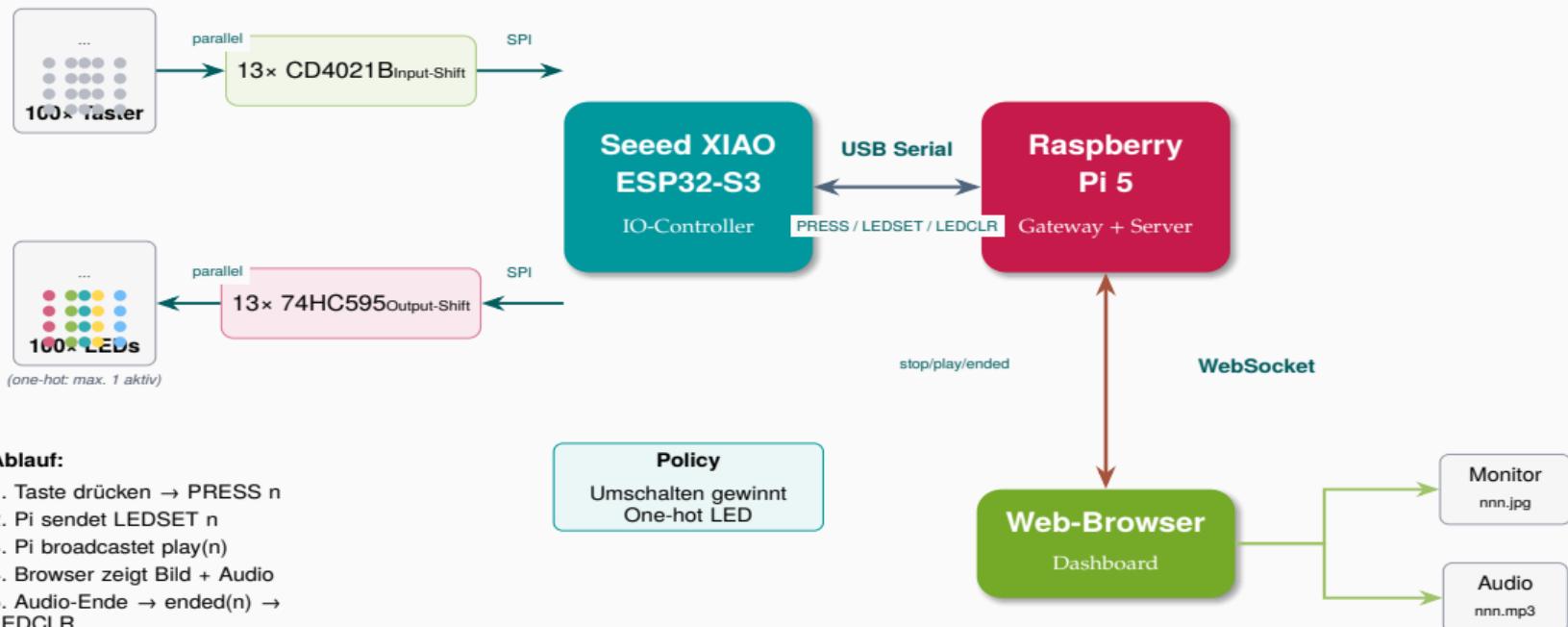
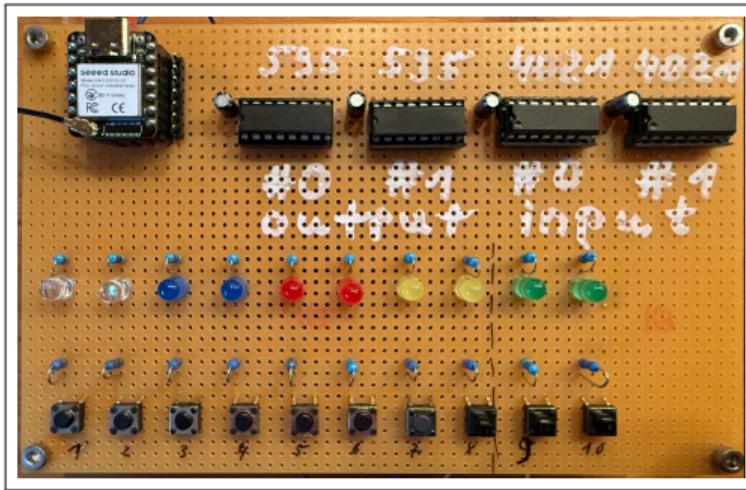


