740 Display-Steuerung Zugzielanzeiger v3.0

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.

Vorgeschichte

Die Zugzielanzeiger sind ein eigenständiges Projekt neben der MobaLedLib.
Von der Idee bis zur heutigen Umsetzung vergingen zehn Jahre. Aus dem ursprünglichen Thread entwickelten sich auch andere Lösungen, wie beispielsweise das RocMQTTdisplay von Chrisweather, welches aber zwingend die Kommunikation per MQTT voraussetzt (z. B. mit Rocrail).

12-2015	TobiBS stellt seine Eigenentwicklung erstmalig vor.
02-2019	Hardi wird auf das Thema aufmerksam und entwickelt gemeinsam mit Fred und Tobias den vorhandenen Code zu einem optimierten Sketch weiter.
08-2019	Passend zum Sketch entwickelt Hardi die 760 Kofferplatine zur Aufnahme der winzigen 0,87“-Displays sowie einen passenden Schaltplan zur Steuerung der Kofferplatinen.
02-2024	Michael nimmt sich der Platine für die Steuerung an und kombiniert zwei Arduinos zur Ansteuerung von 24 Displays. Die ersten Tests verlaufen positiv. Trotz aller Bemühungen kommt es in bestimmten Einbausituationen zu Störungen auf den Slave-Displays. Benutzer berichten von Störungen durch elektrische Fußbodenheizungen, zu lange Kabelwege oder zu dünne Kabel. Es kommt zu Abstürzen. Wird die Schaltung nur mit Master-Displays (also max. acht) betrieben, treten keine Fehler auf.
08-2025	Der Durchbruch! Hardi und Michael starten einen letzten Versuch. **Ein** ESP32 soll die Steuerung aller Displays eines Bahnhofs übernehmen! Die Anpassungen im Sketch übernimmt Hardi in nächtelanger Arbeit. Nach knapp einer Woche laufen die Displays auf dem Steckbrett. Michael baut eine neue Platine auf, die sich komplett von Master und Slave-Anschlüssen befreit und auf einen ESP32 setzt.

Die Highlights im Überblick

- Nur ein ESP32 steuert 32 OLEDs
- Bis zu 20 Displays mit Lauftext
- 32 Displays mit Lauftext bei verringerter Geschwindigkeit möglich
- Über 200 Ziele vordefiniert
- Anzahl der Gleise frei wählbar
  (z. B. 4 Gleise à 8 Displays oder 16 Gleise à 2 Displays)
- Unterschiedliche Anzahl an Displays je Gleis möglich (z. B. 8 Displays an Gleis 1 und je 4

Displays an Gleis 2-7)
- Ansteuerung per DCC oder per CAN
- Anschluss eines LDR für weitere Spielereien am Sketch vorbereitet
- Platine mit optionalen Testanschlüssen (abtrennbar)

Funktionsweise der Display-Steuerung

Der absolute Clou der Neuauflage ist die Art und Weise, wie der überarbeitete Sketch die Displays aktualisiert. Ziel war es, sich von den Slave Displays zu befreien. Alle angeschlossenen Displays sollten als Master angeschlossen werden, um die Kommunikation auf dem I²C-Bus nicht zu stören. Die größte Hürde war dabei die Zeit, die ein Arduino oder auch ein ESP32 benötigt, um die 4.096 Pixel des OLED-Displays neu zu „zeichnen“. Dafür benötigen beide Prozessoren 20ms. Wenn die Aktualisierung jedes einzelnen OLEDs jedoch länger als 100ms dauert, läuft der Lauftext nicht mehr flüssig.
Das bedeutet, dass beide Prozessoren spätestens nach dem fünften Display wieder das erste neu zeichnen müssen, damit der Lauftext in Bewegung bleibt. Doch hier kommt der neue Sketch ins Spiel. Die verwendeten OLED-Displays sind intern in vier Bereiche (Pages) unterteilt, die einzeln aktualisiert werden können. Nach vielen Versuchen gelang es uns, diesen Bereich separat anzusprechen.
Nun müssen nur noch 1.024 Pixel neu gezeichnet werden, was ein ESP32 ich knapp 5ms erledigt. Erst wenn das OLED ein anderes Ziel anzeigen soll, werden alle 4.096 Pixel neu gezeichnet.

