

# Vorbereitung Pi

## Hinweis

Für Python Compilieren scheint es besser zu sein eher **32Bit als OS** zu verwenden.  
Die 64Bit Variante hat sich mehrfach und reproduzierbar einfach aufgehängt.

## Pakete

- Allgemeines Update & Tools
  - `sudo apt update && sudo apt upgrade -y && sudo apt install -y git git-lfs silversearcher-ag wavemon hexedit sudoku tcpdump iptraf mc htop dcfldd nano usbutils openvpn ranger tldr ncd uutils multitail fd-find lsof x11vnc terminator minicom cutecom joystick jstest-gtk i2c-tools speedtest-cli iotop fio ir-keytable curl inxi stress-ng && mkdir -p ~/.local/share && tldr -u`
- Devtools installieren
  - `sudo apt install -y thonny cutecom sqlitebrowser build-essential pkg-config libusb-1.0-0-dev cmake make gcc python3-dev libhidapi-dev python3-virtualenv python3-tk lm-sensors libopenblas-dev linux-perf`
- Unnützen Kram deinstallieren
  - `sudo apt autoremove -y modem* cups* pulse* avahi* triggerhappy*`
- Bluetooth
  - `sudo apt-get install blueman pi-bluetooth minicom bluez bluez-tools`
- Python Tools nachinstallieren
  - `sudo apt update && sudo apt install -y build-essential zlib1g-dev libexpat1-dev libxml2-dev libxslt1-dev libpq-dev libjpeg-dev libpng-dev libfreetype6-dev pkg-config checkinstall python3-dev libssl-dev libbz2-dev libffi-dev libncurses5-dev libncursesw5-dev libreadline-dev libsqlite3-dev liblzma-dev libgdbm-dev libdb5.3-dev uuid-dev tk-dev libzstd-dev`
- One 4 All
  - `sudo apt update && sudo apt upgrade -y && sudo apt install -y git git-lfs silversearcher-ag wavemon hexedit sudoku tcpdump iptraf mc htop dcfldd nano usbutils ranger tldr ncd uutils multitail fd-find lsof minicom cutecom joystick i2c-tools speedtest-cli iotop fio ir-keytable curl inxi stress-ng build-essential pkg-config libusb-1.0-0-dev cmake make gcc python3-dev libhidapi-dev python3-virtualenv python3-tk lm-sensors libopenblas-dev linux-perf pi-bluetooth bluez bluez-tools zlib1g-dev libexpat1-dev libxml2-dev libxslt1-dev libzstd-dev libpq-dev libjpeg-dev libpng-dev libfreetype6-dev checkinstall libssl-dev libbz2-dev libffi-dev libncurses5-dev libncursesw5-dev libreadline-dev libsqlite3-dev liblzma-dev libgdbm-dev libdb5.3-dev uuid-dev tk-dev && sudo apt autoremove -y modem* cups* pulse* avahi* triggerhappy* && mkdir -p ~/.local/share && tldr -u`

## config.txt

- `sudo nano /boot/firmware/config.txt`
- Pi Zero MCC Activity LED
  - `dtparam=act_led_trigger=mmc0`
  - `dtparam=act_led_activelow=on`
- Disable Audio
  - `#dtparam=audio=on`
- Overclocking
  - `arm_freq=1300`
  - `over_voltage=4`

## cmdline.txt

- `sudo nano /boot/firmware/cmdline.txt`
- `isolcpus=2,3` hinzufügen
- (!) Das darf nur eine Textzeile sein in der Konfig!

## SWAP erhöhen

- stop the swap
  - `sudo dphys-swapfile swapoff`
- Modify the size of the swap. As root , edit the file `/etc/dphys-swapfile` and modify the variable `CONF_SWAPSIZE` : `CONF_SWAPSIZE=1024`
  - `sudo nano /etc/dphys-swapfile`
- Start the swap
  - `sudo dphys-swapfile swapon`
- Restart
  - `sudo reboot`

## RAM Disk anlegen

- `sudo mkdir /mnt/ramdisk`
- Temporäres Mounten zur Überprüfung
  - `sudo mount -t tmpfs -o size=50M tmpfs /mnt/ramdisk`
  - `df -h`
- Permanente Ramdisk
  - `sudo nano /etc/fstab`
  - Eintrag hinzufügen
    - `tmpfs /mnt/ramdisk tmpfs defaults,size=50M 0 0`

