

Automatisierte Vogelbeobachtung mit Raspberry Pi, Python und KI

Adrian Imme und Roland Imme

LinuxDay Dornbirn

27. September 2025



Vogelhaus mit 2 Kameras

Inhaltsverzeichnis

1 Einleitung

2 Technik & Hardware

3 Software & Workflow

4 Beispieldideos

5 Fazit & Fragen

Was macht jeder mit KI?

- ChatGPT & Large Language Models
- Generative AI für Bilder (DALL-E, Midjourney)
- AI-Coding-Assistenten (GitHub Copilot)
- Machine Learning as a Service (MLaaS)
- Autonome Fahrzeuge & Robotik
- AI-Voice-Assistenten & Sprachsynthese

Und wir?

Praktische Anwendung mit Raspberry Pi!

Einleitung

- Ziel des Vortrags (Erfahrungsaustausch)
- Motivation: Warum Vogel-Videos?
- Entstehung durch anderes Projekt:
 - Kamerawagen-Linux
 - Erfahrungen mit Raspberry Pi Kameras
 - Übertragung auf Vogelbeobachtung



Kamerawagen-Linux

YouTube-Kanal



Technik & Hardware

- Vogelhaus
- Raspberry Pi 5
- AI Kit
- Kameratypen
- Mikrofon
- Stromversorgung
- Montage und Standortwahl

Das Vogelhaus – 3D-Druck und Konstruktion

- **3D-Druck Technologie**

- Komplettes Vogelhaus aus dem 3D-Drucker
- Modularer Aufbau aus mehreren Komponenten

- **Zusammenbau und Verbindungstechniken**

- Schraubverbindungen
- Karabinerverbindungen
- Präzise Passgenauigkeit

- **Vorteile des modularen Designs**

- Einfache Wartung und Reinigung
- Austauschbare Kameramodule
- Anpassbare Konfiguration



3D-gedrucktes Vogelhaus

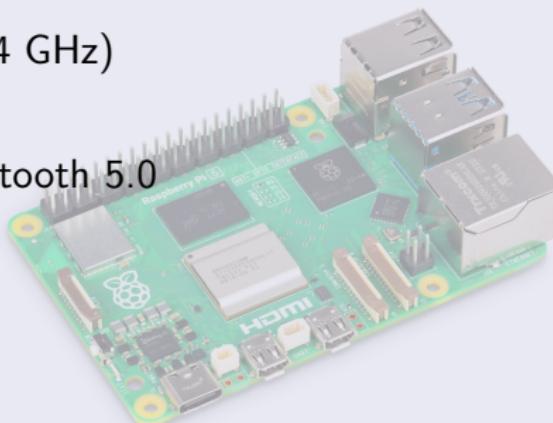
Modulare Bauweise

Raspberry Pi 5 – Technik

Eingesetzte Hardware

Raspberry Pi 5 mit 8GB RAM

- Quad-Core ARM Cortex-A76 CPU (2.4 GHz)
- 8GB LPDDR4X RAM
- Gigabit Ethernet, WiFi 802.11ac, Bluetooth 5.0
- USB 3.0, USB-C Power
- HDMI, GPIO, Kameraanschluss
- Für Aufnahme und Remotezugriff

