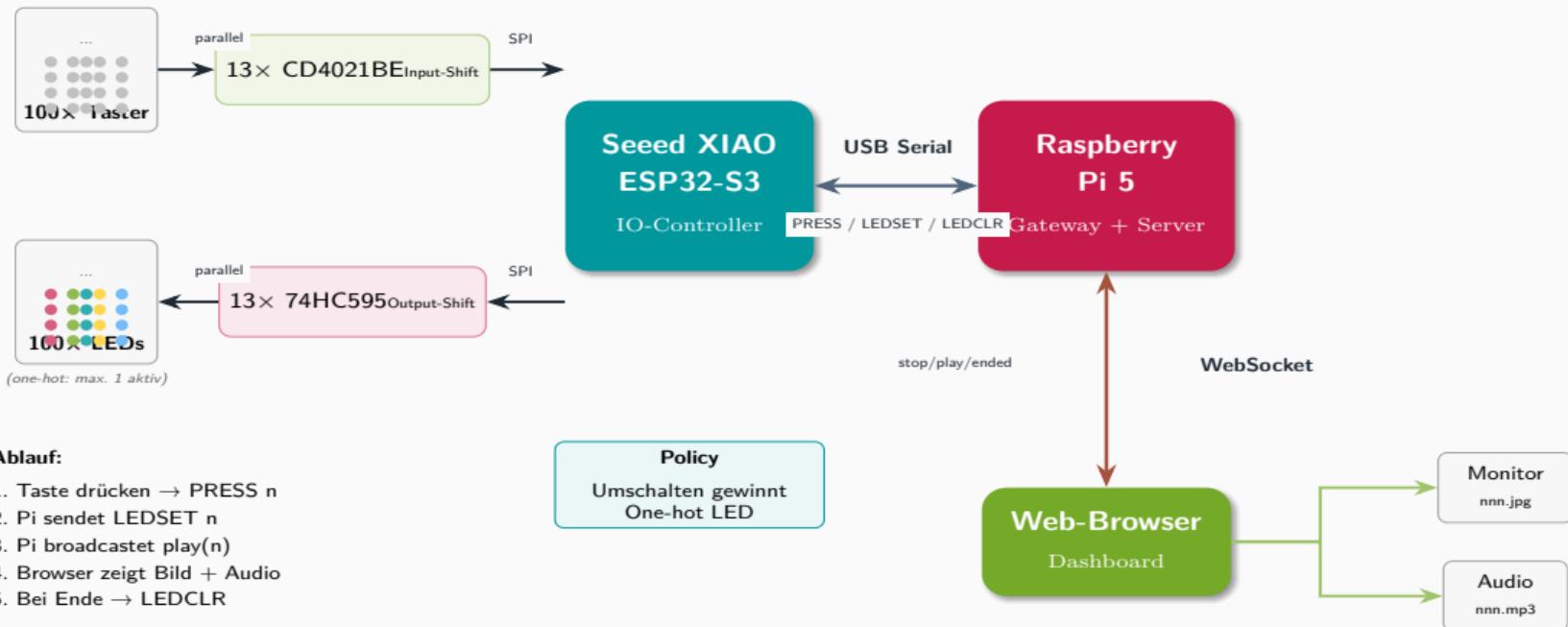


Interaktives Auswahlpanel

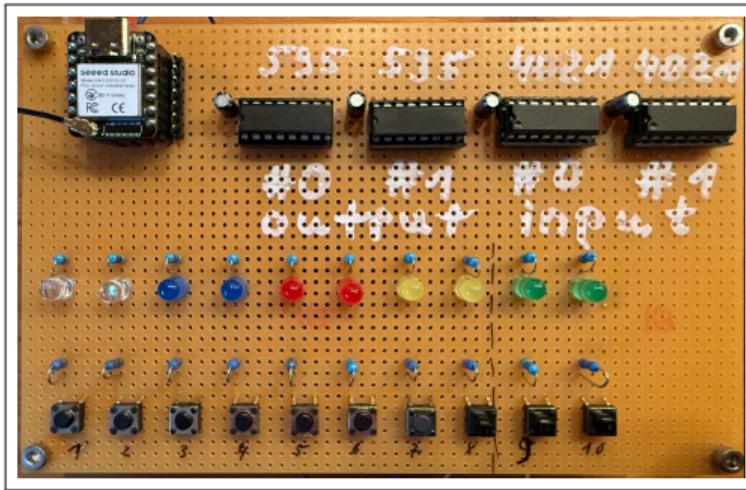
100x Taster/LED + 100x Bild/MP3



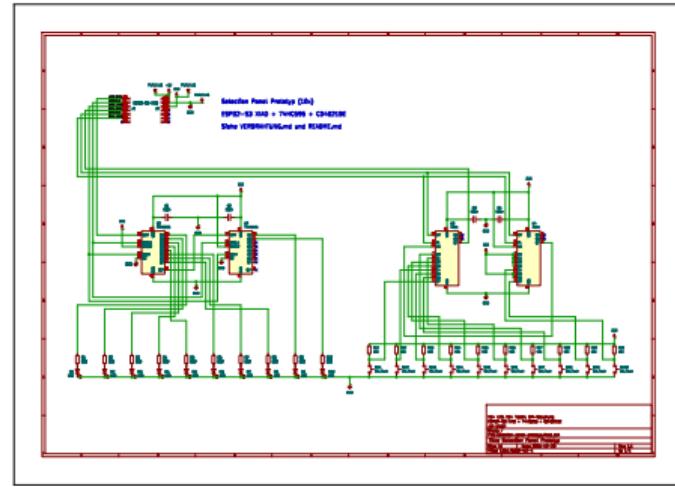
Ablauf:

1. Taste drücken → PRESS n
2. Pi sendet LEDSET n
3. Pi broadcastet play(n)
4. Browser zeigt Bild + Audio
5. Bei Ende → LEDCLR

Hardware-Prototyp v2



