740 Display-Steuerung Zugzielanzeiger v3.0

von Michael

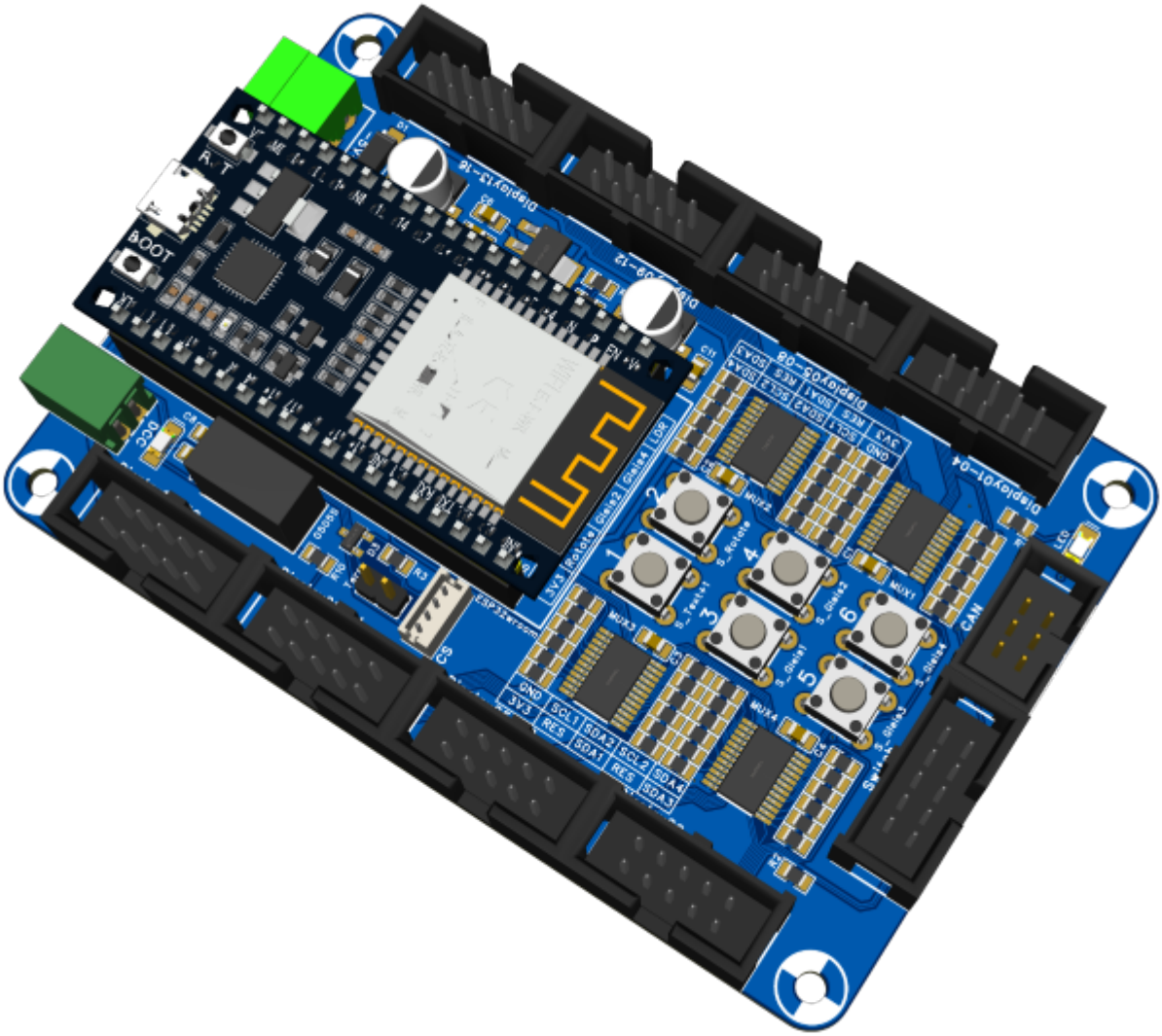
Zur alten Anleitung (v2.1): zur Vorgängerplatine

Hier entsteht die Anleitung zur überarbeiteten Display-Steuerung für Zugzielanzeiger in Version v3.
Die alte Anleitung ist oben verlinkt.
Die Platine ist in Planung und wird erst nach ausgiebigen Tests veröffentlicht.

Alter guten Dinge sind drei

Mit der ZZA Steuerung v3 wird aus den einzelnen Komponenten endlich ein System.

