Voice Assistant Projekt - Übergabe-Dokument

📋 Projekt-Übersicht

Projektname: Voice Assistant - KI Sprachsteuerung
GitHub Repository: https://github.com/KoMMb0t/voice_assi
Aktueller Status: Funktionsfähiger Prototyp mit Wake-Word-Doppelerkennungs-Problem
Nächster Schritt: Eigenes "Computer" Wake-Word trainieren + LLM-Integration

---

🎯 Was bisher erreicht wurde

✅ Erfolgreich implementiert:

1. **Voice Assistant Grundsystem**
   - Wake-Word Detection mit OpenWakeWord (aktuell: "hey jarvis")
   - Speech-to-Text mit Vosk (deutsches Modell)
   - Text-to-Speech mit Edge TTS (Stimme: Katja)
   - Voice Activity Detection (VAD) für automatische Aufnahme-Beendigung
2. **Befehle implementiert:**
   - Programme öffnen (Taschenrechner, Notepad, Explorer, Firefox)
   - Webseiten öffnen (YouTube, ChatGPT, Google, Gmail, GitHub, Wikipedia)
   - Datum & Uhrzeit (mit deutscher Übersetzung)
   - Höflichkeits-Befehle ("Danke", "Abbrechen", "Nichts")
   - Hilfe-Befehl

3. **Projekt-Setup:**
    - Git Repository initialisiert
    - `.gitignore` erstellt (schließt `.venv` aus)
    - `requirements.txt` generiert
    - README.md mit deutscher und englischer Dokumentation
    - Alles auf GitHub hochgeladen
4. **Benutzerfreundlichkeit:**
    - Batch-Datei zum Starten (`Start_Voice_Assistant.bat`)
    - Desktop-Icon mit eigenem Symbol
    - PowerShell-Alias geplant (noch nicht vollständig implementiert)

❌ Aktuelles Problem:

Wake-Word-Doppelerkennung:

- Das Wake-Word “hey jarvis” wird manchmal doppelt erkannt, ohne dass der Nutzer es erneut sagt
- Score ist konstant sehr hoch (0.99-1.00)
- Verschiedene Fixes wurden versucht (Threshold senken, Cooldown verlängern, Audio-Buffer leeren)
- Problem besteht weiterhin

🚀 Nächste Schritte (Option 1)

Hauptziel: Eigenes “Computer” Wake-Word trainieren

Warum?

- “Computer” ist das gewünschte Wake-Word (wie in Star Trek)
- Könnte das Doppelerkennungs-Problem lösen
- Steht auf der Roadmap

Schritte:

1. **Aufnahmen sammeln**
    - 100-200 Aufnahmen von “Computer” in verschiedenen Tonlagen
    - Hintergrundgeräusche aufnehmen
    - Negative Samples (andere Wörter)
2. **Modell trainieren**
    - OpenWakeWord Training-Pipeline nutzen
    - Oder: Porcupine Custom Wake Word (kostenpflichtig, aber einfacher)
    - Oder: Snowboy (veraltet, aber kostenlos)
3. **Modell integrieren**
    - In den bestehenden Code einbauen
    - Testen und optimieren
4. **Auf GitHub hochladen**
    - Neues Modell committen
    - README aktualisieren

🤖 LLM-Integration (Roadmap)

Ziel: Intelligente Konversationen statt nur Befehle

Geplante LLM-Integrationen:

- ChatGPT (OpenAI API)
- Perplexity (für Recherche)
- Monica (Browser-Extension)
- Manus (für komplexe Aufgaben)

Architektur:

1. Wake-Word erkannt → "Ja?"
2. Nutzer spricht Befehl/Frage
3. **NEU:** Wenn kein Befehl erkannt → An LLM senden
4. LLM antwortet → TTS spricht Antwort

Beispiel:

- "Computer, wie wird das Wetter morgen?" → Perplexity API
- "Computer, schreibe eine E-Mail an Max" → ChatGPT
- "Computer, erstelle eine Präsentation über KI" → Manus

🔧 Technische Details

Projektstruktur:

```
C:\Users\ModBot\ki-sprachsteuerung\
├── .venv/                          # Virtual Environment (nicht auf GitHub)
├── .gitignore                      # Git-Ignore-Datei
├── requirements.txt                # Python-Abhängigkeiten
├── README.md                       # Projekt-Dokumentation
├── voice_assistant_edge.py         # Hauptprogramm (aktuelle Version)
├── voice_assistant_tts.py          # TTS-Test-Version
├── listen.py                       # Wake-Word-Test
├── listen_and_transcribe.py        # STT-Test
├── download_models.py              # Modell-Downloader
└── Start_Voice_Assistant.bat       # Batch-Datei zum Starten
```

Wichtige Konfigurationen:

```
WAKE_WORD = "hey jarvis"            # Soll zu "computer" werden
SAMPLE_RATE = 16000                 # Audio-Sample-Rate
CHUNK_SAMPLES = 1280                # Audio-Chunk-Größe
SILENCE_TIMEOUT = 2.0               # Stille-Erkennung (Sekunden)
MAX_RECORD_TIME = 30                # Max. Aufnahmezeit (Sekunden)
TTS_VOICE = "de-DE-KatjaNeural"     # Edge TTS Stimme
```