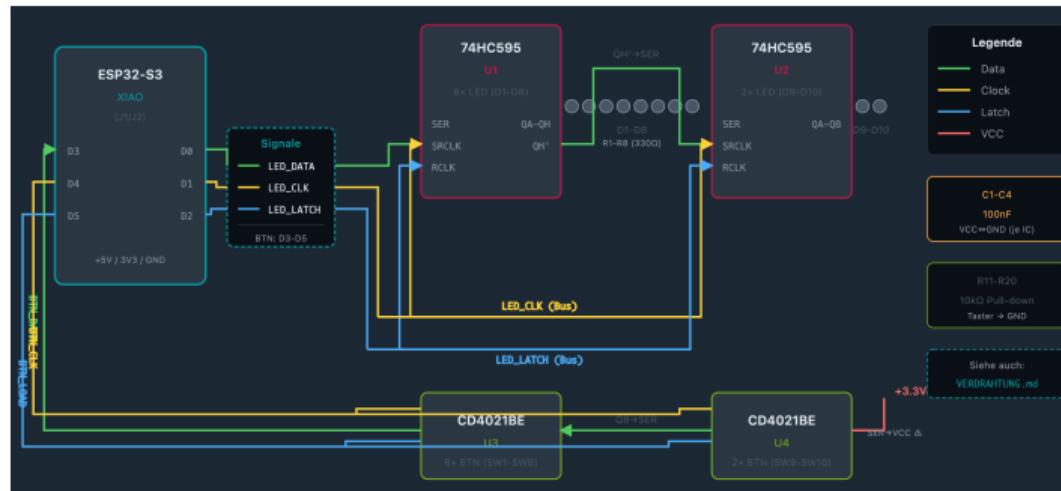
Bestückungsseite – ESP32 + ICs



KiCad – Schaltplan

Komponenten: ESP32-S3 XIAO · 2x 74HC595 · 2x CD4021BE · 10 LEDs · 10 Taster

Schieberegister-Architektur



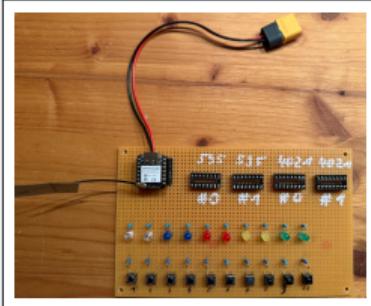
Komponenten

- **ESP32-S3 XIAO** – Mikrocontroller
- **2x 74HC595** – LED-Ausgabe
- **2x CD4021BE** – Taster-Eingabe

