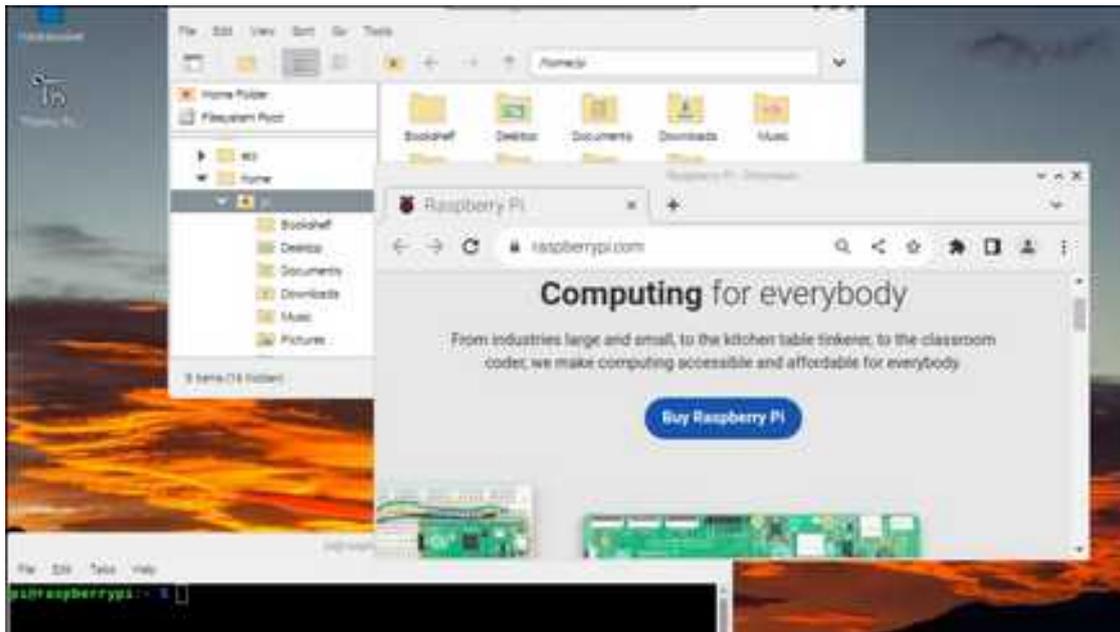
26.10.22

Arduino & Co

3/16

Ein zu empfehlendes Nachschlagewerk

Linux Oberfläche



26.10.22

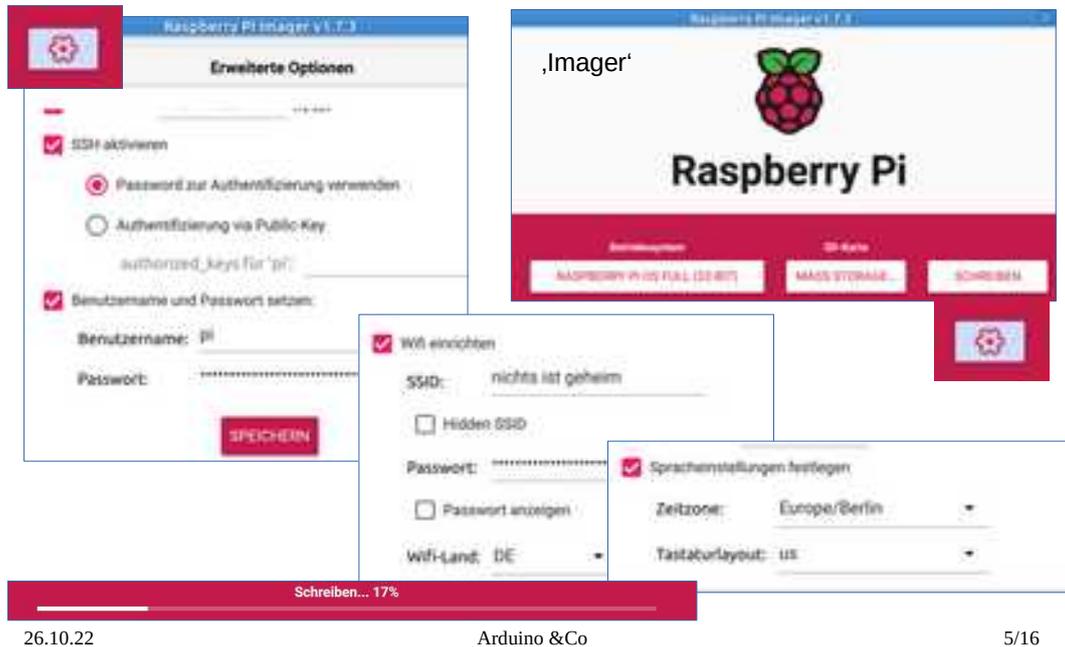
Arduino &Co

4/16

Dies ist die Desktop -Ansicht des Raspberry Pi; einige Apps kann man schon wie bei Windows verwenden (Dateimanager, Browser, Terminal, etc).

Die Fernbedienung geschieht über VNC Viewer, wodurch die Oberfläche des ‚Raspi‘ mit exakt demselben Bildschirm auf dem PC angezeigt wird. Dies muss in der Konfiguration unter Punkt ‚13‘ VNC enable eingestellt werden.

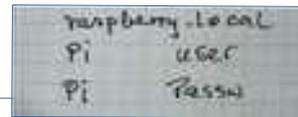
Installation mit ‚Imager‘



In den meisten Fällen will man den Raspberry Pi aber nicht als PC benutzen, sondern als eine freie programmierbare Platine, die auch Pins zum Steuern hat und WLAN als Netzwerkverbindung.

Man gibt schon beim Schreiben des Systems auf SD-Karte mittels ‚Imager‘ die Zugangsdaten für das heimische WLAN an und dass man den ‚Raspi‘ fernbedienen möchte.

Erster Kontakt



```
dk2jk@samsung-laptop:~$ ping raspberrypi.local -c 1
PING raspberrypi.local (192.168.178.21) 56(84) Bytes Daten:
64 Bytes von raspi21.fritz.box (192.168.178.21): icmp_seq=1 ttl=64 Zeit=5.01 ms

--- raspberrypi.local ping statistics ---
1 Pakete übertragen, 1 empfangen, 0% Paketverlust, Zeit 0ms