## unnötige Timer stoppen

- Liste anzeigen `systemctl list-timers`

- Deaktiviert tägliche APT-Paketlisten-Updates; reduziert Netz- und CPU-Last, da du manuell updaten kannst.  
`sudo systemctl stop apt-daily.timer && sudo systemctl disable apt-daily.timer`
- Deaktiviert potenzielle automatische Upgrades; verhindert unerwartete Systemänderungen und Ressourcenverbrauch.  
`sudo systemctl stop apt-daily-upgrade.timer && sudo systemctl disable apt-daily-upgrade.timer`
- Deaktiviert tägliche Manpage-Indexierung; unnötig, wenn du selten Hilfeseiten suchst, spart CPU.  
`sudo systemctl stop man-db.timer && sudo systemctl disable man-db.timer`
- Deaktiviert tägliche Backups der Paketdatenbank; bedingt nützlich, aber deaktivierbar bei seltenen Paketänderungen, um Last zu minimieren.  
`sudo systemctl stop dpkg-db-backup.timer && sudo systemctl disable dpkg-db-backup.timer`

## unnötig

- `sudo apt autoremove -y modem* cups* pulse* avahi* triggerhappy*`

## Passwortlos anmelden

- Passwort vom User löschen  
`sudo passwd -d pi`
- `sudo nano /etc/ssh/sshd_config`
  - `PermitEmptyPasswords yes` einfügen
- `sudo reboot`

## WLAN Probleme

Es kommt beim Raspberry Pi Zero 2W öfter vor, dass das WLAN einfach aus geht. Scheinbar ein Problem mit dem Powermanagement ...

- **Powermanagement abschalten**  
`sudo iw dev wlan0 set power_save off`
- Permanagement permanent aus
  - `sudo nmcli con mod preconfigured wifi.powersave disable` oder ...
  - `sudo crontab -e`  
`@reboot /usr/sbin/iw wlan0 set power_save off > /home/<user>/power_save_log.txt 2>&1`
- Check  
`iwconfig` → Power Management:off
- Bring UP  
`sudo nmcli device connect wlan0`  
Falls das nicht funktioniert, versuchen Sie zunächst `sudo ip link set wlan0 up`

# Bluetooth

Bluetooth wird für zwei Dinge verwendet:

- Serielle Verbindung für Logging und einfache Settingsanpassungen im Betrieb
- Gamepad Anbindung für manuelle Steuerung
- Addon : <https://bluedot.readthedocs.io/en/latest/index.html>
- Status prüfen:  
`sudo systemctl status bluetooth`

## aktivieren

- `sudo systemctl enable hciuart.service`  
`sudo systemctl enable bluetooth.service`
  - <https://docs-os.mainsail.xyz/faq/enable-bluetooth-on-rpi>

## SPP Profil

Ziel ist es auf dem Raspberry Pi eine serielle Schnittstelle zu haben auf die man sich von einem anderen Rechner aus verbinden kann

- SPP Profil (BT Uart) → <https://scribles.net/setting-up-bluetooth-serial-port-profile-on-raspberry-pi/>
- Open Bluetooth service configuration file  
`sudo nano /etc/systemd/system/dbus-org.bluez.service`
- Look for a line starts with "ExecStart" and add compatibility flag '-C' at the end of the line  
`ExecStart=/usr/lib/bluetooth/bluetoothd -C`
- Add a line below immediately after "ExecStart" line, then save and close the file  
`ExecStartPost=/usr/bin/sdptool add SP`

```
[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/libexec/bluetooth/bluetoothd -C
ExecStartPost=/usr/bin/sdptool add SP
NotifyAccess=main
#WatchdogSec=10
```

- `sudo systemctl daemon-reload`
- `sudo systemctl restart bluetooth.service`
- Status prüfen:  
`sudo systemctl status bluetooth`  
Liefert jetzt eine extra Zeile mit  
Jun 27 04:05:43 Make-seKwaI sdptool[16038]: Serial Port service registered
- Grundsätzlich muss man pairen.
- Dann muss rfcomm auch laufen, weil der erstellt den seriellen Port!
- Und mit `minicom -b 9600 -o -D /dev/rfcomm0` kann man dann interagieren 😊