Basierend auf Ideen von Tobias, Klaus, und Fredddy entwickelte Hardi 2019 einen optimierten Sketch, mit dem es erstmals möglich war, 0,87“ und 0,91“ Monochrom-Displays per DCC-Befehl zu aktualisieren.
Mich hat diese Idee vom ersten Tag an so fasziniert, dass ich zwei Jahre intensiv daran gearbeitet habe.



Warum gerade diese Lösung und nicht MQTT?

Ihr könnt diesen Punkt überspringen, wenn ihr direkt zur Anleitung wollt. Für Einsteiger und Umsteiger ist dieser Part aber ggf. von Interesse.

MQTT gilt als moderne, serverbasierte Lösung, bei der das Steuerungsprogramm (z. B. RocRail) die Befehle an einen MQTT Broker sendet (typischerweise ein Raspberry Pi), auf dem die Verwaltung der Zugziele läuft. Sobald der Broker einen Befehl von der Steuerungssoftware bekommt, schickt er die Daten zum ESP.

Die Vorteile von MQTT

- Die Zugziele müssen nicht im Arduino Sketch gepflegt werden. Sie werden bequem über die Weboberfläche WYSIWYG eingepflegt.
- Deutsche Umlaute können direkt eingegeben werden.
- Es ist nur ein einziger Befehl der Steuerungssoftware an den MQTT Broker nötig, um das Zugziel an das richtige Gleis zu senden.

Die Nachteile von MQTT

- Der Installationsaufwand stellt für den ungeübten Bastler mit wenig IT-Kenntnissen eine

zusätzliche Hürde dar.
- Das Umschalten der Displays setzt zwingend eine Steuerungssoftware (z. B. RocRail) voraus oder einen PC in unmittelbarer Nähe, zum Schalten per Weboberfläche.

Als ich mein Projekt im Dezember 2023 startete, war RocRail das einzige Programm, das MQTT unterstützte. Ich selbst nutze iTrain, welches MQTT bis heute nicht unterstützt, ebenso wenig wie TrainController. WinDigipet unterstützt es erst seit 2025. Es ist davon auszugehen, dass MQTT in naher Zukunft von allen vier genannten Steuerungen unterstützt wird. Die im Stummiforum vorgestellte Lösung läuft hingegen komplett auf dem Arduino und wird über die serielle Schnittstelle oder per DCC Befehl gesteuert.

Die Vorteile des Sketchs

- Steuerbar per serieller Schnittstelle, DCC Handregler und Steuerungssoftware.
- Kein zusätzlicher Server nötig.
- Steuerbar mit iTrain (Aktionen), RocRail (ergänzen?), TrainController Gold (Bahnwärter) und WinDigipet (Stellwerkswärter)

Die Nachteile des Sketchs

- Aufwändig zu pflegende Zugziele direkt im Programmcode (große Hürde)
- Deutsche Umlaute müssen in Octal Codes eingeben werden.
- Maximal 100 Ziele im Flash des Arduinos speicherbar.
- Eine Anpassung bzw. Ergänzung der Zugziele setzt zwingend eine USB Verbindung zum Arduino voraus.
- Es sind immer zwei direkt aufeinanderfolgende DCC-Befehle nötig, um das richtige Gleis zu wählen und den passenden Text zu senden. Das können nicht alle Steuerungen (siehe oben)

Obwohl die Nachteile dieser Lösung überwogen, versuchte ich in Ermangelung einer MQTT Unterstützung durch iTrain daher, die Nachteile der Arduino Lösung zu kompensieren. Leider entpuppte sich der Arduino Nano als Sackgasse. Trotz zahlreicher Versuchsaufbauten und passend entwickelter Platinen (v1 und v2), konnte der kleine Mikrocontroller die Zugzielanzeiger nicht stabil und zuverlässig ansteuern. Ursache waren letztendlich die als Slave oder Follower bezeichneten, gespiegelten Displays. Diese Erkenntnis hätte das gesamte Konzept beinahe zum Erliegen gebracht, wenn da nicht ein letzter Strohhalm gewesen wäre: der ESP32

Nach anfänglichen Startschwierigkeiten und ernüchternder Erkenntnisse ebnete der ESP32 letztendlich den Weg zum Durchbruch. Mit den hier erlangten Erfahrungen hätte man das Projekt auch auf einem Arduino Nano umsetzen können, doch Hardi und ich waren bereits zu weit fortgeschritten, um die beiden Hauptvorteile aufzugeben: Den auf dem zweiten Prozessorkern ausgelagerten DCC Prozess und die WLAN Antenne des ESP32.