Umschaltung

Als klar wurde, dass der ESP32 ohne Probleme **20 OLEDs mit Lauftext** betreiben kann, war die Lösung über MOSFETs hinfällig. Man hätte dafür 20 freie Pins oder einen zusätzlichen IC zur Umschaltung der MOSFETs benötigt. Daher entschieden wir uns für vier Multiplexer, die selbst auch über den I²C-Bus umgeschaltet werden. An diesen vier Multiplexern lassen sich 32 OLEDs betreiben, die natürlich alle mit Lauftext funktionieren würden, allerdings nur mit einer Refresh-Rate von 160ms. Wem das zu langsam erscheint, kann bei mehr als 20 OLEDs auf eine zweite Steuerung „upgraden“.

Kofferplatine oder OLED-Adapter?

Da die neue Schaltung keine Slaves mehr ansteuert, sondern nur noch Master und sich jeweils zwei SDA-Leitungen eine SCL-Leitung teilen, funktioniert das Ganze sowohl mit den Kofferplatinen von Hardi als auch mit den OLED-Adaptern von Michael, selbstverständlich mit vorder- und rückseitigem Lauftext.

Stückliste

Nachfolgend findet man die Stückliste der notwendigen Bauteile.

Die Stückliste ist für die voll bestückte Variante ausgelegt. Möglicherweise entfallende Bauteile oder andere Wannenstecker sind nicht berücksichtigt.

Anzahl	Bezeichnung	Beschreibung	Bestellnummer	vorbestückt	Alternativen, Bemerkungen
1	Board	Platine	740-Display-Steuerung		
4	TCA9548APWR	I²C-Multiplexer, SMD		ja	
2	D1, D2	Schottkydiode, 40 V, 1 A, DO-214AC/SMA	B 140 F	ja	
1	D3	PESD2CAN		ja	nur für CAN
1	R1	SMD-Widerstand, 0805, 470 Ohm, 250 mW		ja	
1	R2	SMD-Widerstand, 0805, 4,7 kOhm, 250 mW		ja	
1	R3	SMD-Widerstand, 0805, 120 Ohm, 250 mW		ja	nur für CAN
1	R4	SMD-Widerstand, 0805, 150 Ohm, 250 mW		ja	
58	R5-R76	SMD-Widerstand, 0805, 1 kOhm, 250 mW		ja	
8	C1-C6, C9, C11	Vielschicht-Kerko, 0805, 100nF, 50V		ja	
2	C7, C8	Vielschicht-Kerko, 0805, 4,7uF, 50V		ja	nur für CAN
1	C10	Vielschicht-Kerko, 0805, 1uF, 50V		ja	
2	C12, C13	SMD Elko, 100 uF, 16 V		ja	
1	AMS1117-3.3	Spannungsregler, 3,3 Vout		ja	
8	LED1	LED 3mm, bedrahtet, grün		nein	
1	6N137	Optokoppler	6N 137	nein	
1	Fassung für 6N137	IC-Sockel, 8-polig	GS 8	nein	
1	Fassung für ISO1050	IC-Sockel, 8-polig	GS 8	nein	nur für CAN
2	-5V+	Anschlussklemme, 2-pol, RM 3,5 mm, 90°	CTB932HD-2	nein	
1	0505S	B0505S-W5R3		nein	nur für CAN
2	DCC	Anschlussklemme, 2-pol, RM 2,54mm		nein	
8	Display	Wannenstecker, 10-polig	WSL 10G	nein	
1	Switch	Wannenstecker, 10-polig	WSL 10G	nein	
1	CAN	Wannenstecker, 6-polig		nein	nur für CAN
1	CS	B4B-ZR-3.4		nein	nur für CAN

Anzahl	Bezeichnung	Beschreibung	Bestellnummer	vorbestückt	Alternativen, Bemerkungen
6	S_Text+1, S_Rotate, S_GleisX	Kurzhubtaster 6x6mm, Höhe: 20,0mm	Schalter Dip 6x6x20	**nein**	
4	A1, A2	Buchsenleiste, 19-pol		**nein**	**Die Buchsenleiste muss geteilt werden.**
2	ESP32wroom	DevKit ESP32 38 Pins		**nein**	