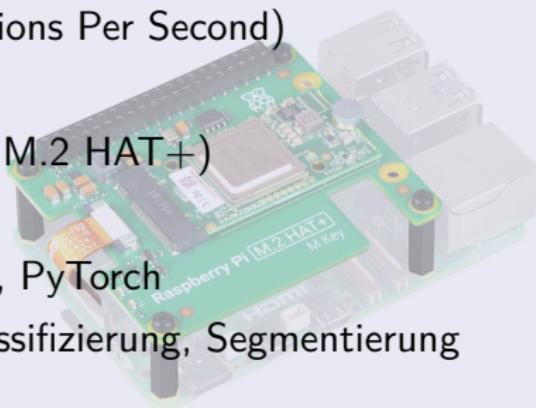


Hersteller: <https://www.raspberrypi.com/products/raspberry-pi-5/>

AI Kit für Raspberry Pi, M.2 HAT + Hailo-8L AI Accelerator

Technische Spezifikationen

- **AI-Prozessor:** Hailo-8L Neural Processing Unit (NPU)
- **Performance:** 13 TOPS (Tera Operations Per Second)
- **Anschluss:** M.2 2242 Key-E Slot
- **Kompatibilität:** Raspberry Pi 5 (über M.2 HAT+)
- **Stromverbrauch:** < 2W typisch
- **Frameworks:** TensorFlow Lite, ONNX, PyTorch
- **Anwendungen:** Objekterkennung, Klassifizierung, Segmentierung
- **Betriebstemperatur:** 0°C bis +70°C
- **Abmessungen:** 22 × 30 × 2,3 mm (M.2 2242)



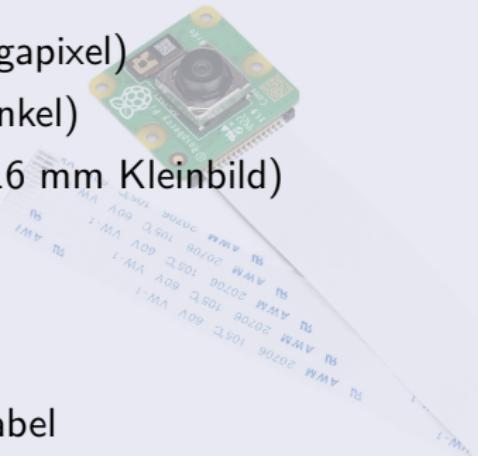
Hersteller: <https://www.raspberrypi.com/products/ai-kit/>

Kameratypen

Raspberry Pi Camera Module 3 Wide, 12MP

Technische Spezifikationen

- **Sensor:** Sony IMX708 CMOS
- **Auflösung:** 4608×2592 Pixel (12 Megapixel)
- **Sichtfeld:** 120° diagonal (Ultra-Weitwinkel)
- **Brennweite:** 2,75 mm (äquivalent zu 16 mm Kleinbild)
- **Blende:** f/2.2
- **Video:** 4K@30fps, 1080p@60fps
- **HDR-Unterstützung:** Ja
- **Anschluss:** 22-poliger CSI-Flachbandkabel
- **Abmessungen:** $25 \times 24 \times 11,5$ mm

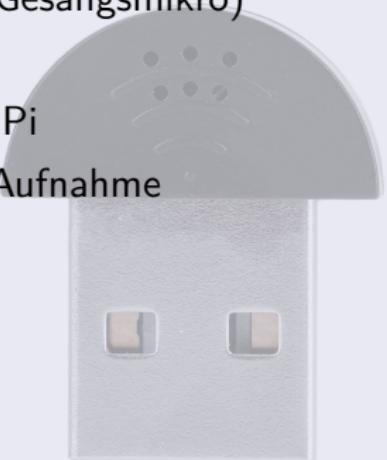


Hersteller: <https://www.raspberrypi.com/products/camera-module-3/>

Dioche USB-Gesangsmikrofon

Audio-Setup für Vogelbeobachtung

- **Mikrofontyp:** USB-Kondensatormikrofon (Gesangsmikro)
- **Anschlüsse:** USB & 3,5mm Klinke
- **Kompatibilität:** PC, Notebook, Raspberry Pi
- **Richtcharakteristik:** Niere für fokussierte Aufnahme
- **Frequenzbereich:** 20 Hz - 20 kHz
- **Sampling-Rate:** 48 kHz / 16 Bit
- **Stromversorgung:** USB-Bus-powered
- **Design:** Tragbares Studio-Sprachmikrofon
- **Farbe:** Schwarz
- **Software:** ALSA-Support unter Linux