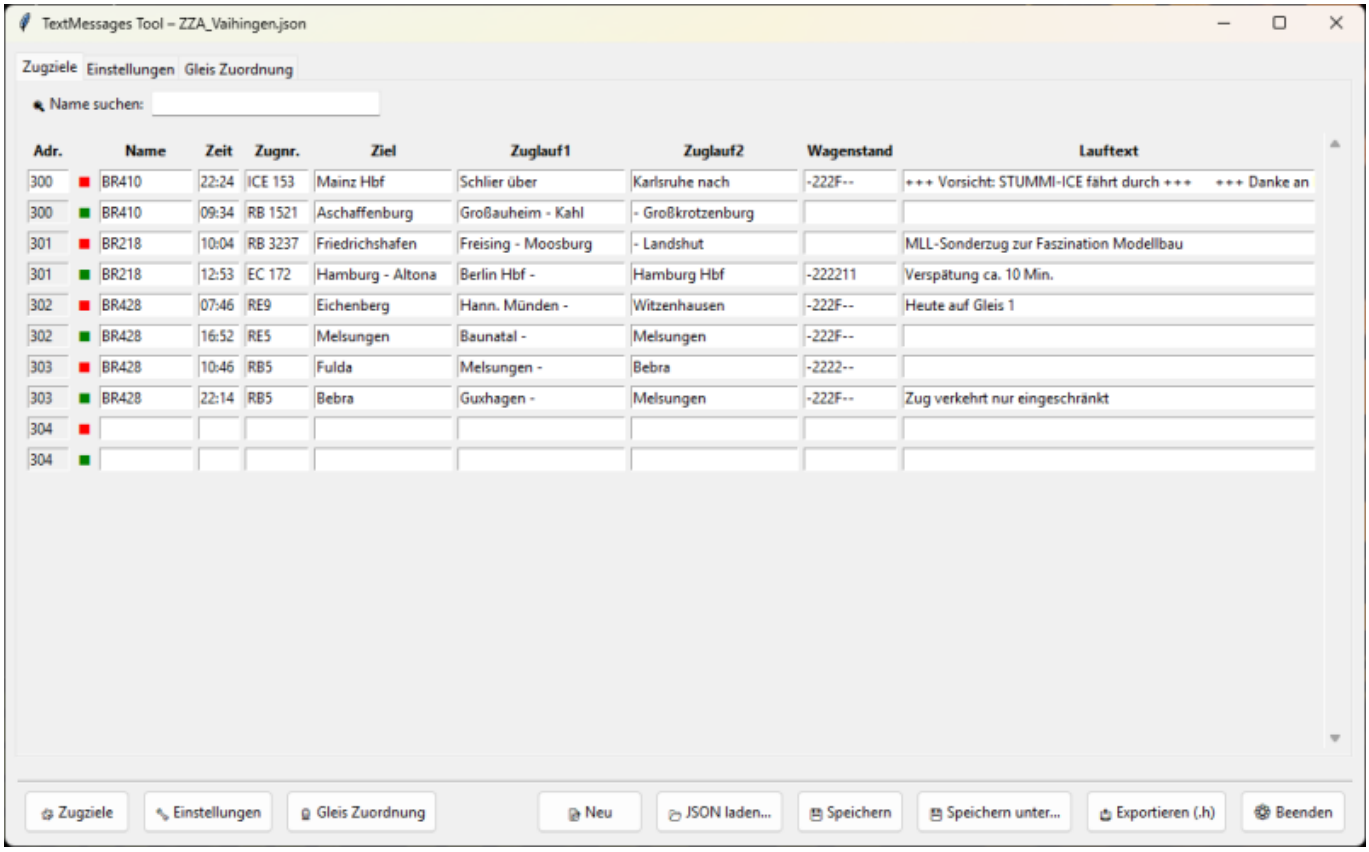
So implementierte Hardi den Page Modus, eine Funktion, bei der nur ein Viertel des Displays aktualisiert werden muss, um den Lauftext in einem Viertel der Zeit zu erzeugen und eine Funktion zur Bildung von Display-Gruppen, wodurch mehrere Displays mit nur einem DCC Befehl dasselbe Zugziel empfangen, egal, ob links- oder rechtsseitige Ausrichtung. Neben den üblichen Anpassungen an die neue Hardware (ESP32) spendiert er dem Sketch knapp 250 vordefinierte Zugziele, um den händigen Aufwand zu minimieren. Eine kurze Recherche ergab dann, dass ich mit knapp einem Dutzend Zeilen Code das WLAN zum Übertragen des Sketchs aktivieren konnte.