# Interaktives Auswahlpanel

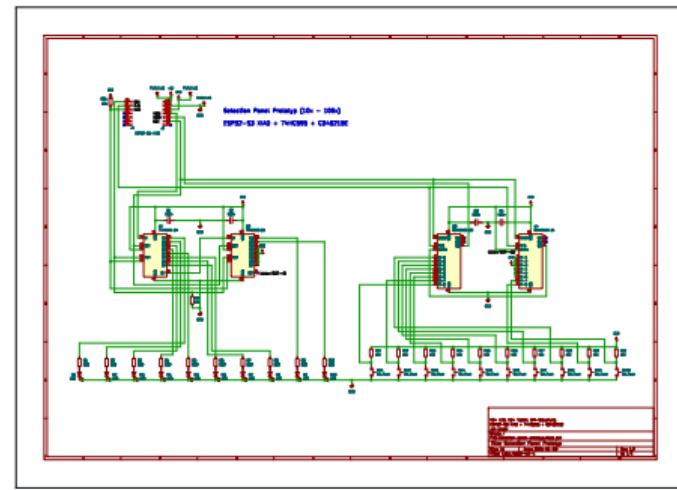
100× Taster/LED + 100× Bild/MP3



# Hardware-Prototyp v2



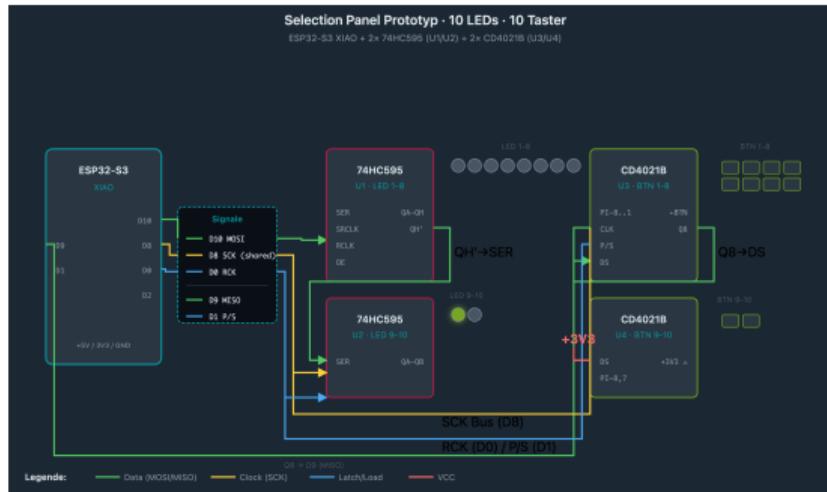
Bestückungsseite – ESP32 + ICs



KiCad – Schaltplan

**Komponenten:** ESP32-S3 XIAO · 2x 74HC595 · 2x CD4021B · 10 LEDs · 10 Taster

# Schieberegister-Architektur



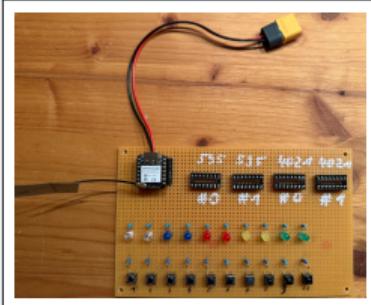
## Komponenten

- **ESP32-S3 XIAO** – Mikrocontroller
- **2x 74HC595** – LED-Ausgabe
- **2x CD4021B** – Taster-Eingabe

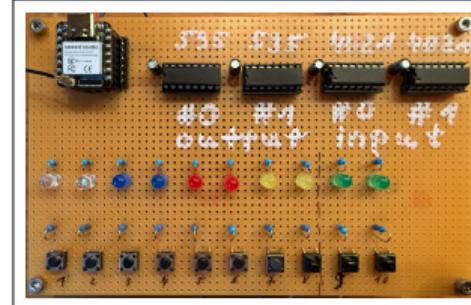
## Signale (SPI-ähnlich)

- **Data** – SER/Q8 (grün)
- **Clock** – SRCLK/CLK (gelb)
- **Latch** – RCLK/P/S (blau)