Raspberry Pi 27W USB-C Netzteil

Technische Spezifikationen

- **Leistung:** 27W (5.1V / 5A)
- **Anschluss:** USB-C für Raspberry Pi 5
- **Kabel:** 1,5m fest angeschlossen
- **Effizienz:** > 80% (Energieeffizienzklasse VI)
- **Schutzfunktionen:** Überspannung, Überstrom, Kurzschluss
- **Betriebstemperatur:** 0°C bis +40°C
- **Gehäuse:** Wetterfest für Außenmontage
- **Zusatzversorgung:** USB-A Port für Peripherie
- **Zertifizierungen:** CE, FCC, RoHS



Hersteller: <https://www.raspberrypi.com/products/27w-power-supply/>

Montage und Standortwahl

Standort: Terrasse

- **Standortwahl:**

- Gute Sicht auf Futterstellen
- Ruhige Umgebung für Vogelbeobachtung
- Stabile Montageoptionen verfügbar

- **Montage-Aspekte:**

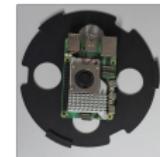
- Sichere Befestigung des Vogelhauses
- Optimale Kameraausrichtung
- Wetterschutz für Elektronik
- Einfacher Zugang für Wartung

- **Technische Überlegungen:**

- WLAN-Reichweite zum Router
- Stromversorgung in der Nähe
- Schutz vor direkter Sonneneinstrahlung



Vogelhaus-Setup



Raspberry Pi 5

Software & Workflow

- Kamera-Software
- Aufnahme-Parameter
- Remote-Steuerung & AI-Automatisierung
- Code-Beispiele
- GitHub Repository
- Videoschnitt

AI-Enhanced Videoaufnahme mit Raspberry Pi

Kamera & Audio-Aufnahme mit KI-Integration

- **Kamera-Software:**

- `rpicam-vid` mit AI-Modul Integration
- Single-Kamera-Setup mit Hailo YOLOv8 Inference
- Post-Processing: `hailo_yolov8_inference.json`
- Konfigurierbare Kamera-ID (0 oder 1)
- HDR-Unterstützung und Autofokus

- **Audio-Aufnahme:**

- Automatische USB-Audio-Geräte-Erkennung
- `arecord` für Audio-Capture unter Linux
- Parallele Audio-Video-Aufnahme mit Threading

AI-Konfiguration & Video-Processing

Parameter und KI-Enhanced Verarbeitung

- **Aufnahme-Parameter:**

- H.264 Codec für optimale Kompression
- 4K Standard-Auflösung (4096x2160)
- Framerate-Einstellungen (Standard: 15fps)
- Config-basierte Pfad-Organisation
- Versionsverwaltung mit --version

- **Video-Processing:**

- `ffmpeg` für Audio-Video-Kombination
- H.264 zu MP4 Konvertierung mit AAC Audio
- Automatische Datei-Bereinigung
- Fortschrittsanzeige mit `tqdm`

Python-basierte KI-Integration

Fernsteuerung und AI-Workflow-Automatisierung

- **Remote-Steuerung:**

- SSH/SCP für Remote-Zugriff (paramiko, SCPClient)
- Config-Klasse für zentrale Konfiguration
- Automatische Konfigurationsvalidierung
- Erreichbarkeitsprüfung und Fehlerbehandlung

- **AI-Video-Processing:**

- Hailo YOLOv8 Real-time Objekterkennung
- ffmpeg für Audio-Video-Synchronisation
- MP4-Export mit Zeitstempel und Auflösung
- Automatische H.264/WAV zu MP4 Konvertierung

Intelligente Automatisierung & Prozess-Management

Moderne Python-Automatisierung für nahtlose Abläufe

- **Environment-basierte Konfiguration:**

- .env-Dateien für sichere Credential-Verwaltung
- Zentrale Konfiguration von SSH-Zugangsdaten
- Trennung von Code und Konfiguration
- Einfache Anpassung ohne Code-Änderungen

- **Automatische Verzeichnisstruktur:**

- AI-HAD Prefix für eindeutige Identifikation
- Zeitstempel-basierte Ordnererstellung
- Automatische Pfad-Validierung und -Erstellung

