Vorbereitung Pi

Pakete

- `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 ncdu can-utils multitail fd-find lsof x11vnc terminator minicom cutecom joystick jstest-gtk i2c-tools speedtest-cli iotop fio ir-keytable curl inxi && mkdir -p ~/.local/share && tldr -u`
- Devtools nbachinstallieren
  - `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 mariadb-server mariadb-client libopenblas-dev`
- `sudo apt autoremove -y modem* cups* pulse* avahi* triggerhappy*`
- `sudo apt-get install blueman pi-bluetooth minicom bluez bluez-tools`

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`

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`

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

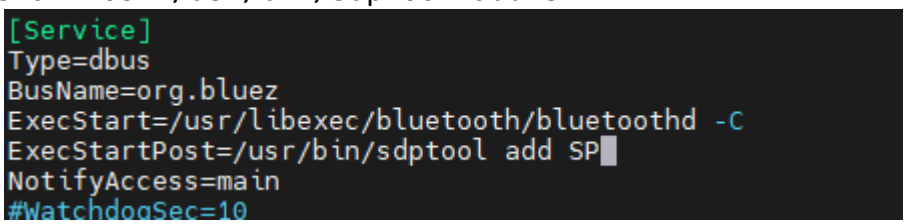
- Status prüfen:
  `sudo systemctl status bluetooth`

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`
  - 

- `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 🙂

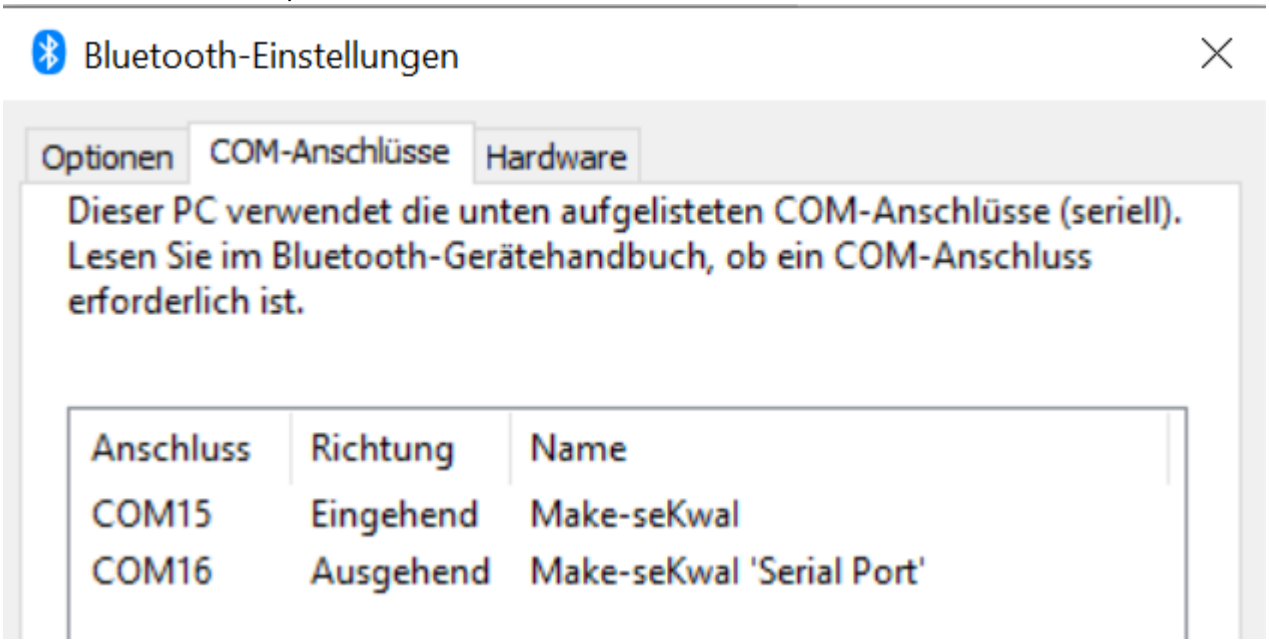
Pairing

- `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 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A Modalias: bluetooth:v0006p0001d0A00
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 00001000-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 0000110b-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 0000111e-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 0000111f-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device B4:B5:B6:92:61:9A UUIDs: 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

```python
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

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`

Compilieren

- Basierend auf:
  - https://raspberrytips.com/install-latest-python-raspberry-pi/
  - https://wiki.ubuntuusers.de/Python/manuelle_Installation/
- Download von hier : https://www.python.org/downloads/source/
- `wget https://www.python.org/ftp/python/3.14.0/Python-3.14.0b3.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
- `tar zxvf Python-3.14.0b3.tgz`
- `cd Python…`
- `./configure –enable-optimizations`
- `make -j2`
  - Damit das klappt sollte man auf dem Pi Zero 2 den Swap erhöhen auf 2GB 🙂
- `sudo make altinstall`

venv

- `cd ~ && mkdor sekwai && cd sekwai`
- `python -m venv _envR`
- `source _envR/bin/activate`
- `pip install rpi-lgpio spidev`

Links

- rpi-lgpio

- https://github.com/waveform80/rpi-lgpio?tab=readme-ov-file
- https://rpi-lgpio.readthedocs.io/en/latest/
- 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

Links

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