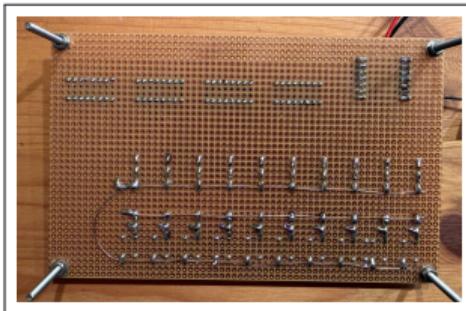
# Prototyp-Evolution



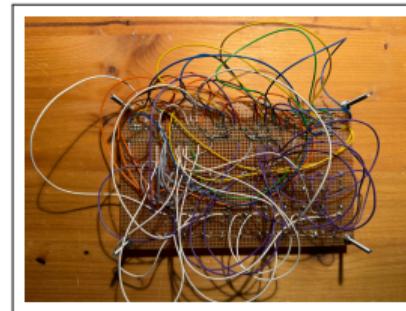
**v1 Bestückung** – Sockel ohne ICs



**v2 Bestückung**

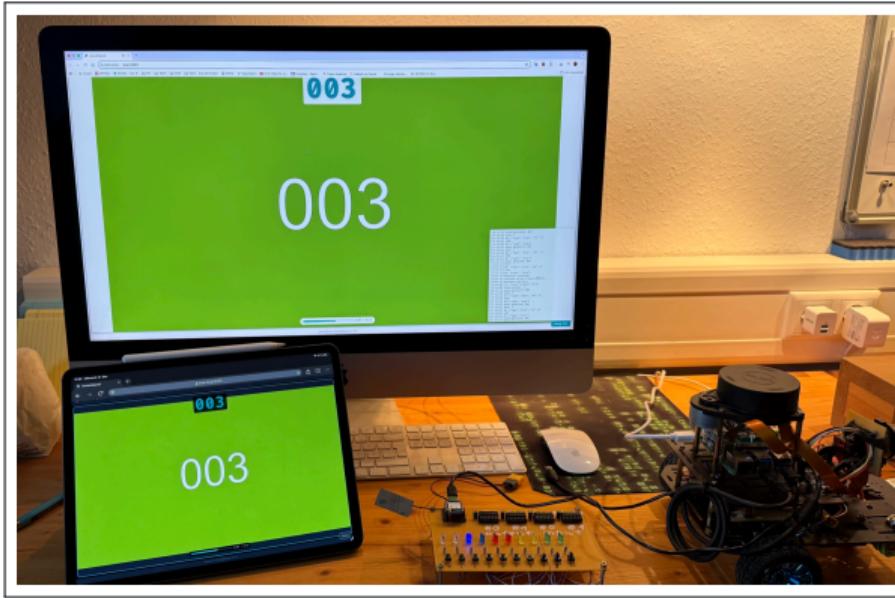


**v1 Lötseite**



**v2 Lötseite**

# Web-Dashboard



**Zugriff**  
rover:8080

**Multi-Device**  
iMac, iPad, iPhone

**4K-Medien**  
Farbschema-Bilder

**Synchron**  
WebSocket Broadcast

# Was ist das Auswahlpanel?

## Anwendungsfall

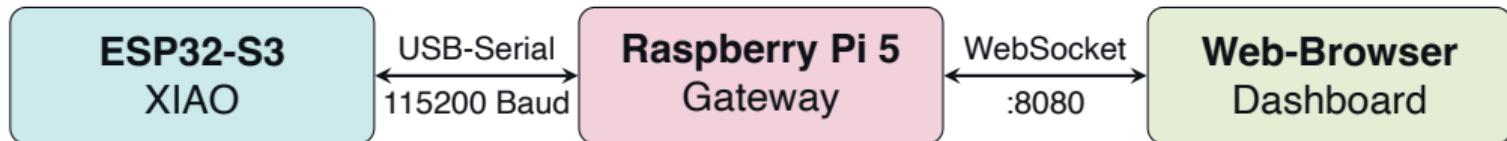
- ▶ 100 physische Taster als Eingabe
- ▶ Jeder Taster triggert ein Bild + Audio
- ▶ LED zeigt aktive Auswahl an

## Policy: „Preempt – Umschalten gewinnt“

- ▶ Neuer Tastendruck → **sofortiger Wechsel**
- ▶ **One-hot LED:** Immer nur eine LED aktiv
- ▶ Nach Audio-Ende → alle LEDs aus

**Merksatz:** Der letzte Tastendruck zählt – kein Warten, kein Queue.

# Systemarchitektur – Überblick



Komponente	Aufgabe
ESP32-S3 XIAO	100 Taster scannen, 100 LEDs steuern, Echtzeit
Raspberry Pi 5	Event-Gateway, WebSocket-Server, Media-Host
Browser	Bild fullscreen anzeigen, Audio abspielen, Ende melden