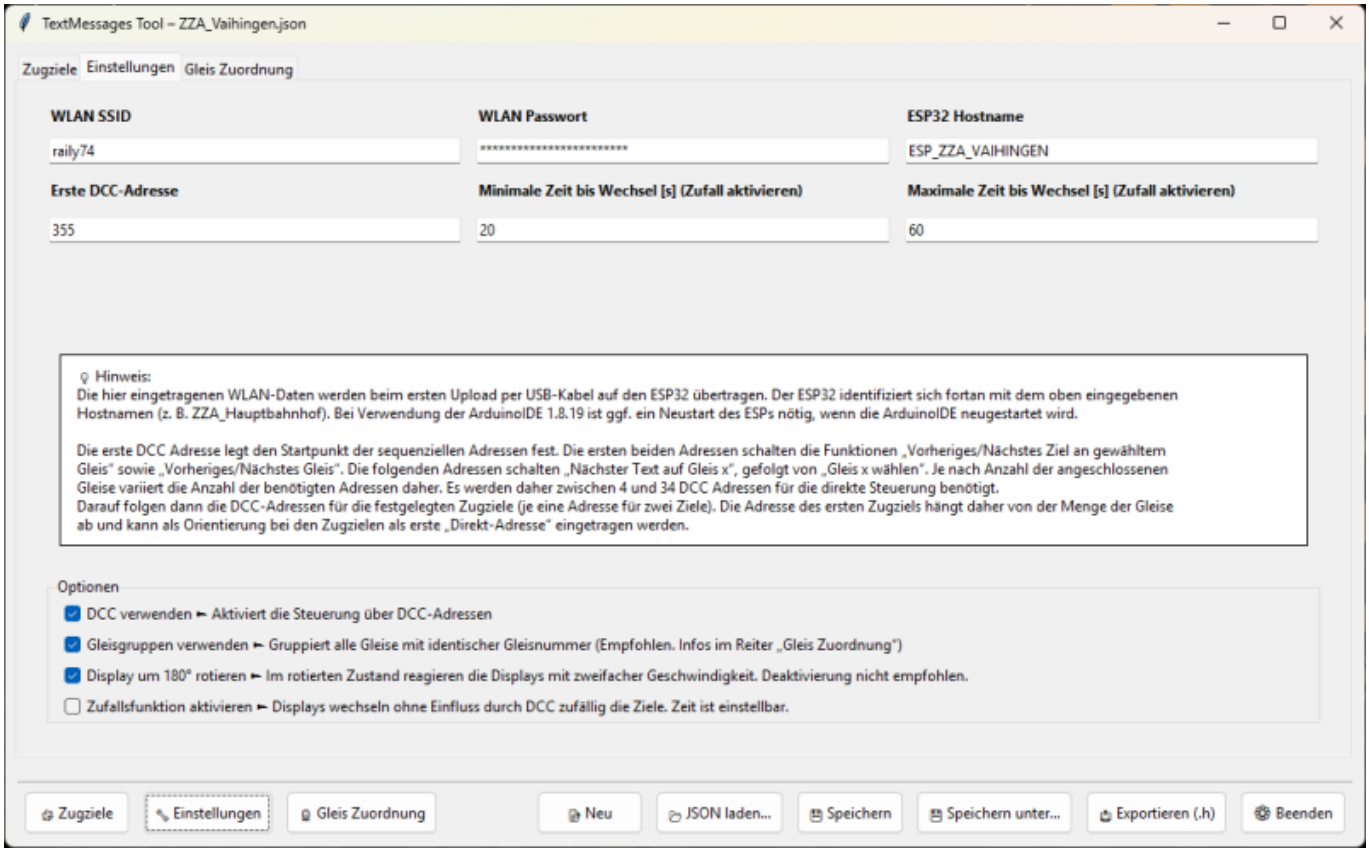
Nun blieben nur noch zwei Nachteile: Die überhaupt nicht intuitive Eingabe der Texte und Einstellungen sowie die zwei aufeinanderfolgenden DCC Befehle.

