# 📘 Projektstatus: QR-Display-Prototyp

Stand: 8. Oktober 2025

Ziel: ESP32 DevKitC V4 steuert ein 2.4″ Hailege ILI9341-SPI-Display an. Anzeige von Text + QR-Code, Updates via WLAN aus MERN-Backend.

## ⚙️ Hardware-Setup (geprüft & stabil)

|  |  |  |
| --- | --- | --- |
| Komponente | Bezeichnung | Bemerkung |
| Mikrocontroller | ESP32 DevKitC V4 (AZ-Delivery, USB-C unten) | Entwicklungsboard, 3 V3-Logik, 5 V-Versorgung |
| Display | Hailege 2.4″ TFT ILI9341 SPI (SD-Slot oben, Pins nach unten) | 240×320 Pixel, Farb-TFT, 8-polige SPI-Schnittstelle |
| Verbindung | Zwei Breadboards nebeneinander | Gemeinsame Strom-/Masse-Leisten verbunden |
| Versorgung | 5 V USB-Netzteil oder Powerbank (≥ 1 A) | Autarker Betrieb ohne USB-Datenleitung möglich |

## 🔌 Steckplan: Display ↔ ESP32 (geprüft)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Display-Pin | ESP32-Pin | Signal / Funktion | Kabelfarbe | Erklärung |
| VCC | 5 V | Versorgung | Rot | Versorgt Display mit 5 V-Betriebsspannung |
| GND | GND | Masse | Schwarz | Gemeinsamer Massebezug (muss verbunden sein) |
| LED | 3V3 | Hintergrundbeleuchtung | Orange | Steuert LED-Backlight (immer an) |
| SCK | GPIO 18 | SPI Clock | Blau | Taktleitung für SPI-Datenübertragung |
| SDI (MOSI) | GPIO 23 | SPI Data Out | Gelb | Sendet Daten vom ESP zum Display |
| SDO (MISO) | GPIO 19 | SPI Data In (optional) | Grau | Rückdatenleitung, aktuell ungenutzt |
| DC | GPIO 15 | Data/Command Select | Braun | Steuert Modus zwischen Befehlen und Daten |
| RESET | GPIO 4 | Reset | Violett | Setzt Display beim Start sauber neu |
| CS | GPIO 5 | Chip Select | Grün | Aktiviert Display für SPI-Übertragung |

## 💾 Firmware-Umgebung

Pfad: C:\Users\Lenovo\Documents\PlatformIO\Projects\qr\_display\_fw

Framework: Arduino / PlatformIO

Libraries: TFT\_eSPI 2.5.43, ArduinoJson 7.x, QRCode (ricmoo)

## 🧠 Firmware-Funktion (aktuell stabil)

• Initialisiert Display mit hellgrauem Hintergrund (TFT\_SILVER).  
• Zeigt Kopfzeile 'ecily.com – QR PROTOTYP' in dunkelblau/weiß.  
• WLAN-Status, Health-Check (HTTP 200) und Textanzeige 'Jetzt kaufen' (rot).  
• QR-Code mittig, Scale 3, kein Überlappen, kein Flackern.  
• Pollt Backend regelmäßig: /config (JSON {text, qr}) und /health.  
• Automatischer WLAN-Reconnect, stabiler Dauerbetrieb.

## 🔋 Betrieb ohne USB

ESP32 kann direkt über 5 V (bzw. VIN) mit externer Quelle betrieben werden.  
Nach dem Start verbindet er sich selbständig mit dem WLAN und aktualisiert das Display regelmäßig.

## ✅ Aktuell funktionsgeprüft

|  |  |
| --- | --- |
| Funktion | Status |
| SPI-Kommunikation Display | ✅ stabil |
| Farb-Rendering & Fonts | ✅ korrekt |
| WLAN-Verbindung / Auto-Reconnect | ✅ stabil |
| JSON-Abruf vom Backend | ✅ erfolgreich |
| Flackerfreie Darstellung | ✅ |
| Autarker Betrieb über 5 V | ✅ getestet |

## 🔭 Nächster Schritt

• Fertigstellung des MERN-Prototyps (Backend + Frontend)  
• Test: Browser-Eingabe → Display-Update via WLAN  
• Später: Push-Mechanismus (WebSocket/SSE) statt Polling