# Pairing

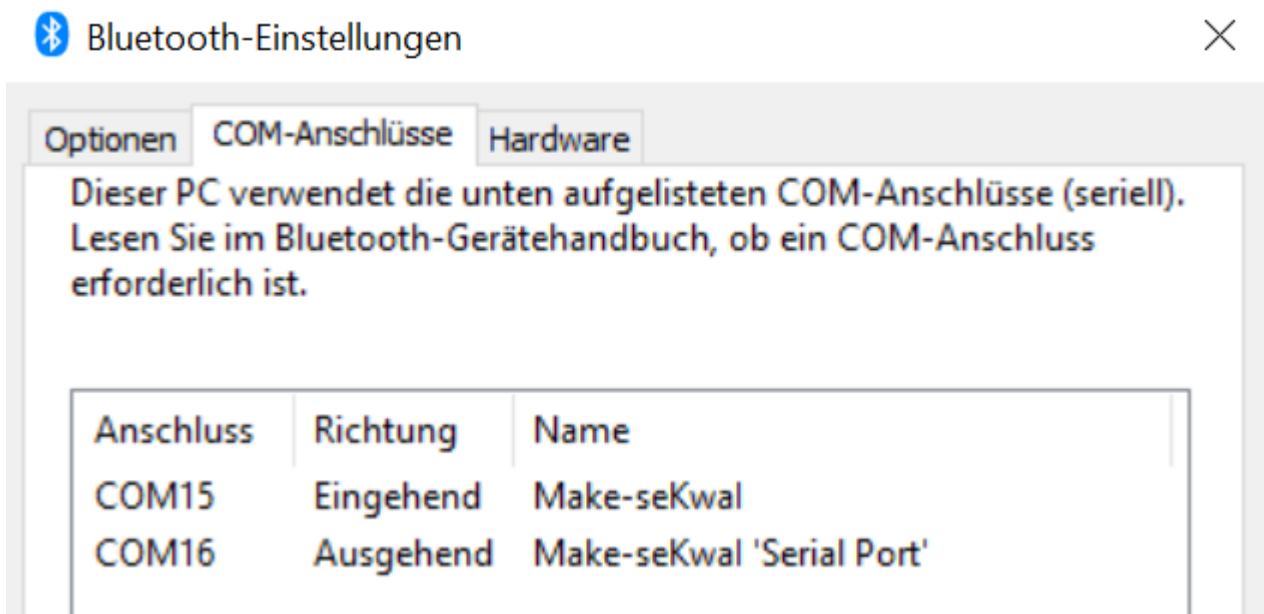
- bluetoothctl
- discoverable on
- On the phone, scan for Raspberry Pi and pair. You should be able to see something like below.  
[CHG] Device XX:XX:XX:XX:XX:XX Paired: yes
- Press Ctrl+D to quit.
- Es muss ggf. auf dem Rechner und dem Pi das Pairing zugestimmt werden

```

pi@Make-seKwaI:~ $ bluetoothctl
Agent registered
[CHG] Controller B8:27:EB:6F:34:92 Pairable: yes
[bluetooth]# discoverable on
Changing discoverable on succeeded
[CHG] Controller B8:27:EB:6F:34:92 Discoverable: yes
[NEW] Device B4:B5:B6:92:61:9A COMP-P-AMD
Request confirmation
[agent] Confirm passkey 675591 (yes/no): yes
[CHG] Device B4:B5:B6:92:61:9A Bonded: yes
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A Modalias: bluetooth:v0006p0001d0A00
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 00001000-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000110b-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000111e-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 0000111f-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIIDs: c7f94713-891e-496a-a0e7-983a0946126e
[CHG] Device B4:B5:B6:92:61:9A ServicesResolved: yes
[CHG] Device B4:B5:B6:92:61:9A Paired: yes
Authorize service
[agent] Authorize service 0000110e-0000-1000-8000-00805f9b34fb (yes/no): yes

```

- Unter Windows sollten jetzt auch 2 serielle Ports angelegt worden sein. Die findet man unter "Weitere Bluetooth Optionen"



- Eingehender COM-Port : Wird typischerweise verwendet, wenn das Bluetooth-Gerät (z. B. ein Sensor oder ein anderes Peripheriegerät) aktiv eine Verbindung herstellt und Daten an den Computer sendet.
- **Ausgehender COM-Port** : Wird genutzt, wenn der Computer aktiv eine serielle

Verbindung zu einem Bluetooth-Gerät herstellt, z. B. für Konfigurationszwecke oder Steuerung.

