Cross-Platform Kompatibilitäts-Guide

Portierung auf Linux, Jetson Nano & Raspberry Pi

Inhaltsverzeichnis

Übersicht

Der Voice Assistant ist primär für Windows entwickelt, kann aber mit minimalen Anpassungen auf Linux-Systemen laufen.

Plattform-Kompatibilität

Plattform	CPU	RAM	Porcupine	Vosk	Edge TTS	Status
Windows 10/11	x86_64	4GB+	✅	✅	✅	✅ Vollständig
Raspberry Pi 4	ARM64	4GB+	✅	✅	✅	✅ Getestet
Raspberry Pi 5	ARM64	8GB+	✅	✅	✅	✅ Empfohlen
Jetson Nano	ARM64	4GB	✅	✅	✅	⚠ Experimentell
Ubuntu/Debian	x86_64	4GB+	✅	✅	✅	✅ Vollständig
macOS	x86_64/ARM	8GB+	✅	✅	✅	⚠ Nicht getestet

Raspberry Pi 4/5

Hardware-Anforderungen

Minimum:

- Raspberry Pi 4 Model B (4GB RAM)
- USB-Mikrofon oder USB-Soundkarte mit Mikrofon
- Lautsprecher (3.5mm oder USB)
- MicroSD-Karte (32GB+, Class 10)

Empfohlen:

- Raspberry Pi 5 (8GB RAM)
- USB-Mikrofon mit Rauschunterdrückung (z.B. Blue Yeti Nano)
- USB-Lautsprecher oder DAC (bessere Qualität als 3.5mm)
- MicroSD-Karte (64GB+, A2)

Software-Setup

1. Raspberry Pi OS installieren

```
# Raspberry Pi Imager verwenden (empfohlen)
# https://www.raspberrypi.com/software/

# Oder manuell:
# Download Raspberry Pi OS (64-bit, Lite oder Desktop)
# https://www.raspberrypi.com/software/operating-systems/

# Flash auf MicroSD mit dd oder Etcher
```

Empfehlung: Raspberry Pi OS 64-bit (Bookworm) - Lite für Headless, Desktop für GUI

2. System aktualisieren

```
sudo apt update && sudo apt upgrade -y
sudo reboot
```

3. Python & Dependencies installieren

```
# Python 3.11+ (sollte bereits installiert sein)
python3 --version

# pip
sudo apt install python3-pip python3-venv -y

# System-Dependencies
sudo apt install -y \
    portaudio19-dev \
    python3-pyaudio \
    libasound2-dev \
    libportaudio2 \
    libportaudiocpp0 \
    ffmpeg \
    git
```

4. Voice Assistant Dependencies

```
# Erstelle Virtual Environment
python3 -m venv ~/voice_assi_env
source ~/voice_assi_env/bin/activate

# Installiere Python-Pakete
pip install --upgrade pip

# Core Dependencies
pip install pvporcupine
pip install vosk
pip install edge-tts
pip install sounddevice
pip install numpy
pip install pygame
pip install webrtcvad
pip install python-dotenv

# Optional: LLM Integration
pip install openai
```

5. Audio-Setup (wichtig!)

```
# Liste Audio-Geräte
python3 -c "import sounddevice as sd; print(sd.query_devices())"

# Setze Standard-Geräte (falls nötig)
# Bearbeite ~/.asoundrc
nano ~/.asoundrc
```

~/.asoundrc Beispiel:

```
pcm.!default {
    type asym
    playback.pcm "plughw:1,0"  # Lautsprecher
    capture.pcm "plughw:2,0"   # Mikrofon
}
```

Test Mikrofon:

```
# Aufnahme
arecord -d 5 -f cd test.wav

# Wiedergabe
aplay test.wav
```

6. Voice Assistant klonen & konfigurieren

```
cd ~
git clone https://github.com/KoMMb0t/Computer-Voice-Assi.git
cd Computer-Voice-Assi

# Konfiguration anpassen
nano config.ini

# Picovoice Access Key eintragen
# Pfade anpassen (falls nötig)
```

7. Modelle herunterladen

```
# Porcupine Wake-Word Modell
# Trainiere auf https://console.picovoice.ai
# Download und speichere in models/computer.ppn

# Vosk Modell (Deutsch, klein für Raspberry Pi)
cd models/
wget https://alphacephei.com/vosk/models/vosk-model-small-de-0.15.zip
unzip vosk-model-small-de-0.15.zip
rm vosk-model-small-de-0.15.zip
```

8. Starten

```
source ~/voice_assi_env/bin/activate
cd ~/Computer-Voice-Assi
python3 15_voice_assistant_configurable.py
```

