Home Assistant Integration

REST API, WebSocket API & MQTT

Inhaltsverzeichnis

Übersicht

Home Assistant ist eine Open-Source-Plattform für Smart Home Automation. Die Integration ermöglicht Sprachsteuerung von Lichtern, Thermostaten, Kameras und mehr.

Warum Home Assistant?

Vorteile:

- ✅ **Open Source** & selbst-gehostet (Datenschutz)
- ✅ **Plattform-unabhängig** (Philips Hue, IKEA, Xiaomi, etc.)
- ✅ **Lokale Steuerung** (kein Cloud-Zwang)
- ✅ **Kostenlos** (keine Abo-Gebühren)
- ✅ **Erweiterbar** (1000+ Integrationen)

Use-Cases:

- "Computer, mach das Licht im Wohnzimmer an"
- "Computer, stelle die Heizung auf 22 Grad"
- "Computer, zeig mir die Kamera an der Haustür"
- "Computer, starte die Kaffeemaschine"

Home Assistant Setup

Installation

Option 1: Home Assistant Operating System (empfohlen)

```
# Raspberry Pi 4/5
# Download Image: https://www.home-assistant.io/installation/raspberrypi

# Flash mit Raspberry Pi Imager
# Starte Raspberry Pi
# Öffne http://homeassistant.local:8123
```

Option 2: Home Assistant Container (Docker)

```
docker run -d \
  --name homeassistant \
  --privileged \
  --restart=unless-stopped \
  -e TZ=Europe/Berlin \
  -v /PATH_TO_YOUR_CONFIG:/config \
  --network=host \
  ghcr.io/home-assistant/home-assistant:stable
```

Option 3: Home Assistant Core (Python)

```
python3 -m venv homeassistant
source homeassistant/bin/activate
pip install homeassistant
hass
```

Zugriff

- **Web-Interface:** http://homeassistant.local:8123
- **REST API:** http://homeassistant.local:8123/api/
- **WebSocket API:** ws://homeassistant.local:8123/api/websocket

Access Token erstellen

1. Öffne Home Assistant Web-Interface
2. Gehe zu **Profil** (unten links)
3. Scrolle zu **Long-Lived Access Tokens**
4. Klicke **Create Token**
5. Name: "Voice Assistant"
6. Kopiere Token (nur einmal sichtbar!)

Speichere Token:

```
# .env Datei
HOMEASSISTANT_TOKEN=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
HOMEASSISTANT_URL=http://homeassistant.local:8123
```

Methode 1: REST API

Beschreibung

Die REST API ist die einfachste Methode für grundlegende Operationen (Licht an/aus, Temperatur setzen, etc.).

Python-Library

```
pip install HomeAssistant-API
```

Beispiel-Code

```python
import os
from homeassistant_api import Client

# Initialisiere Client
url = os.getenv('HOMEASSISTANT_URL', 'http://homeassistant.local:8123')
token = os.getenv('HOMEASSISTANT_TOKEN')

client = Client(url, token)

# Liste alle Entities
entities = client.get_entities()
print(f"Gefundene Entities: {len(entities)}")

for entity in entities:
    print(f"- {entity.entity_id}: {entity.state.state}")

# Schalte Licht an
light = client.get_entity(entity_id="light.wohnzimmer")
light.turn_on()

# Schalte Licht aus
light.turn_off()

# Setze Helligkeit (0-255)
light.turn_on(brightness=128)

# Setze Farbe (RGB)
light.turn_on(rgb_color=[255, 0, 0])  # Rot

# Thermostat setzen
climate = client.get_entity(entity_id="climate.heizung")
climate.set_temperature(temperature=22)

# Schalter
switch = client.get_entity(entity_id="switch.kaffeemaschine")
switch.turn_on()
```

Manuelle REST Calls (ohne Library)

```python
import requests
import os

url = os.getenv('HOMEASSISTANT_URL')
token = os.getenv('HOMEASSISTANT_TOKEN')

headers = {
    "Authorization": f"Bearer {token}",
    "Content-Type": "application/json"
}

# GET: Liste States
response = requests.get(f"{url}/api/states", headers=headers)
states = response.json()

for state in states:
    print(f"{state['entity_id']}: {state['state']}")

# POST: Schalte Licht an
data = {"entity_id": "light.wohnzimmer"}
response = requests.post(
    f"{url}/api/services/light/turn_on",
    headers=headers,
    json=data
)

print(f"Response: {response.status_code}")
```

Verfügbare Services

Licht:

- `light.turn_on` - Licht einschalten
- `light.turn_off` - Licht ausschalten
- `light.toggle` - Licht umschalten

Klima:

- `climate.set_temperature` - Temperatur setzen

- `climate.set_hvac_mode` - Modus setzen (heat, cool, auto, off)

Schalter:

- `switch.turn_on` - Schalter einschalten
- `switch.turn_off` - Schalter ausschalten

Media Player:

- `media_player.play_media` - Medien abspielen
- `media_player.pause` - Pause
- `media_player.volume_set` - Lautstärke setzen

Vollständige Liste: https://www.home-assistant.io/integrations/#all

---

Methode 2: WebSocket API

Beschreibung

Die WebSocket API ist besser für Echtzeit-Updates und bidirektionale Kommunikation.