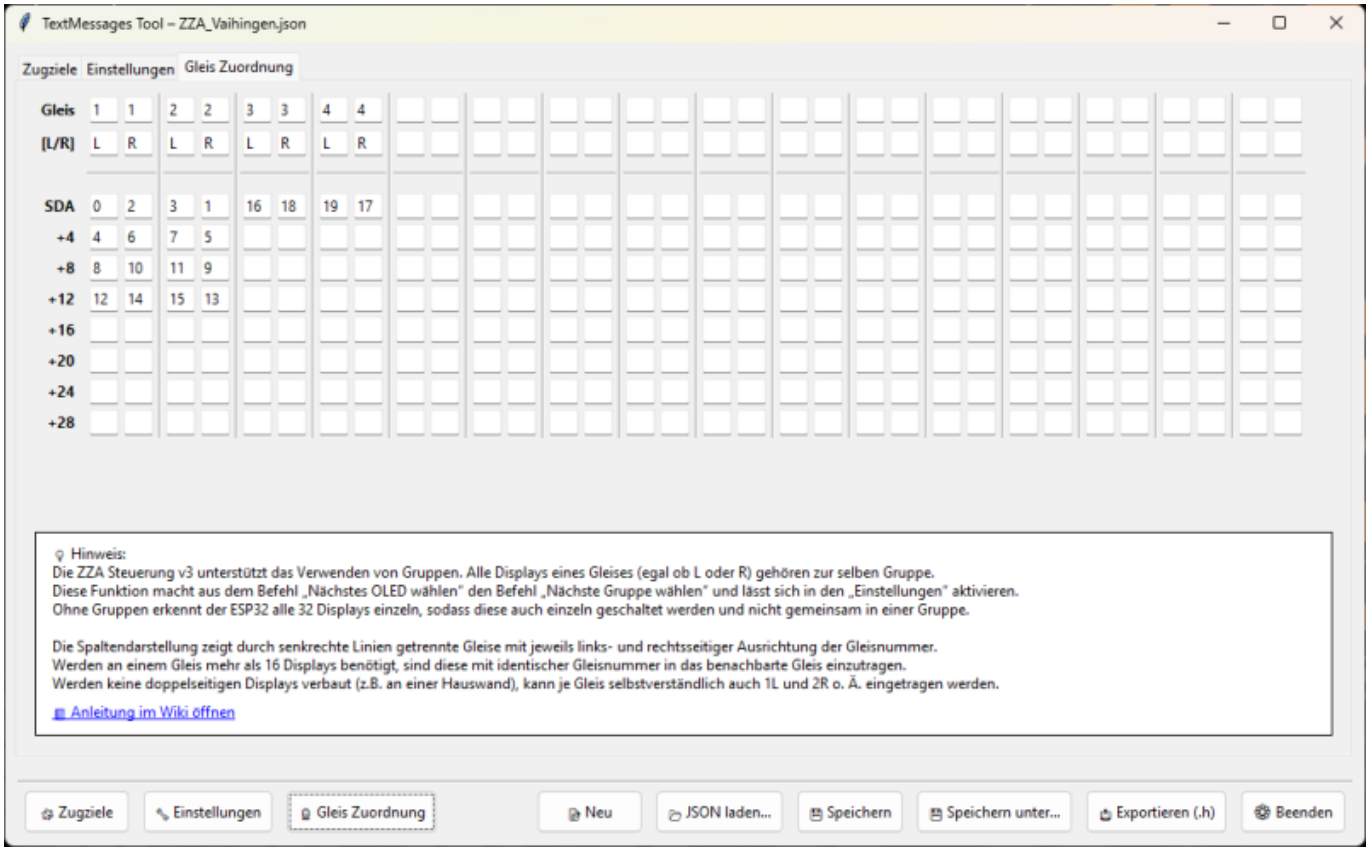
Die Eingabe der Daten sowie die Umwandlung in den Sketch konnte ich mit einem selbst erstellten Programm abfangen. Dieses macht die Eingabe genauso komfortabel wie die Weboberfläche eines MQTT Servers, bietet dabei aber die Vorteile der Arduino Lösung.

Für das versetzte Senden zweier DCC Befehle gibt es Workarounds, iTrain beherrscht es ab Werk. Es bleiben also keine wirklichen Nachteile mehr.

Das TextMessages Tool







Pin-Belegung des Wannensteckers