- **Robuste Prozess-Verwaltung:**

- Signal-Handler (SIGINT, SIGTERM) für sauberes Beenden
- Threading für parallele Audio-Video-Aufnahme
- Fehlerbehandlung und Logging

Code Beispiel 1: AI-Enhanced Kamera-Kommando

rpicam-vid mit Hailo YOLOv8 AI-Integration

```
# AI-Enhanced Video-Aufnahme mit Real-time Objekterkennung
def get_remote_video_command():
    remote_path = config.get_remote_video_path(year, timestamp)
    roi_param = f"--roi {args.roi}" if args.roi else ""
    return f"""
mkdir -p {remote_path} &&
cd {remote_path} &&
rpicam-vid --camera {args.cam} --hdr {args.hdr} \
--post-process-file /usr/share/rpi-camera-assets/hailo_yolov8_inference.json \
--width {args.width} --height {args.height} --codec {args.codec} \
--rotation {args.rotation} --framerate {args.fps} \
--autofocus-mode {args.autofocus_mode} {roi_param} \
-o "video.h264" -t {recording_duration_s * 1000} & \
arecord -D {audio_device} -f S16_LE -r 44100 -c 1 \
-t wav -d {recording_duration_s} {remote_path}/audio.wav
"""
```

Code Beispiel 2: SSH Remote-Steuerung

Sichere Remote-Verbindung mit Paramiko

```
# Sichere SSH-Verbindung mit automatischer Host-Key-Verwaltung
def execute_remote_command(command):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        ssh.connect(remote_host['hostname'],
                    username=remote_host['username'],
                    key_filename=remote_host['key_filename'])

        stdin, stdout, stderr = ssh.exec_command(command)
        output = stdout.read().decode()
        print(f"Ausgabe auf {remote_host['hostname']}: {output}")

        # Warte, bis der Befehl abgeschlossen ist
        stdout.channel.recv_exit_status()
        ssh.close()
    except Exception as e:
        print(f" Fehler bei der Verbindung: {e}")
```

Code Beispiel 3: Threading & Signal-Handler

Parallele Prozesse und sauberes Beenden

```
# Signal-Handler für Graceful Shutdown
def signal_handler(sig, frame):
    print("Beenden des Skripts...")
    stop_event.set()

# Setze den Signal-Handler
signal.signal(signal.SIGINT, signal_handler)

# Threading für parallele Audio-Video-Aufnahme
stop_event = threading.Event()
threads = []

video_thread = threading.Thread(target=execute_remote_command,
                                 args=(get_remote_video_command(),))
threads.append(video_thread)
video_thread.start()

# Fortschrittsanzeige mit tqdm
progress = tqdm(total=recording_duration_s, desc="Fortschritt", unit="s")
for _ in range(recording_duration_s):
    if stop_event.is_set():
        break
    time.sleep(1)
    progress.update(1)
```

Code Beispiel 4: Automatische USB-Audio-Erkennung

Intelligente Hardware-Erkennung auf Remote-Host

```
# USB-Audio-Gerät automatisch ermitteln
def get_usb_audio_device_remote():
    try:
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(remote_host['hostname'],
                    username=remote_host['username'],
                    key_filename=remote_host['key_filename'])

        # Führe arecord -l auf dem Remote-Host aus
        stdin, stdout, stderr = ssh.exec_command("arecord -l")
        output = stdout.read().decode()
        ssh.close()

        # Suche nach einem Gerät mit "USB" im Namen
        for line in output.splitlines():
            if "USB" in line:
                parts = line.split()
                card_index = parts[1].replace(":", "") # Karte
                subdevice_index = "0" # Standard-Subgerät
                return f"hw:{card_index},{subdevice_index}"
        return None
    except Exception as e:
        print(f" Fehler beim Ermitteln des USB-Audio-Geräts: {e}")
        return None
```