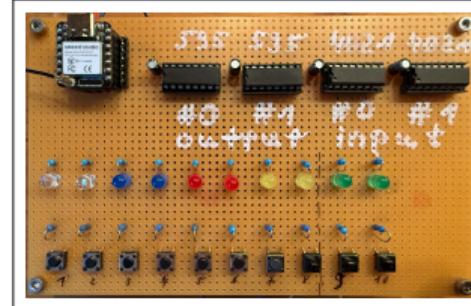
Signale (SPI-ähnlich)

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

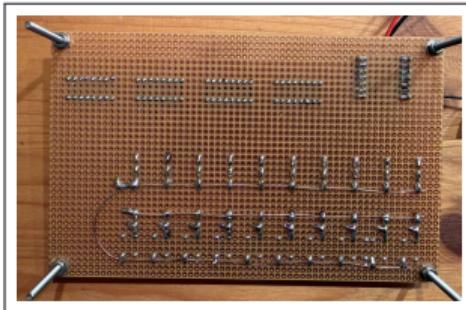
Prototyp-Evolution



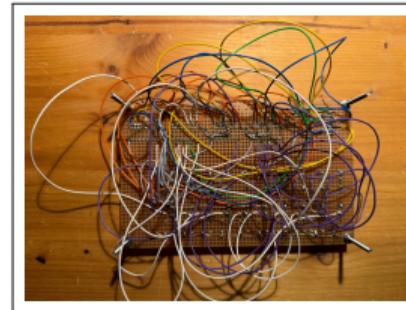
v1 Bestückung – Sockel ohne ICs



v2 Bestückung

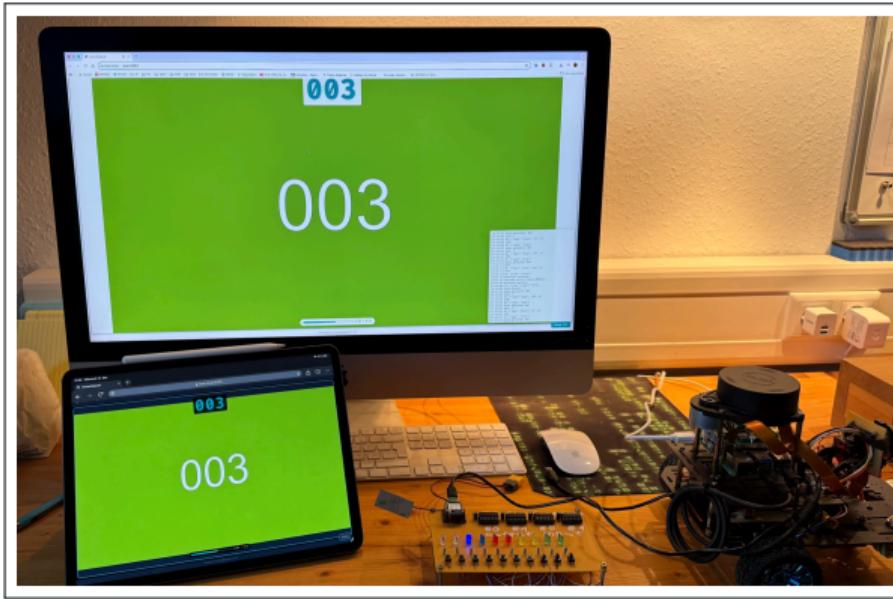


v1 Lötseite



v2 Lötseite

Web-Dashboard



Zugriff

rover.local:8080

Multi-Device

iMac, iPad, iPhone

4K-Medien

Farbschema-Bilder

Synchron

WebSocket Broadcast

Was ist das Auswahlpanel?

Anwendungsfall

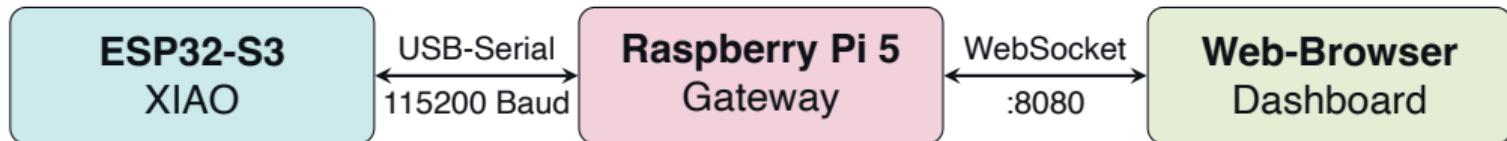
- ▶ 100 physische Taster als Eingabe
- ▶ Jeder Taster triggert ein Bild + Audio
- ▶ LED zeigt aktive Auswahl an