## rfcomm0

Damit wir eine Verbindung vom Windows zum Pi aufbauen können muss der Pi auf eine eingehende SPP Verbindung horchen. Das geht manuell mittels

```
sudo rfcomm watch hci0
```

Das Device /dev/rfcomm0 wird dann erstellt wenn eine Verbindung von Windows aufgebaut wird. Damit das aber autoamtisch geht muss man einen extra Dienst einrichten:

- `sudo nano /etc/bluetooth/rfcomm.conf`

```
rfcomm0 {
    bind yes;
    channel 1;
    comment "Serial Port for Windows Connection";
}
```

- `sudo nano /etc/systemd/system/rfcomm.service`

```
[Unit]
Description=RFCOMM Service
After=bluetooth.service
Requires=bluetooth.service

[Service]
ExecStart=/usr/bin/rfcomm watch hci0 1
Restart=always

[Install]
WantedBy=multi-user.target
```

- `sudo systemctl enable rfcomm.service`
- `sudo systemctl start rfcomm.service`
- Status prüfen: `sudo systemctl status rfcomm.service`

## Test

### Hinweis:

Scheinbar gibt es Probleme mit .Net Anwendungen. Die Verbindung wird aufgebaut, aber sofort wieder geschlossen. Mit MobaXTerm, Putty oder auch Python passiert das nicht.

\* `minicom -b 9600 -o -D /dev/rfcomm0` auf dem Pi starten

- Ende mittels STRG + A + Z und dann Shift + X
- Python Test Script

[testser.py](#)

```
import serial
```

```
import time
from datetime import datetime

def send_serial_messages(port):
    try:
        # Serielle Verbindung öffnen
        ser = serial.Serial(
            port=port,
            baudrate=9600, # Standard-Baudrate, anpassen falls
nötig
            timeout=1
        )

        # Kurze Pause, um die Verbindung zu stabilisieren
        time.sleep(2)

        print(f"Connected to {port}. Sending time and 'Hallo'
every 2 seconds...")

        while True:
            # Aktuelle Zeit holen
            current_time = datetime.now().strftime("%H:%M:%S")

            # Zeit und "Hallo" als Nachricht zusammenstellen
            message = f"{current_time} Hallo\n"

            # Nachricht senden
            ser.write(message.encode('utf-8'))
            print(f"Sent: {message.strip()}")

            # 2 Sekunden warten
            time.sleep(2)

    except serial.SerialException as e:
        print(f"Error: Could not open port {port}: {e}")
    except KeyboardInterrupt:
        print("\nStopped by user.")
        ser.close()
        print("Serial connection closed.")
    except Exception as e:
        print(f"An error occurred: {e}")
        if 'ser' in locals():
            ser.close()

if __name__ == "__main__":
    # Port definieren
    port = "COM16"

    # Funktion aufrufen
    send_serial_messages(port)
```

- Ein weiterer Test geht mit [https://play.google.com/store/apps/details?id=de.kai\\_morich.serial\\_bluetooth\\_terminal](https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal)

## Links

- <https://willy-tech.de/raspberry-pi-bluetooth-verbinden/>

## SPI

### aktivieren

### BNO085

## Python

### Pakete installieren



- `sudo apt update && sudo apt install -y build-essential zlib1g-dev libexpat1-dev libxml2-dev libxslt1-dev libpq-dev libjpeg-dev libpng-dev libfreetype6-dev pkg-config checkinstall python3-dev libssl-dev libbz2-dev libffi-dev libncurses5-dev libncursesw5-dev libreadline-dev libsqlite3-dev liblzma-dev libgdbm-dev libdb5.3-dev uuid-dev tk-dev`