rtt min/avg/max/mdev = 5.011/5.011/5.011/0.000 ms
```

```
dk2jk@samsung-laptop:~$ ssh 192.168.178.21 -l pi
pi@192.168.178.21's password: **
Linux raspberrypi 5.15.61-v7+ #1579 SMP Fri Aug 26 11:10:59 BST 2022 armv7l
pi@raspberrypi:~$
```

```
pi@raspberrypi:~$ sudo raspi-config
```



26.10.22

Im Terminal, bei Windows: CMD

1. Test:

```
ping raspberrypi.local
```

Hurra! → Raspi ist im Netz

2. Test:

```
ssh 192.168.178.21 -l pi
```

Der Prompt hat sich in pi@raspberrypi:~ \$

Geändert: → wir sind auf dem Linux Terminal
des Raspi gelandet

3. Schritt:

```
sudo raspi-config
```

--→ weitere Einstellungen

Raspi-Anwendung: DB0QH Anzeige



<http://dmrnsk.spdns.de:59876/ui/>

26.10.22

Arduino & Co

7/16

So sieht die DB0QH- Status- Anzeige aus.

Die Messwerte werden durch Python-
Programme geholt.

Die Anzeige-Elemente wurden mit NodeRed -
Funktionsblöcken programmiert

Beispiel NodeRed DB0QH



26.10.22

Arduino &Co

8/16

Node-RED ist ein flussbasiertes Entwicklungstool für die visuelle Programmierung.

Node-RED bietet einen Webbrowser- basierten Flow-Editor, der auf dem Raspi läuft.

Man hat Bausteine, die zu verbinden sind. Links ein Sekundentakt. In der Mitte: Daten holen und anpassen. Rechts in Blau : Anzeige-Elemente.