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	CD4021 Q8
RCK	D0	74HC595 Latch
OE	D2	74HC595 Enable (PWM)
P/S	D1	CD4021 Load



CD4021 Timing-Diagramm

Achtung: CD4021 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	Module
app/	io_task, serial_task
logic/	debounce, selection
drivers/	cd4021, hc595
hal/	spi_bus (Mutex)

Dual-Core ESP32-S3

Core	Nutzung	Tasks
Core 0	WiFi/BLE	(reserviert)
Core 1	Anwendung	IO (Prio 5), Serial (Prio 2)

Vorteil: SPI-Bus mit Mutex erlaubt sichere Zugriffe; Queue entkoppelt IO von Logging.

Serial-Protokoll

ESP32 → Raspberry Pi

Nachricht	Bedeutung
READY	Controller bereit
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 v2.2.5

- ▶ **Fullscreen:** Bild füllt Viewport
- ▶ **Overlays:** ID oben, Progress unten
- ▶ **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 laden und abspielen
- ▶ audio.onended: Ende-Event senden

Medien-Mapping (1-basiert)

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

Getestete Funktionen

Firmware v2.5.0 (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 v2.4.1 (Raspberry Pi)

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

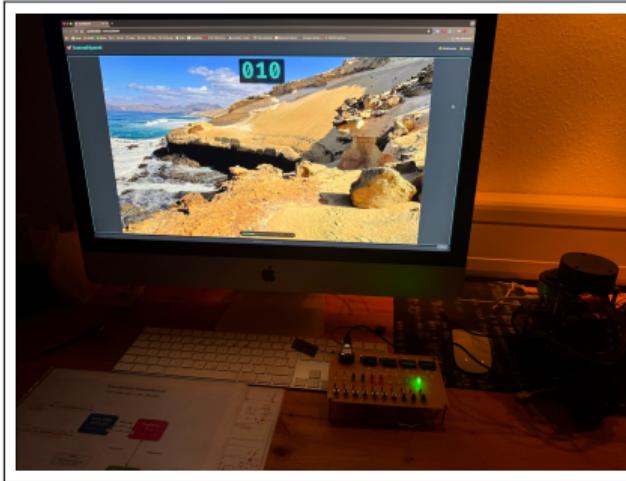
Dashboard v2.2.5 (Browser)

- ▶ Fullscreen-Bildanzeige
- ▶ Audio mit Preempt-Verhalten
- ▶ iOS Audio-Unlock (AudioContext)
- ▶ Multi-Device synchron

Hardware-Prototyp

- ▶ 10 Taster funktionsfähig
- ▶ 10 LEDs funktionsfähig
- ▶ Bit-Mapping korrekt
- ▶ CD4021 + 74HC595 stabil

Video-Demo



vd

Terminal im Hintergrund: Video abspielen

Live-Demo

Dashboard öffnen (alle Geräte):

```
http://rover.local:8080/
```

Tastendruck simulieren (ohne Hardware):

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

Preempt testen (während Audio läuft):

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

Status abfragen:

```
curl http://rover.local: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 CD4021BE (statt 2x)
- ▶ PROTOTYPE_MODE = false
- ▶ Gleiche Firmware/Server

Stückliste (100x)

Bauteil	Anzahl
74HC595	13x
CD4021BE	13x
LED 5mm	100x
Taster	100x
Widerstand 220Ω	100x

Zusammenfassung

Phase	Inhalt	Status
1	Infrastruktur & Toolchain	✓
2	Hardware-Prototyp (10x)	✓
3	ESP32 Firmware v2.5.0	✓
4	Raspberry Pi Server v2.4.1	✓
5	Web-Dashboard v2.2.5	✓
6	Hardware-Integration (10x)	✓
7	Skalierung (100x)	◀ Nächste Phase
8	Produktivbetrieb	□

Kernbotschaft: ESP32 für Echtzeit, Pi als Gateway, Browser für UI. Skalierung 10→100 per PROTOTYPE_MODE Switch.

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