SYSTEMDOKUMENTATION & FUNKTIONSÜBERSICHT • STAND: 20.11.2026

KAI Control Center (v4.0) – Bedienungsanleitung

Zentrale Steuerungs-, Diagnose- und Sicherheitssoftware des humanoiden Roboters KAI (`/usr/local/newpykai/app.py`)

1. Dashboard-Kopfzeile & Notfallsteuerung

Die obere Steuerleiste des Dashboards ermöglicht jederzeit den sofortigen Zugriff auf sicherheitsrelevante Notfallfunktionen sowie visuelle Anpassungen der Benutzeroberfläche. Zusätzlich sorgt eine automatische Startansage dafür, dass die Betriebsbereitschaft akustisch und über die Mundsimulation gemeldet wird.

> **SICHERHEIT – Notstopp & Torque Off:** Ein Klick auf **EMERGENCY STOP (ALL SERVOS TORQUE OFF)** sendet augenblicklich ein MQTT-Signal (`kai/emergency_stop` mit `action: "stop"`). Alle Servomotoren werden sofort kraftlos geschaltet und der Roboter bestätigt dies akustisch (*„Alle Motoren sind jetzt ausgeschaltet“*).

- **Reaktivieren (Torque On):** Stellt die Stromversorgung und Haltekraft aller Servomotoren wieder her (`action: "reset"`), synchronisiert das automatische Blinzeln und meldet die Betriebsbereitschaft per Sprachausgabe.
- **Design-Wahl (Farbschemata):** Über das Popover-Menü oben rechts stehen fünf Farbschemata zur Auswahl: *Blau (Standard)*, *Grün (Matrix)*, *Violett (Cyber)*, *Orange (Industrie)* sowie *Rot (Alarm)*.

2. Kai-Cam IP-Stream & Live-Hardware-Telemetrie

In der linken Dashboard-Spalte werden der externe Raum- und Roboter-Stream sowie alle elektrischen und thermischen Sensorwerte in Echtzeit überwacht:

- **Kai-Cam Live-Stream:** Bindet einen externen IP-Kamerastream (Standard: `http://192.168.6.75:8889/cam/`) responsiv und überlappungsfrei ein. Über das Aufklappmenü *Kai-Cam URL & Format* lassen sich Stream-URL, Aktiv-Status und Seitenverhältnis (`16 / 9` , `4 / 3` , `16 / 10` , `21 / 9`) persistent in `camera_config.json` speichern.
- **USB-Powermeter Live (TC66C):** Liest über `/dev/ttyACM1` (AES-entschlüsseltes Protokoll) im 1,5-Sekunden-Takt Spannung (V), Strom (mA), Leistung (W) sowie die kumulierte Energie (Wh) und Ladung (mAh) aus. Über **Reset** lassen sich die kumulierten Zähler nullen.
- **Servo-Powermeter Live (Adafruit INA260):** Überwacht über den I²C-Bus (Adresse `0x40`) die Versorgungsspannung, Stromaufnahme, Leistung sowie den kumulierten Energieverbrauch der Servomotoren inkl. eigener Reset-Funktion.
- **Raspberry Pi 5 Hardware Live:** Zeigt kontinuierlich die aktuelle CPU-Temperatur (°C), die Lüfter-Drehzahl (U/min), das PWM-Tastverhältnis (%) sowie die aktive Kühlstufe des Pi 5 an.

3. System-Services, Pi-System & 3-stufiges Backup

Das Control Center überwacht acht autarke Systemd-Hintergrunddienste mit einem 2-Sekunden-Live-Polling. Jeder Schaltzustand (Start/Stopp) wird persistent in `services_config.json` gesichert und beim Neustart von `app.py` automatisch wiederhergestellt.

Dienstname	Systemd Unit	Funktionsbeschreibung
Control Center App	`kai-dashboard.service`	Web-UI Dashboard & zentrale Bedienoberfläche.
Audio Server	`rhasspy-audio.service`	Erfassung und Verarbeitung der Mikrofon-Audiodaten.
Wake Word	`rhasspy-wake.service`	Erkennung des Aktivierungsworts („Hey Kai“ / „Kai“).
STT/NLU Brain	`rhasspy-brain.service`	Spracherkennung und Absichts-Interpretation (Intent-Erkennung).
TTS Voice	`rhasspy-tts.service`	Sprachsynthese-Ausgabe über `espeak-ng`.
LEDs	`rhasspy-leds.service`	Ansteuerung der Status-LEDs und optischen Signale.
Robot Handler	`rhasspy-handler.service`	Zentrale Servosteuerung, Gesten, Blinzeln, Mund und Smalltalk-Engine.
Gesichtserkennung	`kai-facerec.service`	Kamerabasierte Personen- und Blickkontakterkennung (4s-Trigger).