Linux Terminal

```
pi@db0qh-svx:~/python_p $ cd ..
pi@db0qh-svx:~$ ls
build  usr  de_DE  downloads  multimonSRG  nul  python_p  rtl-sdr  swalink  xx.txt
pi@db0qh-svx:~$ cd python_p
pi@db0qh-svx:~/python_p $ date
Mon 24 Oct 17:29:46 CEST 2022
pi@db0qh-svx:~/python_p $ python cpu_temperatur.py
CPU-Temperatur: 59.6°C
Betriebssystem: Raspbian GNU/Linux 10 (buster)
pi@db0qh-svx:~/python_p $ python get_temp2.py
30.9
pi@db0qh-svx:~/python_p $ ls *.py
abgleich.py          get_env_temp.py    rk.py
AD81x15.py           get_hp.py          sensordaten_direkt_lesen.py
am2320_i2c.py        get_p.py           setup.py
hmc280.py            get_ptt.py         strom.py
cpu_temperatur.py   get_swert.py       test.py
cpu_werte.py         get_swr.py         test_ramdisk.py
doc.py               get_temp2.py       test_strom.py
ds1820.py            halt.py            ts_db0qh.py
ds1820_v2.py         my_adc.py          ts_luftdruck.py
ds18b20_sensor_2.py pcf8574_raspi.py  watchgpio.py
get_cpu_temp.py     read_abgleich.py  webPageMain.py
pi@db0qh-svx:~/python_p $
```

cdchange directory

ls list directory

date.... Zeit und Datum

python xxx.py ... python Script starten

Beispiele; Lesen der CPU-Temperatur und
Lesen der Schrank-Temperatur

Python: Die „Muttersprache“ des Raspi

Python -Script:

```
pi@raspberrypi:~ $ cat app/countdown.py
#countdown.py

from time import sleep

start=4
ende=-1
for n in range (start,ende,-1):
    sleep(.5)
    print(n)
print( 'lift off !')
```

Das kommt heraus:

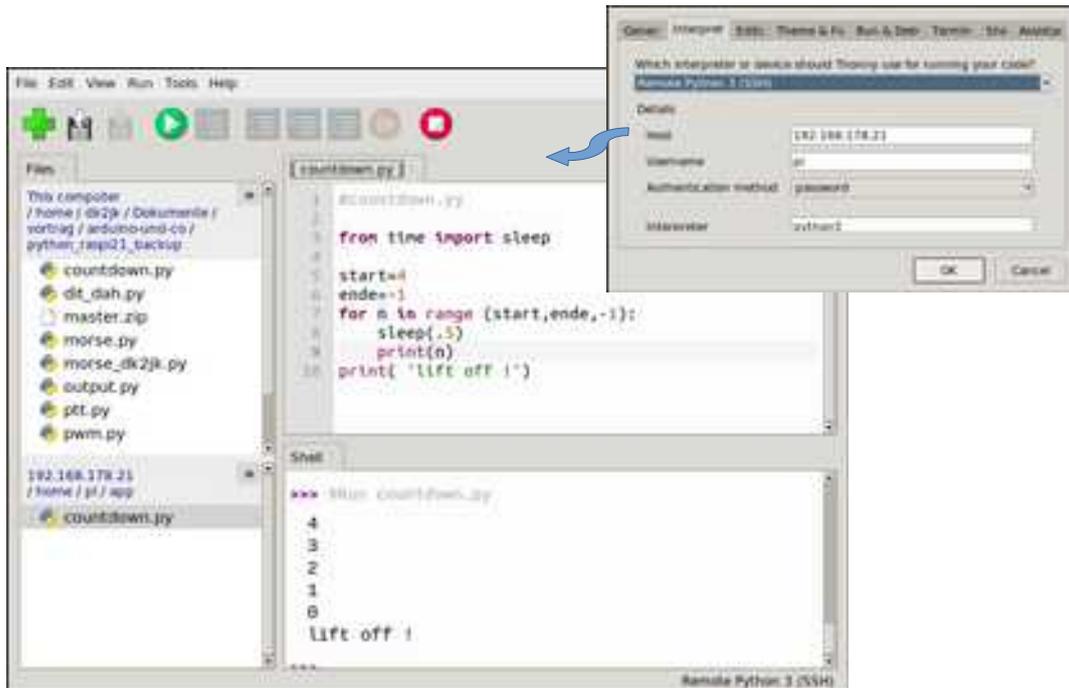
```
pi@raspberrypi:~ $ python
app/countdown.py
4
3
2
1
0
lift off !
```

Hier ein einfaches Python- Script, im Terminal angezeigt.