Performance-Optimierung (Raspberry Pi)

CPU-Frequenz erhöhen

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

# Füge hinzu:
over_voltage=6
arm_freq=2000  # Pi 4: max 2000, Pi 5: max 2400
```

GPU-Memory reduzieren (Headless)

```
# In /boot/config.txt
gpu_mem=16  # Minimum für Headless
```

Swap erhöhen (falls <4GB RAM)

```
sudo dphys-swapfile swapoff
sudo nano /etc/dphys-swapfile

# Ändere:
CONF_SWAPSIZE=2048

sudo dphys-swapfile setup
sudo dphys-swapfile swapon
```

Autostart einrichten

```
# Systemd Service erstellen
sudo nano /etc/systemd/system/voice-assistant.service
```

voice-assistant.service:

```
[Unit]
Description=Computer Voice Assistant
After=network.target sound.target

[Service]
Type=simple
User=pi
WorkingDirectory=/home/pi/Computer-Voice-Assi
Environment="PATH=/home/pi/voice_assi_env/bin"
ExecStart=/home/pi/voice_assi_env/bin/python3 15_voice_assistant_configurable.py
Restart=always
RestartSec=10

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

Aktivieren:

```
sudo systemctl daemon-reload
sudo systemctl enable voice-assistant.service
sudo systemctl start voice-assistant.service

# Status prüfen
sudo systemctl status voice-assistant.service

# Logs
sudo journalctl -u voice-assistant.service -f
```

NVIDIA Jetson Nano

Hardware-Anforderungen

- Jetson Nano Developer Kit (4GB)
- USB-Mikrofon
- Lautsprecher
- MicroSD-Karte (64GB+)

- 5V 4A Power Supply (wichtig!)

Software-Setup

1. JetPack installieren

```
# Download JetPack SD Card Image
# https://developer.nvidia.com/embedded/jetpack

# Flash mit Etcher oder balenaEtcher
# https://www.balena.io/etcher/
```

2. System aktualisieren

```
sudo apt update && sudo apt upgrade -y
```

3. Dependencies

```
# System-Pakete
sudo apt install -y \
    python3-pip \
    python3-venv \
    portaudio19-dev \
    libasound2-dev \
    ffmpeg \
    git

# Virtual Environment
python3 -m venv ~/voice_assi_env
source ~/voice_assi_env/bin/activate

# Python-Pakete (wie Raspberry Pi)
pip install pvporcupine vosk edge-tts sounddevice numpy pygame webrtcvad python-
dotenv
```

4. Porcupine für ARM64

Wichtig: Porcupine unterstützt Jetson Nano offiziell. Verwende ARM64-Version.

```
# Prüfe Architektur
uname -m
# Sollte "aarch64" ausgeben

# pvporcupine sollte automatisch richtige Version installieren
pip show pvporcupine
```

5. Audio-Setup

```
# Wie Raspberry Pi (siehe oben)
python3 -c "import sounddevice as sd; print(sd.query_devices())"
```

6. GPU-Acceleration (Optional)

Jetson Nano hat CUDA-fähige GPU, kann für zukünftige Features (z.B. Silero VAD) genutzt werden.

```
# PyTorch mit CUDA (für Advanced VAD)
# https://forums.developer.nvidia.com/t/pytorch-for-jetson/72048

# Vorerst nicht nötig, da Vosk CPU-basiert ist
```

Bekannte Probleme

Problem: Porcupine langsam auf Jetson Nano

Lösung: Verwende kleineres Modell oder reduziere Sensitivity

Problem: Audio-Dropouts

Lösung: Verwende USB-Soundkarte statt onboard Audio

Linux Desktop (Ubuntu/Debian)

Ubuntu 22.04/24.04

```
# System aktualisieren
sudo apt update && sudo apt upgrade -y

# Dependencies
sudo apt install -y \
    python3 \
    python3-pip \
    python3-venv \
    portaudio19-dev \
    python3-pyaudio \
    libasound2-dev \
    ffmpeg \
    git

# Voice Assistant Setup (wie Windows)
git clone https://github.com/KoMMb0t/Computer-Voice-Assi.git
cd Computer-Voice-Assi

python3 -m venv venv
source venv/bin/activate

pip install -r requirements.txt

# Konfigurieren
nano config.ini

# Starten
python3 15_voice_assistant_configurable.py
```

Debian 12

Identisch zu Ubuntu, funktioniert out-of-the-box.