- **app.py Update:** Ermöglicht im Menü *Pi-System* das direkte Hochladen einer neuen `app.py` mit anschließendem automatischem Dienst-Neustart.
- **Raspberry Pi Power:** Neustart und Herunterfahren sind durch das Sicherheitspasswort `newpykai` geschützt. Vor der Ausführung spricht KAI eine Ankündigung mit synchroner Mundbewegung aus und wartet deren Ausklingen ab.
- **3-stufiges Backup-System:** Erstellt ein vollständiges ZIP-Archiv aller Konfigurationen und sichert dieses: (1) lokal unter `/usr/local/newpykai/backups`, (2) per SCP auf dem lokalen PC (`192.168.6.18:/ExtraData/backup_newpykai`) sowie (3) via SSH-Key auf dem externen Server (`root@www.mindach.net:/usr/local/newpykai_backup`). Zusätzlich steht ein direkter Browser-Download bereit.

4. Dual-Kamera Stereo-Vision, NPU-Verpixelung & Mimik

Über den integrierten Multi-Thread MJPEG-Videoserver (Port `8088`) werden beide Augen-Kameras unabhängig voneinander gestreamt (`http://<PI_IP>:8088/video_feed?cam=X`). Alle Einstellungen werden in `camera_config.json` gespeichert.

- **Auge 1 & Auge 2:** Einstellbarer V4L2-Kamera-Index (0–9) sowie individuelle Ein/Aus-Schalter zur Ressourcenschonung.
- **Gesicht verpixeln (Demo-Anonymität auf Auge 2):** Erkennt Gesichter und Köpfe im Vorder- und Hintergrund mithilfe von OpenCV-Detektoren und der **Hailo-8L NPU** (YOLOv8s Personen-/Kopf-Tracking). Pixel-Blockgröße (6–40 px) sowie ein Nachlauf-Puffer (0,5–6,0 s gegen kurzes Aufblitzen bei Drehungen) sind feinjustierbar.
- **Automatisches Blinzeln (Gruppe `lider`):** Ein/Aus-Schalter für realistisches, zufälliges Blinzeln der Servos ID 22 & 28 sowie Button *Jetzt blinzeln*. Minimal-/Maximalpause, Lid-Schlussdauer (ms) und Geschwindigkeit (ms) werden persistent in `blink_state.json` gespeichert.
- **Mundbewegung beim Sprechen (ID 7):** Direkter Ein/Aus-Schalter auf der Hauptseite zum Aktivieren oder Deaktivieren der automatischen Mundbewegung bei Sprachausgaben (`mouth_config.json`).

5. Spracherkennung (ASR), Verlauf & Datumsabfragen

Der Bereich *Spracherkennung Live-Test* visualisiert die Spracheingabe im Sekundentakt und steuert die Aufnahme über MQTT:

- **Jetzt sprechen (Aufnahme starten):** Sendet gleichzeitig `hermes/hotword/default/detected` und `hermes/asr/startListening`, um das Mikrofon direkt ohne Wakeword zu öffnen.
- **Live-Sprachbox & Wiederholung:** Zeigt den zuletzt erkannten Satz, die farblich kodierte Fuzzy-Erkennungsrate (Grün ≥ 80 %, Gelb ≥ 65 %, Orange < 65 %), die ausgeführte Geste und den Zeitstempel an. Mit *Letzten Befehl erneut senden* wird der Text erneut injiziert.
- **Verlauf der letzten 20 erkannten Sätze:** Aufklappbare Historie (`/tmp/speech_history.json`) mit Zeitstempel, Erkennungsrate und Aktion. Jeder historische Satz kann ausgewählt und erneut ausgeführt oder der gesamte Verlauf geleert werden.
- **Text-Befehl Injektion & Datumsfragen:** Ermöglicht das Eintippen von Befehlen ohne Mikrofon. Fragen nach Datum (*„Welcher Tag ist heute?“*) oder Uhrzeit (*„Wie spät ist es?“*) werden direkt über `date_time_handler.py` beantwortet.

6. Audio- & Sprachkonfiguration (Vollbild-Unterseite)

Über den Navigations-Button **Audio & Sprach-Konfiguration** öffnet sich eine Vollbild-Unterseite zur Steuerung der Sprachausgabe und Systemlautstärke:

- **espeak-ng Parameter:** Auswahl der Stimme (`mb-de6`, `mb-de1`, `de`, `de+m3` u. a.), Sprechgeschwindigkeit (80–350 WPM) und Tonhöhe (0–99) mit Speicherung in `tts_config.json`, Reset auf Vorgabewerte und direkter Audio-Probe.
- **System-Lautstärkeregelung (ALSA):** Stufenloser Schieberegler (0–100 %), Stummschaltung sowie Sinus-Testton-Ausgabe.
- **Audio- & Interaktionstests:** Vorlesen beliebiger Testtexte sowie Schnellausgabe von Datum, Uhrzeit oder kombiniertem Datum/Zeit-Satz mit einstellbarer Mund-Synchronisation (`Sek / Zeichen`).