Start des Python- Scripts im Terminal.

(Dieses Python- Script würde auch auf einem Windows-PC laufen)

Programmieren mit Thonny



Das gleiche Python-Script wie auf der letzten Seite.

Jedoch: jetzt alles in einem Fenster und per Mausklick bedienbar.

Man beachte: Thonny läuft hier nicht auf dem Raspi, sondern auf dem PC; der Raspi ist fernbedient via SSH.

GPS entschlüsselt

```
>>> Run gps_decoder3.py
*data.txt*...   Eingangsdaten vom GPS-Empfänger ( hier aus Datei)
$abode,xx0544
$GPRMC,222815.000,A,5128.9775,N,00819.3163,E,4.10,000619,0.0,A*7D
0x72...

ID ... $GPRMC   Message ID Symbol   Gesuchtes Schlüsselwort
1 ... 222815.000  off of position   Code für die Zeit
2 ... A   Status-Arbeitsmodus
3 ... 5128.9775  latitude
4 ... N
5 ... 00819.3163  longitude
6 ... E
7 ... 4.10   speed over the ground in knots
8 ...
9 ... 000619   Date
Uhrzeit: 22:28:15
km/h: 7.59   Ergebnis
```



Die üblichen GPS-Empfänger geben ein leicht lesbares ASCII-Protokoll aus.

Hier werden die Daten auseinander gepflückt.

Python Script: GPS Decoder

```
"nmea.txt"....
$abcde,xxx566
$GPRMC,222615.000,A,5128.9775,N,00819.3163,E,4.10,,060619,,A*7D
$xyz...
```

```
gps_decoder2.py
1 #gps-decoder
2 with open("nmea.txt") as f:
3     data = f.read()
4 print( data)
5 fundstelle=(data.find("$GPRMC")) # suche schlüsselwort
6 RMC_str= data[fundstelle : -1] # fundstelle bis ende
7 listen_werte= RMC_str.split(',') # komma separierte werte in liste schreiben
8 for i in range (0, 10): # eine Anzahl listenwerte anzeigen
9     print( i, "...",listen_werte[i])
10 zeit= listen_werte[1] # listenwert Nr. 1 ist die zeit
11 zeit_formatiert= zeit[0:2]+':'+zeit[2:4]+':'+zeit[4:6]
12 print("Uhrzeit:",zeit_formatiert)
13 knoten= float(listen_werte[7]) # Nr. 7, String nach fließkomma umwandeln
14 kmh= knoten * 1.852 # umrechnung knoten in km/h
15 kmh0 = format(kmh, '.2f') # rundung x stellen hinterm komma
16 print(" km/h:",kmh0)
```

Hier das Python Script zum Herausfiltern der GPS-Daten Zeit und Geschwindigkeit

Vergleich: ‚C‘ mit Python

```
1
2  /* Blinkende LED */
3
4  void setup()
5  {
6    pinMode(13, OUTPUT);
7  }
8
9
10 void loop()
11 {
12   digitalWrite(13, HIGH);
13   delay(1000);
14   digitalWrite(13, LOW);
15   delay(1000);
16 }
```

```
1 from machine import Pin
2 from time import sleep
3
4 led= Pin(2,Pin.OUT)
5
6 while True:
7     sleep(.5)
8     led.value(1)
9     sleep(.5)
10    led.value(0)
```

26.10.22

Arduino & Co

14/16

Wiki:

Python ist eine interpretierte, höhere Programmiersprache.

Blöcke werden nicht durch geschweifte Klammern- wie bei ‚C‘- , sondern durch Einrückungen strukturiert.