Abhängigkeiten (requirements.txt):

```
openwakeword
vosk
edge-tts
sounddevice
numpy
pygame
```

📝 Bekannte Probleme & Lösungsansätze

Problem 1: Wake-Word-Doppelerkennung

Versuchte Lösungen:

- Threshold von 0.95 auf 0.7 gesenkt
- Cooldown von 0.5 auf 2.0 Sekunden erhöht
- Audio-Buffer nach Wake-Word leeren
- Cooldown-Flag hinzugefügt

Noch nicht versucht:

- Threshold auf 0.5 oder niedriger senken
- Cooldown auf 4+ Sekunden erhöhen
- Wake-Word-Listening während Befehlsverarbeitung komplett pausieren
- Anderes Wake-Word-Modell testen (z.B. “alexa” - aber nicht gewünscht)

Problem 2: Mikrofon-Empfindlichkeit

Lösung: Mikrofon-Lautstärke in Windows reduziert (bereits erledigt)

🎓 Manus-Features nutzen

Parallele Aufgaben in Manus:

Manus kann mehrere Aufgaben gleichzeitig bearbeiten!

Beispiel-Workflow:

1. **Chat 1:** "Computer" Wake-Word trainieren
    - Aufnahmen machen
    - Modell trainieren
    - Testen
2. **Chat 2:** LLM-Integration vorbereiten
    - API-Keys besorgen
    - Code-Struktur planen
    - Erste Tests
3. **Chat 3:** GitHub-Dokumentation verbessern
    - README erweitern
    - Screenshots hinzufügen
    - Roadmap aktualisieren

So startest du parallele Aufgaben:

- Öffne mehrere Manus-Chats (neue Tabs)
- Gib jedem Chat eine spezifische Aufgabe
- Manus arbeitet an allen gleichzeitig

🔗 Wichtige Links

- **GitHub Repo:** https://github.com/KoMMb0t/voice_assi
- **OpenWakeWord Docs:** https://github.com/dscripka/openWakeWord

- **Vosk Models:** https://alphacephei.com/vosk/models
- **Edge TTS:** https://github.com/rany2/edge-tts

💬 Übergabe-Prompt für neuen Manus-Chat

Kopiere diesen Text in einen neuen Manus-Chat:

```
Hallo! Ich arbeite an einem Voice Assistant Projekt und brauche Hilfe beim nächsten Schritt.

**Aktueller Stand:**
- Funktionsfähiger Voice Assistant mit Wake-Word ("hey jarvis"), STT (Vosk), TTS (Edge TTS)
- Projekt ist auf GitHub: https://github.com/KoMMb0t/voice_assi
- Problem: Wake-Word wird manchmal doppelt erkannt

**Nächstes Ziel:**
1. Eigenes "Computer" Wake-Word trainieren (Option 1)
2. LLM-Integration (ChatGPT, Perplexity, Monica, Manus)
3. Alles auf GitHub dokumentieren

**Technische Details:**
- Windows 11 Mini PC
- Python 3.11
- Virtual Environment: .venv
- Projektordner: C:\Users\ModBot\ki-sprachsteuerung

**Ich habe ein detailliertes Übergabe-Dokument mit allen Informationen.**

Kannst du mir helfen, das "Computer" Wake-Word zu trainieren und die LLM-Integration zu planen?
```

📊 Roadmap (aus README.md)

- [ ] LLM integration for more intelligent conversations (ChatGPT, Perplexity, etc.)
- [ ] Expand to other devices (Jetson Nano, Raspberry Pi, Android)
- [🔄] Train a custom “Computer” wake word model (IN ARBEIT)

- [ ] Integrate with home automation systems
- [ ] Enable secure remote access (VPN/Tailscale)

🎯 Erfolgs-Kriterien für Option 1

Das "Computer" Wake-Word ist erfolgreich, wenn:

1. ✅ Zuverlässige Erkennung (>95% Erfolgsquote)
2. ✅ Keine Doppel-Erkennungen
3. ✅ Keine Fehlerkennungen bei ähnlichen Wörtern
4. ✅ Funktioniert in verschiedenen Umgebungen (leise, laut, mit Hintergrundgeräuschen)
5. ✅ Modell ist auf GitHub verfügbar

🤝 Zusammenarbeit mit anderen KIs

ChatGPT:

- Code-Optimierung
- Algorithmen erklären
- Schnelle Antworten

Perplexity:

- Recherche zu Wake-Word-Training
- Vergleich verschiedener Ansätze
- Aktuelle Best Practices

Monica:

- Browser-basierte Aufgaben
- Schnelle Web-Recherche

Manus:

- Komplexe Projekte (wie dieses!)
- Parallele Aufgaben
- Langfristige Planung

📌 Wichtige Notizen

- Nutzer heißt: ModBot / KoMMb0t (GitHub)
- E-Mail: kommuniverse@gmail.com
- Bevorzugte Sprache: Deutsch
- Erfahrung: Anfänger in Python/Git, aber lernwillig
- Ziel: Geräteübergreifende KI-Sprachsteuerung (Windows, Linux, Android)

Viel Erfolg mit dem Projekt! 🚀