Code Beispiel 5: ffmpeg Audio-Video-Kombination

Automatische MP4-Konvertierung mit Zeitstempel

```
# Konvertierung von H.264 + WAV zu MP4 mit ffmpeg
video_file = f"{base_path}/video.h264"
audio_file = f"{base_path}/audio.wav"
mp4_file = f"{base_path}/{timestamp}_{WIDTH_x_HEIGHT}.mp4"

# Überprüfen, ob die Audio-Datei existiert
if not os.path.exists(audio_file):
    print(f" Fehler: Die Audio-Datei {audio_file} wurde nicht gefunden.")
    exit(1)

# ffmpeg-Kommando für Audio-Video-Synchronisation
ffmpeg_command = f"ffmpeg -fflags +genpts -r {args.fps} -i {video_file} \\
    -i {audio_file} -c:v copy -c:a aac {mp4_file}"

process = subprocess.Popen(ffmpeg_command, shell=True,
                           stdout=subprocess.PIPE, stderr=subprocess.PIPE)
stdout, stderr = process.communicate()

if process.returncode == 0:
    print(f" Video erfolgreich konvertiert: {mp4_file}")
    # Lösche die ursprünglichen Dateien
    os.remove(video_file)
    os.remove(audio_file)
else:
    print(f" Fehler beim Ausführen von ffmpeg: {stderr.decode()}")
```

GitHub Repository

Open Source Projekt & Dokumentation

- **Öffentliches Repository** mit vollständigem Quellcode
- **Umfangreiche Wiki** mit Anleitungen und Dokumentation
- **Community-Beiträge** und Issues willkommen

vogel-kamera-linux

<https://github.com/kamera-linux/vogel-kamera-linux>



QR-Code zum Repository

Kdenlive für Video-Editing

Professioneller Videoschnitt mit Open Source

- **Kdenlive als Haupttool:**

- Professionelles Open-Source-Videoschnittprogramm
- Multi-Track-Timeline für komplexe Projekte
- Native Unterstützung für verschiedene Videoformate

- **Workflow-Integration:**

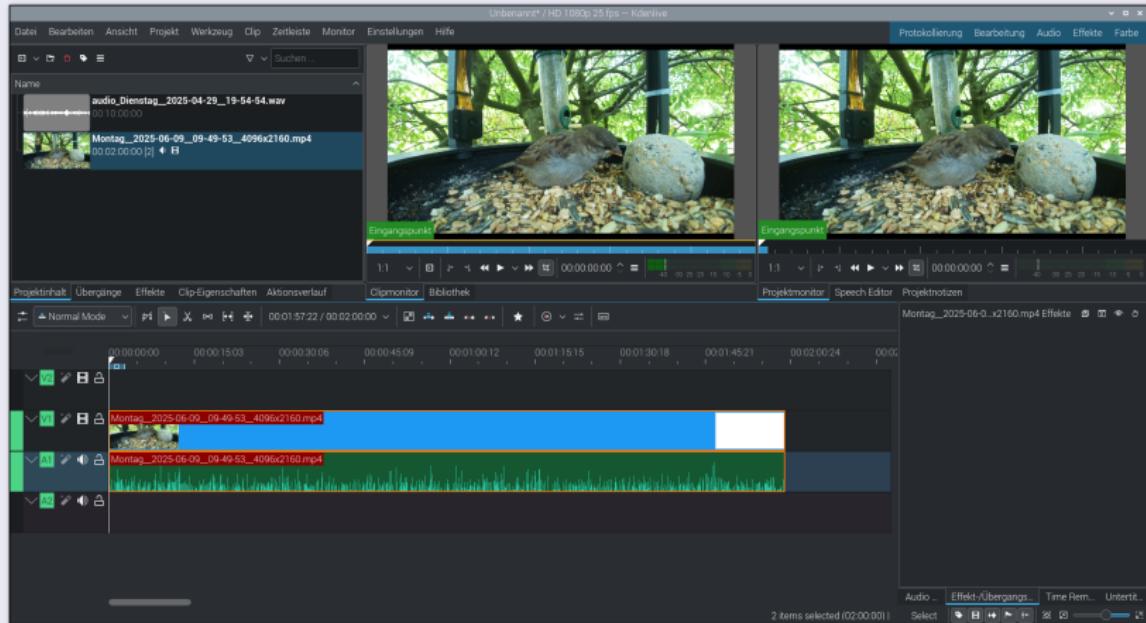
- Import der Split-Screen-Videos aus ffmpeg
- Schnitt und Montage der Vogelbeobachtungen
- Farbkorrektur und Stabilisierung