Abhängigkeiten & Installation

Plattform-spezifische Unterschiede

Paket	Windows	Linux	Raspberry Pi	Jetson Nano
pvporcupine	pip	pip	pip	pip
vosk	pip	pip	pip	pip
edge-tts	pip	pip	pip	pip
sounddevice	pip	pip + portaudio19-dev	pip + portaudio19-dev	pip + portaudio19-dev
pygame	pip	pip	pip	pip
webrtcvad	pip	pip	pip	pip

requirements.txt (Universal)

```
pvporcupine>=3.0.0
vosk>=0.3.45
edge-tts>=6.1.0
sounddevice>=0.4.6
numpy>=1.24.0
pygame>=2.5.0
webrtcvad>=2.0.10
python-dotenv>=1.0.0
openai>=1.0.0
```

Audio-Setup

Mikrofon-Test (alle Plattformen)

```
import sounddevice as sd
import numpy as np

# Liste Geräte
print(sd.query_devices())

# Teste Mikrofon (Device-ID anpassen)
device_id = 0  # Ändere falls nötig
duration = 5  # Sekunden

print(f"Aufnahme von Device {device_id} für {duration}s...")
audio = sd.rec(int(duration * 16000),
               samplerate=16000,
               channels=1,
               dtype='int16',
               device=device_id)
sd.wait()

print(f"Max Amplitude: {np.max(np.abs(audio))}")
print("✅ Mikrofon funktioniert!" if np.max(np.abs(audio)) > 100 else "❌ Mikrofon
zu leise!")
```

Lautsprecher-Test

```
# Linux
speaker-test -t wav -c 2

# Python
python3 -c "import pygame; pygame.mixer.init();
pygame.mixer.music.load('test.mp3'); pygame.mixer.music.play()"
```

Performance-Optimierung

Raspberry Pi / Jetson Nano

Problem: Hohe CPU-Last, Latenz

Lösungen:

1. **Kleineres Vosk-Modell:**

```
# Statt vosk-model-de-0.22 (1.8GB)
# Verwende vosk-model-small-de-0.15 (45MB)
```

2. **Porcupine Sensitivity reduzieren:**

```
# config.ini
[WakeWord]
sensitivity = 0.3  # Statt 0.5
```

3. **VAD Aggressiveness erhöhen:**

```
# config.ini
[Audio]
vad_aggressiveness = 3  # Maximum
```

4. **Disable Debug Mode:**

```
# config.ini
[Advanced]
debug_mode = false
```

5. **CPU-Gouverneur auf Performance:** “`bash

Raspberry Pi

echo performance | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor

Jetson Nano sudo nvpmodel -m 0 # Max Performance Mode sudo jetson_clocks # Max Clock Speeds

```
---

## Troubleshooting {#troubleshooting}

### Problem: "PortAudio not found"

**Lösung:**
```bash
sudo apt install portaudio19-dev
pip install --force-reinstall sounddevice
```

Problem: “No default input device”

Lösung:

```
# Liste Geräte
python3 -c "import sounddevice as sd; print(sd.query_devices())"

# Setze Default in Code
sd.default.device = (2, 1)  # (Input, Output)
```

Problem: “Porcupine: Invalid model file”

Lösung:

- Stelle sicher, dass `computer.ppn` für richtige Plattform trainiert wurde
- Porcupine Console: Wähle “Raspberry Pi” für ARM64
```

Problem: “Edge TTS: Connection error”

Lösung:

```
# Prüfe Internet-Verbindung
ping google.com

# Teste Edge TTS
edge-tts --text "Test" --write-media test.mp3
```

Problem: Hohe CPU-Last

Lösung:

- Verwende kleineres Vosk-Modell
- Reduziere Porcupine Sensitivity
- Disable Debug Mode
- Verwende Performance CPU-Gouverneur

Problem: Audio-Dropouts

Lösung:

```
# Erhöhe Audio-Buffer
# In Code:
sd.default.blocksize = 2048  # Statt 512
```

Zusammenfassung

Empfohlene Plattformen

Für Entwicklung:

- Windows $^{10}/_{11}$ (beste Kompatibilität)

- Ubuntu 22.04/24.04 (beste Linux-Erfahrung)

Für Produktion:

- Raspberry Pi 5 (8GB) - Beste Balance aus Performance & Preis
- Ubuntu Server (x86_64) - Höchste Performance

Für Embedded:

- Raspberry Pi 4 (4GB) - Minimum
- Jetson Nano - Wenn GPU-Acceleration geplant ist

Nächste Schritte

1. Wähle Ziel-Plattform
2. Folge Setup-Anleitung
3. Teste Audio-Setup
4. Optimiere Performance
5. Richte Autostart ein

Dokumentende