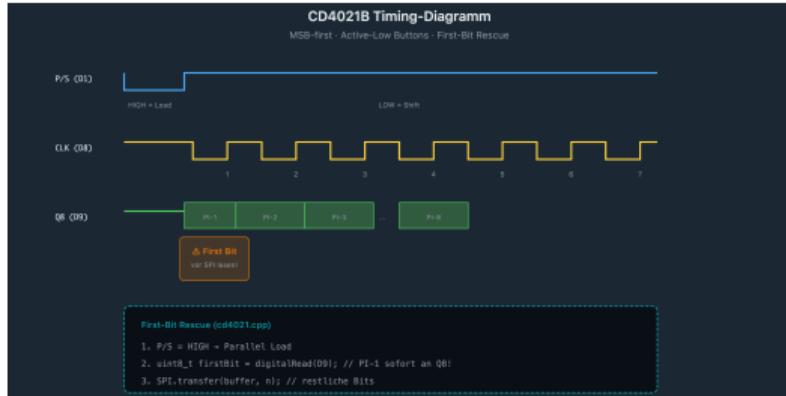
# Hardware – Schieberegister

## Warum Schieberegister?

- ▶ 100 IOs mit **6 GPIO-Pins**
- ▶ Kaskadierbar:  $13 \times 8 = 104$  Kanäle
- ▶ Günstig und robust

## Pinbelegung ESP32-S3 (gemeinsamer SPI)

Signal	Pin	Funktion
SCK	D8	SPI Clock (gemeinsam)
MOSI	D10	74HC595 SER
MISO	D9	CD4021B Q8
RCK	D0	74HC595 Latch
OE	D2	74HC595 Enable (PWM)
P/S	D1	CD4021B Load



CD4021B Timing-Diagramm

**Achtung:** CD4021B hat **invertierte** Load-Logik: P/S = **HIGH** für Load!

# Firmware – Modulare FreeRTOS-Architektur

## IO-Task (Core 1, Prio 5)

- ▶ Polling alle 5 ms (200 Hz)
- ▶ Debouncing pro Taste (30 ms)
- ▶ Events in FreeRTOS-Queue (8 Slots)

## Serial-Task (Core 1, Prio 2)

- ▶ Log-Events aus Queue verarbeiten
- ▶ Debug-Ausgabe über USB-CDC
- ▶ Non-blocking Queue-Read

## Schichtenmodell

Schicht	Inhalt
app/	FreeRTOS Tasks
logic/	Debounce, Selection
drivers/	CD4021B, HC595
hal/	SPI-Bus + Mutex

## SPI-Konfiguration

- ▶ CD4021B: MODE1, 500 kHz
- ▶ 74HC595: MODE0, 1 MHz
- ▶ Mutex für Thread-Safety

# Serial-Protokoll

## ESP32 → Raspberry Pi

Nachricht	Bedeutung
READY	Controller bereit
FW <version>	Firmware-Version
PRESS 042	Taster 42 gedrückt
RELEASE 042	Taster 42 losgelassen
OK	Befehl erfolgreich
PONG	Antwort auf PING

## Raspberry Pi → ESP32

Befehl	Funktion
LEDSET 042	LED 42 ein (one-hot)
LEDON/LEDOFF	Additiv ein/aus
LEDCLR/LEDALL	Alle aus/ein
PING/STATUS	Test/Zustand
TEST/STOP	LED-Test
VERSION/QRESET	Info/Reset

**Debugging:** STATUS liefert LED-Zustand, Button-Zustand, Heap, Queue-Free, Overflow-Count.

# Raspberry Pi Server

## Technologie-Stack:

- ▶ **aiohttp**: Async HTTP/WebSocket-Server
- ▶ **os.open + stty**: Robuste Serial-Kommunikation
- ▶ **Medien-Validierung**: Prüfung beim Start
- ▶ **systemd**: Autostart als Service

## Datenfluss bei Tastendruck:

1. **ESP32** sendet PRESS 042
2. **Server** sendet LEDSET 042 zurück
3. **Server** broadcastet {"type": "stop"} an alle Browser
4. **Server** broadcastet {"type": "play", "id": 42}
5. **Browser** meldet {"type": "ended", "id": 42} nach Audio-Ende
6. **Server** sendet LEDCLR (nur wenn ID noch aktuell)

**Race-Condition-Schutz:** Ende-Event wird nur verarbeitet, wenn ID noch aktuell.

# Dashboard – Implementierung

## Features v2.5.1

- ▶ **Fullscreen:** Bild füllt Viewport
- ▶ **Preloading:** Alle Medien nach Unlock
- ▶ **WebSocket:** Auto-Reconnect (5s)
- ▶ **iOS-Support:** AudioContext API

## Farbschema

- ▶ **Arduino Teal:** Titel, Akzente
- ▶ **Raspberry Red:** Unlock-Button
- ▶ GitHub Dark: Hintergrund