### pre Compile

- configure: WARNING: no system libmpdecimal found; falling back to bundled libmpdecimal (deprecated and scheduled for removal in Python 3.15)
- `sudo apt install -y libmpdec-dev`
  - Geht nicht unter Bookworm
- `wget https://www.bytereef.org/software/mpdecimal/releases/mpdecimal-4.0.1.tar.gz`
- `tar -xzf mpdecimal-4.0.1.tar.gz && cd mpdecimal-4.0.1`
- Build and install  
`./configure && make -j3 && sudo make install && sudo ldconfig`

### Compilieren

- Basierend auf:
  - <https://raspberrytips.com/install-latest-python-raspberry-pi/>
  - [https://wiki.ubuntuusers.de/Python/manuelle\\_Installation/](https://wiki.ubuntuusers.de/Python/manuelle_Installation/)
- Download von hier : <https://www.python.org/downloads/source/>
  - `wget https://www.python.org/ftp/python/3.13.5/Python-3.13.5.tgz`
  - `wget https://www.python.org/ftp/python/3.14.0/Python-3.14.0b4.tgz`

- Wir nutzen hier 3.14 Beta wegen Fixes im GIL
- [https://dev.to/epam\\_india\\_python/python-313-the-gateway-to-high-performance-multithreading-without-gil-1dm7](https://dev.to/epam_india_python/python-313-the-gateway-to-high-performance-multithreading-without-gil-1dm7)
- tar zxvf Python-3.13.5.tgz && tar zxvf Python-3.14.0b4.tgz
- cd ~/build/Python-3.13.5  
cd ~/build/Python-3.14.0b4
- Configuration mit und Ohne GIL
  - ./configure --enable-optimizations --disable-gil
  - ./configure --enable-optimizations
  - <https://docs.python.org/3/howto/free-threading-python.html>
- make -j2
- Damit das klappt sollte man auf dem Pi Zero 2 den Swap erhöhen auf 2GB 
-  Auf dem Pi Zero 2W wirklich **max. 2 parallel** - sonst hakt es ziemlich ...
- sudo make altinstall

3.13 GIL real **52m1.280s** user 134m52.093s sys 4m6.572s

3.13 NO GIL real **71m22.295s** user 153m6.204s sys 4m10.737s

3.14 NO GIL real **77m10.570s** user 174m21.493s sys 5m35.768s

## venv

- cd ~ && makedirs sekwai && cd sekwai
- python -m venv --copies \_envR
- source \_envR/bin/activate
- pip install spidev build psutil

## Rechte für Prozess Prio setzen

Damit der Pi User die Rechte hat die Prozess Prio neu zu definieren braucht es folgendes:

- sudo apt-get install libcapp2-bin
- Python venv anlegen mit --copies sonst hat man nur Links auf Python !
- sudo setcap cap\_sys\_nice+ep \_envR/bin/python3  
Es gibt da noch 2 weitere Varianten ... Die ggf. auch setzen!
- sudo setcap cap\_sys\_nice+ep \_envR/bin/python && sudo setcap cap\_sys\_nice+ep \_envR/bin/python3 && sudo setcap cap\_sys\_nice+ep \_envR/bin/python3.13

## GIL

run with PYTHON\_GIL=0 or -Xgil=0

- PYTHON\_GIL=1 python3.13
- python3.13 -Xgil=1

```
(_envR) pi@Make-sekwaI:~/test $ python -Xgil=0
Python 3.13.5 experimental free-threading build (main, Jul 9 2025,
07:28:07) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit
```

## Links

- rpi-lgpio
- <https://github.com/waveform80/rpi-lgpio?tab=readme-ov-file>
- <https://rpi-lgpio.readthedocs.io/en/latest/>
- RPi.GPIO
  - neue Lib → <https://github.com/phylax2020/RPi.GPIO>
- spidev
- <https://github.com/doceme/py-spidev>
- I2C - smbus2
- <https://github.com/kplindegaard/smbus2>
- <https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial/all>
- servo → <https://projects.raspberrypi.org/en/projects/grandpa-scarer/3>

From:  
<https://www.drklipper.de/> - **Dr. Klipper Wiki**

Permanent link:  
<https://www.drklipper.de/doku.php?id=projekte:sekwai:prepare&rev=1752494626>

Last update: **2025/07/14 14:03**