Python-Library

```
pip install homeassistant-websocket-api
```

Beispiel-Code

```python
import asyncio
from homeassistant_api import Client

async def main():
    url = "http://homeassistant.local:8123"
    token = "YOUR_TOKEN"

    async with Client(url, token) as client:
        # Subscribe zu State-Changes
        async def state_changed(event):
            print(f"State changed: {event}")

        await client.subscribe_events(state_changed, "state_changed")

        # Warte auf Events
        await asyncio.sleep(60)

asyncio.run(main())
```

Manueller WebSocket-Client

```python
import asyncio
import websockets
import json

async def connect_to_ha():
    url = "ws://homeassistant.local:8123/api/websocket"
    token = "YOUR_TOKEN"

    async with websockets.connect(url) as websocket:
        # 1. Empfange Auth Required
        msg = await websocket.recv()
        print(f"Received: {msg}")

        # 2. Sende Auth
        auth_msg = {
            "type": "auth",
            "access_token": token
        }
        await websocket.send(json.dumps(auth_msg))

        # 3. Empfange Auth OK
        msg = await websocket.recv()
        print(f"Auth: {msg}")

        # 4. Subscribe zu Events
        subscribe_msg = {
            "id": 1,
            "type": "subscribe_events",
            "event_type": "state_changed"
        }
        await websocket.send(json.dumps(subscribe_msg))

        # 5. Empfange Events
        while True:
            msg = await websocket.recv()
            data = json.loads(msg)
            print(f"Event: {data}")

asyncio.run(connect_to_ha())
```

Methode 3: MQTT

Beschreibung

MQTT ist ein Messaging-Protokoll für IoT-Geräte. Ideal für bidirektionale Kommunikation und Sensoren.

Setup MQTT Broker in Home Assistant

1. Gehe zu **Settings** → **Add-ons**
2. Suche “Mosquitto broker”
3. Installiere & Starte
4. Gehe zu **Settings** → **Devices & Services**
5. Klicke **Add Integration** → “MQTT”
6. Konfiguriere (localhost:1883)

Python-Library

```
pip install paho-mqtt
```

Beispiel-Code

```python
import paho.mqtt.client as mqtt
import json

# MQTT Callbacks
def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")

    # Subscribe zu State-Changes
    client.subscribe("homeassistant/+/+/state")

def on_message(client, userdata, msg):
    print(f"Topic: {msg.topic}")
    print(f"Payload: {msg.payload.decode()}")

# Initialisiere Client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

# Verbinde zu Broker
client.connect("homeassistant.local", 1883, 60)

# Schalte Licht an via MQTT
payload = json.dumps({"state": "ON"})
client.publish("homeassistant/light/wohnzimmer/set", payload)

# Loop
client.loop_forever()
```

MQTT Discovery

Home Assistant unterstützt MQTT Discovery für automatische Geräte-Erkennung.

```python
import paho.mqtt.client as mqtt
import json

client = mqtt.Client()
client.connect("homeassistant.local", 1883)

# Registriere neuen Sensor
config = {
    "name": "Voice Assistant Status",
    "state_topic": "voice_assistant/status",
    "icon": "mdi:microphone"
}

client.publish(
    "homeassistant/sensor/voice_assistant/config",
    json.dumps(config),
    retain=True
)

# Sende Status
client.publish("voice_assistant/status", "listening")
```

Vergleich der Methoden

Feature	REST API	WebSocket API	MQTT
Einfachheit	⭐⭐⭐⭐⭐	⭐⭐⭐	⭐⭐⭐
Echtzeit	❌	✅	✅
Bidirektional	❌	✅	✅
Overhead	Niedrig	Mittel	Niedrig
Latenz	50-200ms	10-50ms	10-50ms
Use-Case	Einfache Commands	Echtzeit-Updates	IoT-Geräte

Empfehlung

Für Voice Assistant:

- **REST API** (Primär) - Einfach, ausreichend für Commands
- **WebSocket API** (Optional) - Für Echtzeit-Feedback
- **MQTT** (Erweitert) - Für eigene Sensoren/Aktoren