Raspberry Pico , RP2040 Zero

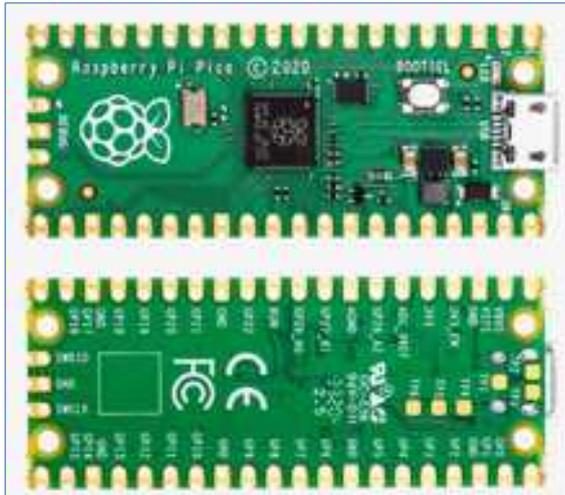
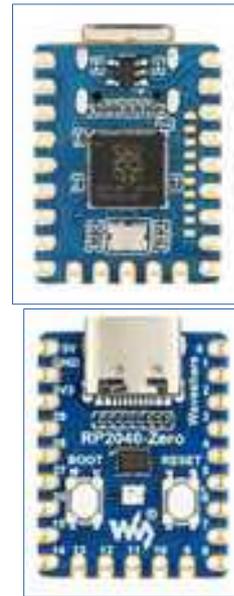


Bild: <https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf>



<https://www.waveshare.com/rp2040-zero.htm>

26.10.22

Arduino &Co

15/16

Raspberry Pico:

Board mit Raspi-CPU, jedoch ohne Betriebssystem, Programmierspache C (z.b. Arduino) oder MicroPython).

Mit Wlan siehe „Raspberry Pi Pico W“

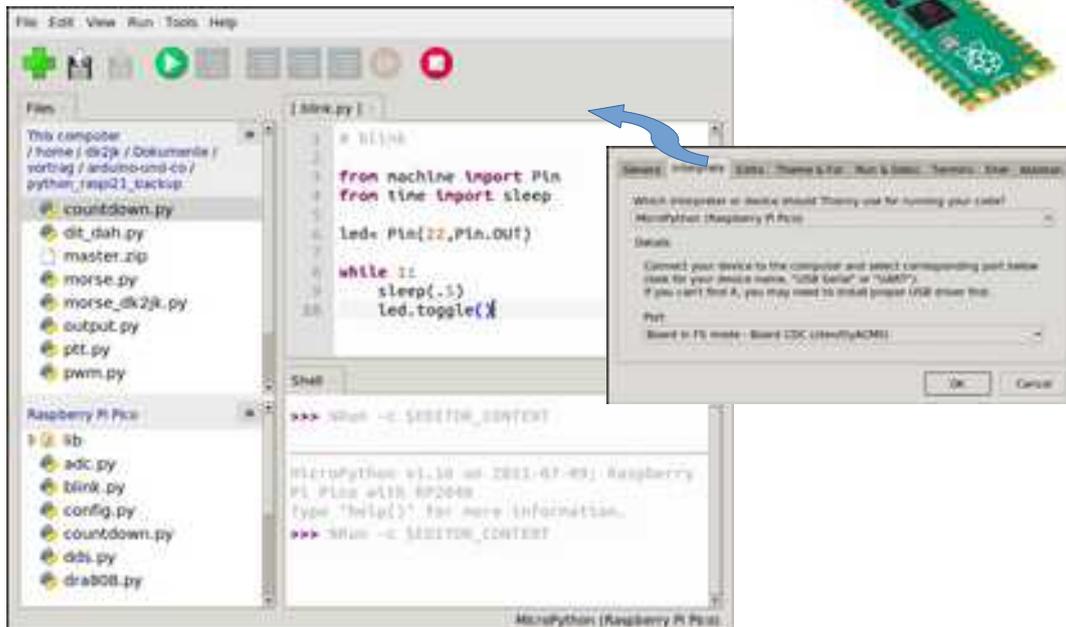
Daten:

- Preiswert (5.95 Euro [Okt 22])
- Viele Pins (26 GPIO)
- 12 Bit ADC
- 2MByte Flash
- 264kB of SRAM,
- 16 × controllable PWM channels
- Micropython

Beschriftung der Pins leider von unten

Kleinere Version : RP2040 Zero

Raspberry Pico



26.10.22

Arduino & Co

16/16

Raspberry Pico:
Board mit Raspi-CPU, jedoch ohne
Betriebssystem, Programmierspache C (z.B.
Arduino) oder MicroPython)

Hier wieder:
Erscheinungsbild in Thonny