- **Export-Optionen:**

- Optimierung für YouTube (H.264, verschiedene Auflösungen)
- Anpassbare Qualitätseinstellungen
- Batch-Export für mehrere Videos

Kdenlive in Aktion

Benutzeroberfläche und Workflow



Kdenlive-Benutzeroberfläche beim Videoschnitt

YouTube-Kanal und Beispielvideos

Unser YouTube-Kanal

Vogel-Kamera-Linux



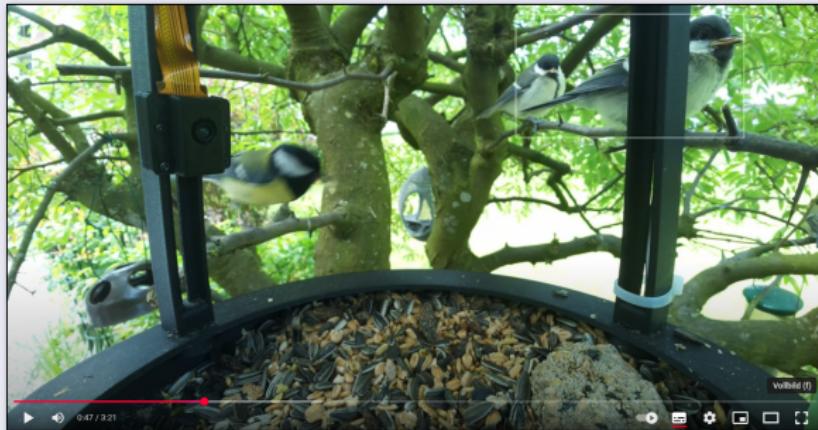
<https://www.youtube.com/@vogel-kamera-linux>



QR-Code zum Kanal

Beispielvideo

Typischer Ausschnitt aus der automatisierten Vogelbeobachtung



► Beispielvideo ab Sekunde 35

Die Wiedergabe muss ggf. manuell gestartet werden.

Fazit: Was haben wir erreicht?

Technische Erfolge

- **Hardware-Integration:** Raspberry Pi 5 + Hailo-8L AI Kit
- **AI-Enhanced Videoaufnahme:** YOLOv8 Real-time Objekterkennung
- **Modulares 3D-Design:** Wartungsfreundliches Vogelhaus
- **Remote-Automatisierung:** SSH + Python für nahtlose Steuerung

Praktische Erkenntnisse

- **Open Source Workflow:** Kdenlive + ffmpeg für Videoproduktion
- **Community-Projekt:** GitHub Repository mit Wiki-Dokumentation
- **Skalierbare Lösung:** Von Prototyp zu produktivem System
- **Praxistauglichkeit:** Stabile und vielseitige Vogelbeobachtung

Fragen & Diskussion

Mögliche Fragen:

- Welche Vögel besuchen euer Vogelhaus am häufigsten?
- Welche Herausforderungen hattet ihr beim 3D-Druck des Vogelhauses?
- Plant ihr weitere AI-Features wie Vogelarten-Klassifizierung?
- Habt ihr schon lustige Vogel-Videos aufgenommen?
- ... und viele mehr!



Fragen?

Vielen Dank!



Fragen, Diskussion & Networking

GitHub Repository:



vogel-kamera-linux

YouTube-Kanal:



@vogel-kamera-linux

Adrian Imme & Roland Imme