Integration in Voice Assistant

HomeAssistantManager Klasse

```python
#!/usr/bin/env python3
"""
Home Assistant Manager für Voice Assistant
"""

import os
import requests
from typing import Optional, Dict, List

class HomeAssistantManager:
    """Verwaltet Home Assistant Integration."""

    def __init__(self, url: Optional[str] = None, token: Optional[str] = None):
        self.url = url or os.getenv('HOMEASSISTANT_URL',
'http://homeassistant.local:8123')
        self.token = token or os.getenv('HOMEASSISTANT_TOKEN')

        if not self.token:
            raise ValueError("Home Assistant Token nicht gefunden!")

        self.headers = {
            "Authorization": f"Bearer {self.token}",
            "Content-Type": "application/json"
        }

        # Teste Verbindung
        self._test_connection()

    def _test_connection(self):
        """Teste Verbindung zu Home Assistant."""
        try:
            response = requests.get(f"{self.url}/api/", headers=self.headers,
timeout=5)
            response.raise_for_status()
            print("✅ Home Assistant verbunden")
        except Exception as e:
            print(f"❌ Home Assistant Verbindung fehlgeschlagen: {e}")
            raise

    def get_entities(self) -> List[Dict]:
```

```
        """Gibt alle Entities zurück."""
        response = requests.get(f"{self.url}/api/states", headers=self.headers)
        return response.json()

    def get_entity_state(self, entity_id: str) -> str:
        """Gibt State einer Entity zurück."""
        response = requests.get(f"{self.url}/api/states/{entity_id}",
headers=self.headers)
        data = response.json()
        return data.get('state', 'unknown')

    def call_service(self, domain: str, service: str, entity_id: str, **kwargs):
        """Ruft einen Service auf."""
        data = {"entity_id": entity_id, **kwargs}

        response = requests.post(
            f"{self.url}/api/services/{domain}/{service}",
            headers=self.headers,
            json=data
        )

        return response.status_code == 200

    # Convenience-Methoden
    def light_on(self, entity_id: str, brightness: Optional[int] = None, rgb:
Optional[List[int]] = None):
        """Schalte Licht an."""
        kwargs = {}
        if brightness:
            kwargs['brightness'] = brightness
        if rgb:
            kwargs['rgb_color'] = rgb

        return self.call_service("light", "turn_on", entity_id, **kwargs)

    def light_off(self, entity_id: str):
        """Schalte Licht aus."""
        return self.call_service("light", "turn_off", entity_id)

    def set_temperature(self, entity_id: str, temperature: float):
        """Setze Temperatur."""
        return self.call_service("climate", "set_temperature", entity_id,
temperature=temperature)

    def switch_on(self, entity_id: str):
        """Schalte Schalter an."""
```

```
        return self.call_service("switch", "turn_on", entity_id)

    def switch_off(self, entity_id: str):
        """Schalte Schalter aus."""
        return self.call_service("switch", "turn_off", entity_id)


# Demo
if __name__ == "__main__":
    ha = HomeAssistantManager()

    # Liste Entities
    entities = ha.get_entities()
    print(f"Entities: {len(entities)}")

    # Schalte Licht an
    ha.light_on("light.wohnzimmer", brightness=128)
```

Beispiel-Befehle

Voice Command Parser

```python
def parse_home_command(text: str, ha_manager: HomeAssistantManager) -> bool:
    """Parse Home Assistant Befehle."""
    text_lower = text.lower()

    # Licht-Befehle
    if "licht" in text_lower:
        # Extrahiere Raum
        room = None
        if "wohnzimmer" in text_lower:
            room = "wohnzimmer"
        elif "schlafzimmer" in text_lower:
            room = "schlafzimmer"
        elif "küche" in text_lower:
            room = "kueche"

        if not room:
            return False

        entity_id = f"light.{room}"

        # An/Aus
        if "an" in text_lower or "ein" in text_lower:
            ha_manager.light_on(entity_id)
            return True
        elif "aus" in text_lower:
            ha_manager.light_off(entity_id)
            return True

    # Heizung
    if "heizung" in text_lower or "temperatur" in text_lower:
        # Extrahiere Temperatur
        import re
        match = re.search(r'(\d+)\s*grad', text_lower)

        if match:
            temp = int(match.group(1))
            ha_manager.set_temperature("climate.heizung", temp)
            return True

    # Kaffeemaschine
```

```
    if "kaffee" in text_lower:
        if "start" in text_lower or "an" in text_lower:
            ha_manager.switch_on("switch.kaffeemaschine")
            return True

    return False


# Integration in Voice Assistant
def execute_command(text: str):
    """Führe Befehl aus (erweitert mit Home Assistant)."""

    # Versuche Home Assistant
    try:
        ha = HomeAssistantManager()
        if parse_home_command(text, ha):
            return True
    except:
        pass  # Home Assistant nicht verfügbar

    # Fallback: Normale Befehle
    # ... (wie bisher)
```

Zusammenfassung

Quick Start

1. **Installiere Home Assistant** (Raspberry Pi oder Docker)
2. **Erstelle Access Token** (Web-Interface)
3. **Installiere Python-Library:** `pip install HomeAssistant-API`
4. **Teste Verbindung** mit Beispiel-Code
5. **Integriere in Voice Assistant** mit HomeAssistantManager

Nächste Schritte

1. ✅ Home Assistant installieren
2. ✅ Geräte hinzufügen (Lichter, Thermostate, etc.)

3. ✅ Access Token erstellen
4. ✅ HomeAssistantManager testen
5. ✅ Voice Commands erweitern
6. ✅ Feedback-System (TTS: "Licht eingeschaltet")

Dokumentende