7. Code & Slot Editor (Dashboard & Vollbild-Unterseite)

Der Editor erkennt alle Konfigurationsdateien aus `/usr/local/newpykai/config` sowie Python-Skripte dynamisch. Über **Vollbild öffnen** steht er in maximaler Breite zusammen mit dem Action-Handler Live-Log und der Befehlsinjektion bereit.

Editor / Datei	Funktion & Besonderheiten
Smalltalk & Fragen `smalltalk_config.json`	Tabellen-Editor für Kategorien, Schlüsselwörter, Antworten (`static` , `dynamic` , `internet` /Wikipedia-Caching) sowie Schnell-Zuweisung nicht erkannter Phrasen (*Unmatched*).
Bewegungen & Befehle `movements_config.json`	Verknüpft gesprochene Befehle und Antworten mit einer Ziel-Geste (`action`) aus `gestures_config.json` inkl. eigener Verwaltung für *Unmatched Bewegungen*.
Gesten & Positionen `gestures_config.json`	Definiert Einzel-Servos oder Servogruppen (z. B. `lider`). Mit *Werte aus servo_config.json füllen* werden Min/Max-Grenzen automatisch übernommen; *roboter.yaml sync* aktualisiert die Grammatik.
Servo Limits `servo_config.json`	Pflege von Servo-ID, Bezeichnung, Typ (`SC` / `ST`), Min, Max, Neutral, Maximalgeschwindigkeit, Invertierung (gegengleich) und Gruppenzugehörigkeit.
Gesichtserkennung `face_config.json`	Konfiguration von Toleranz-Schwellwert, 4s-Blickkontakt-Dauer für bekannte/unbekannte Personen, Blinzel-Toleranz, Cooldowns und Frame-Rate.
Bekannte Personen DB `known_persons`	Datenbank-Editor zum Umbenennen gespeicherter Personen, Anpassen individueller Erkennungstoleranzen (0,40–0,70) oder Löschen von Einträgen.
Grammatik & Code `roboter.yaml` & `*.py`	Tabellen- und Raw-Editor für Rhasspy-Intents sowie direkter Quellcode-Editor für `action_handler.py` , `motion_manager.py` , `eliza_engine.py` u. v. m.

8. Action-Handler Live-Log & MQTT Monitor

Im unteren Bereich der rechten Spalte ermöglichen zwei Echtzeit-Monitore die lückenlose Nachverfolgung aller Systemabläufe:

- **Action-Handler Live-Log (`rhasspy-handler.service`):** Zeigt die aktuellen `journalctl` -Meldungen im Sekundentakt. Über Mehrfachauswahl-Filter lassen sich hochfrequente Servos (standardmäßig z. B. Mund ID 7) ausblenden oder gezielt einzelne Servo-IDs isolieren. Ein **Copy-Button** kopiert das gefilterte Log direkt in die Zwischenablage.
- **MQTT Live-Daten Monitor:** Abonniert alle Topics (`#`) auf `localhost:1883` und stellt die letzten 50 Nachrichten (Zeit, Topic, Payload) tabellarisch dar.

9. Hardware-Servo Diagnose (Vollbild-Unterseite)

Die Diagnose-Unterseite (erreichbar über **Hardware-Servo Diagnose**) dient der direkten Inbetriebnahme, Parametrierung und Fehlersuche am seriellen Bus der SC- und ST-Servos:

- **1. Bus-Scan (konfigurierbarer ID-Bereich):** Scannt thread-sicher über MQTT (`kai/servo/scan_request`) den frei wählbaren Adressbereich (Standard: ID 1 bis 30) und listet alle antwortenden Servos auf.
- **2. Servo-ID Umprogrammieren:** Schreibt eine neue EEPROM-ID in den ausgewählten Servo (mit Auswahl zwischen SC-Serie und ST-Serie).
- **3. Betriebsmodus umschalten:** Schaltet einzelne Servos zwischen Winkel-Modus (*Servo-Modus*) und Endlos-Rotationsmodus (*Motor-Modus*) um.
- **4. Einzel-Servo Echtzeit-Steuerung & Telemetrie:** Live-Positions-Slider mit Laufzeitregelung (10–1000 ms), Auslesen von Ist-Position, Versorgungsspannung (V) und Betriebsmodus, Anpassung und Speicherung von Min/Max-Grenzwerten sowie *Mitte speichern* direkt in `servo_config.json`.
- **5. Synchronisierte Servogruppen-Steuerung:** Steuert vordefinierte sowie dynamisch aus `servo_config.json` geladene Gruppen (z. B. *Linke Hand*, *Beide Lider*, *Augen-Gruppe*, *Arm-Gruppen*, *Kopf-Gruppe*) prozentual (0–100 %) unter automatischer Berücksichtigung individueller Min/Max- und Invertierungs-Werte.