## Event-Handler (JavaScript)

- ▶ stop: Audio pausieren, currentTime = 0
- ▶ play: Bild + Audio aus Cache abspielen
- ▶ audio.onended: Ende-Event senden

## Medien-Mapping (1-basiert)

- ▶ ID 42 → /media/042.jpg
- ▶ ID 42 → /media/042.mp3

# Getestete Funktionen

## Firmware v2.5.2 (ESP32)

- ▶ Modulare Architektur (HAL/Drivers/Logic/App)
- ▶ SPI-Bus mit Mutex (Thread-safe)
- ▶ Debouncing: 30ms stabil
- ▶ Selection: One-Hot LED-Ansteuerung
- ▶ FreeRTOS Queue (IO → Serial)

## Server v2.5.2 (Raspberry Pi)

- ▶ Serial-Kommunikation (os.open)
- ▶ Robuster Parser für Fragmente
- ▶ WebSocket Broadcast funktioniert

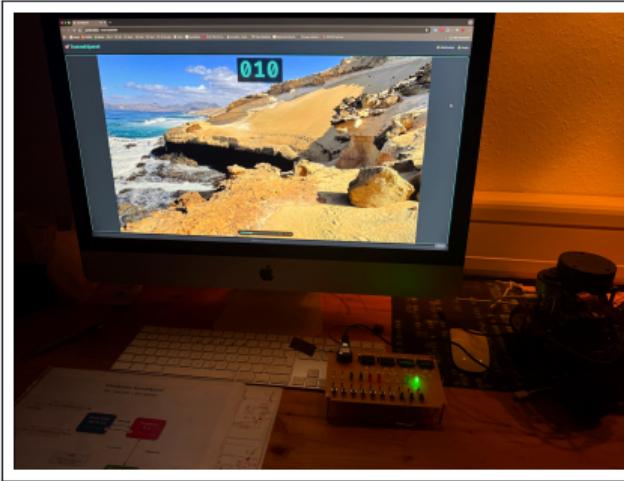
## Dashboard v2.5.1 (Browser)

- ▶ Fullscreen-Bildanzeige
- ▶ Audio mit Preempt-Verhalten
- ▶ Medien-Preloading (instant playback)
- ▶ Multi-Device synchron

## Hardware-Prototyp

- ▶ 10 Taster funktionsfähig
- ▶ 10 LEDs funktionsfähig
- ▶ Bit-Mapping korrekt
- ▶ CD4021B + 74HC595 stabil

# Video-Demo



## Video abspielen

Terminal: `vd`

# Live-Demo

**Dashboard öffnen (alle Geräte):**

```
http://rover:8080/
```

**Tastendruck simulieren (ohne Hardware):**

```
curl http://rover:8080/test/play/5
```

**Preempt testen (während Audio läuft):**

```
curl http://rover:8080/test/play/3
```

**Status abfragen:**

```
curl http://rover:8080/status | jq
```

**Hardware:** 10 Taster drücken → LED leuchtet, Bild + Audio auf allen Geräten synchron.

# Ausblick: Vollausbau 100x

## Änderungen für 100x

- ▶ 13x 74HC595 (statt 2x)
- ▶ 13x CD4021B (statt 2x)
- ▶ PROTOTYPE\_MODE = false
- ▶ Gleiche Firmware/Server



## Stückliste (100x)

Bauteil	Anzahl
74HC595	13x
CD4021B	13x
LED 5mm	100x
Taster	100x
R 330Ω – 3kΩ (LED)	100x
R 10kΩ (Pull-up)	100x
C 100nF	26x

# Zusammenfassung

Phase	Inhalt	Status
1	Infrastruktur & Toolchain	✓
2	Hardware-Prototyp (10x)	✓
3	ESP32 Firmware v2.5.2	✓
4	Raspberry Pi Server v2.5.2	✓
5	Web-Dashboard v2.5.1	✓
6	Hardware-Integration (10x)	✓
7	Raspberry Pi Bridge	✓
8	100-Button + LEDs + Multimedia	◀ Nächste Phase

**Kernbotschaft:** ESP32 für Echtzeit, Pi als Gateway, Browser für UI. Skalierung 10→100 per PROTOTYPE\_MODE Switch.

**Übertragbarkeit:** Voting-Systeme, Quiz-Panels, Museumsinstallationen.

# Credits

Autor

**Jan Unger**

KI-Unterstützung

**Claude 4.5 · ChatGPT 5.2**  
Text- und Code-Vorschläge

Qualitätssicherung

Technische Prüfung und Abnahme durch den Autor

---

Stand: 08.01.2026