Buchsenleiste teilen

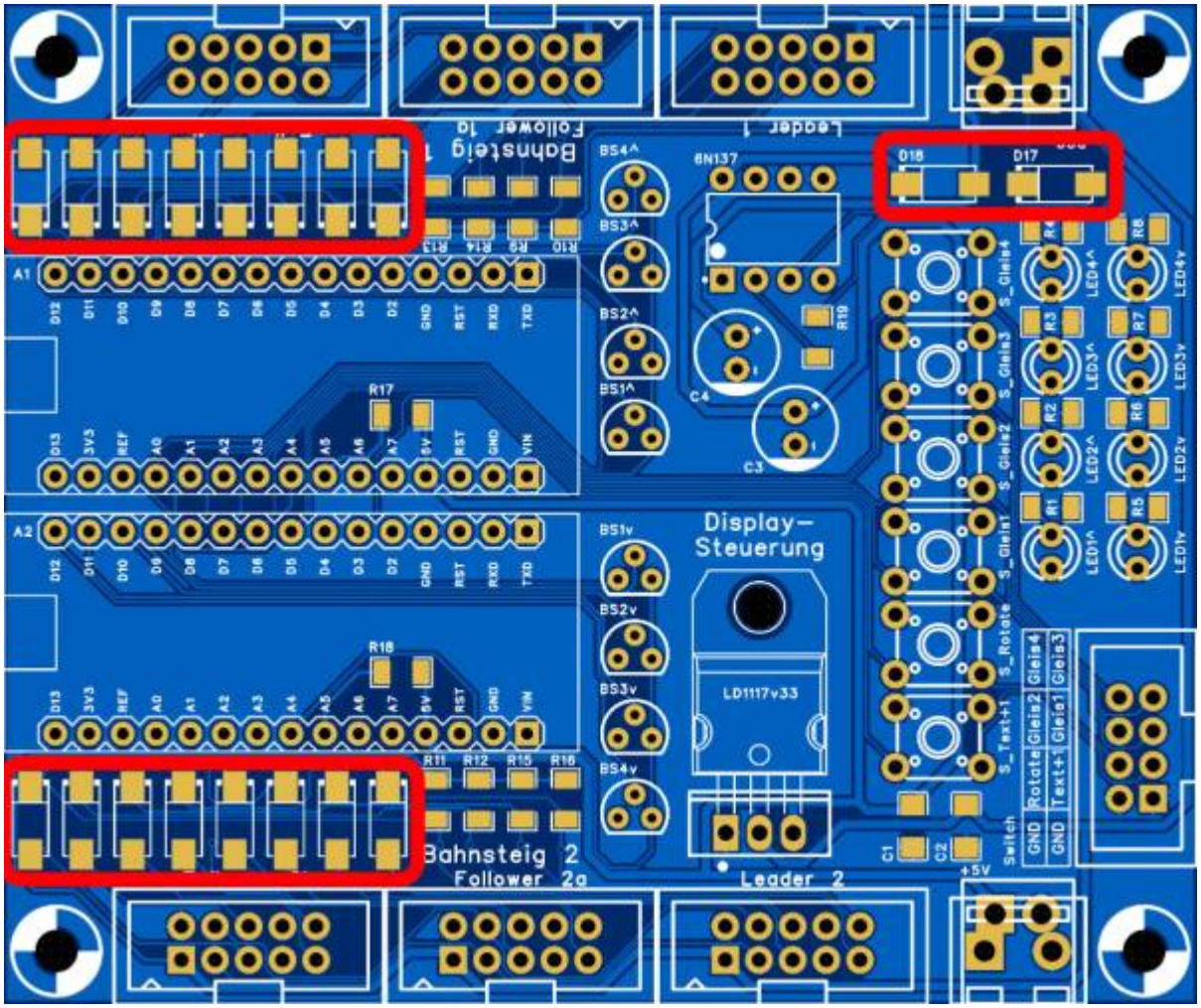
Die 40poligen Buchsenleiste für A1 und A2 wird per Säge auf die notwendigen Teilstücke abgelängt (jeweils etwa 1mm hinter dem letzten benötigten Bein absägen). Aus einer 20poligen Leiste wird eine 15-polige Buchsenleisten erstellt.

Bestückung der Display-Steuerung

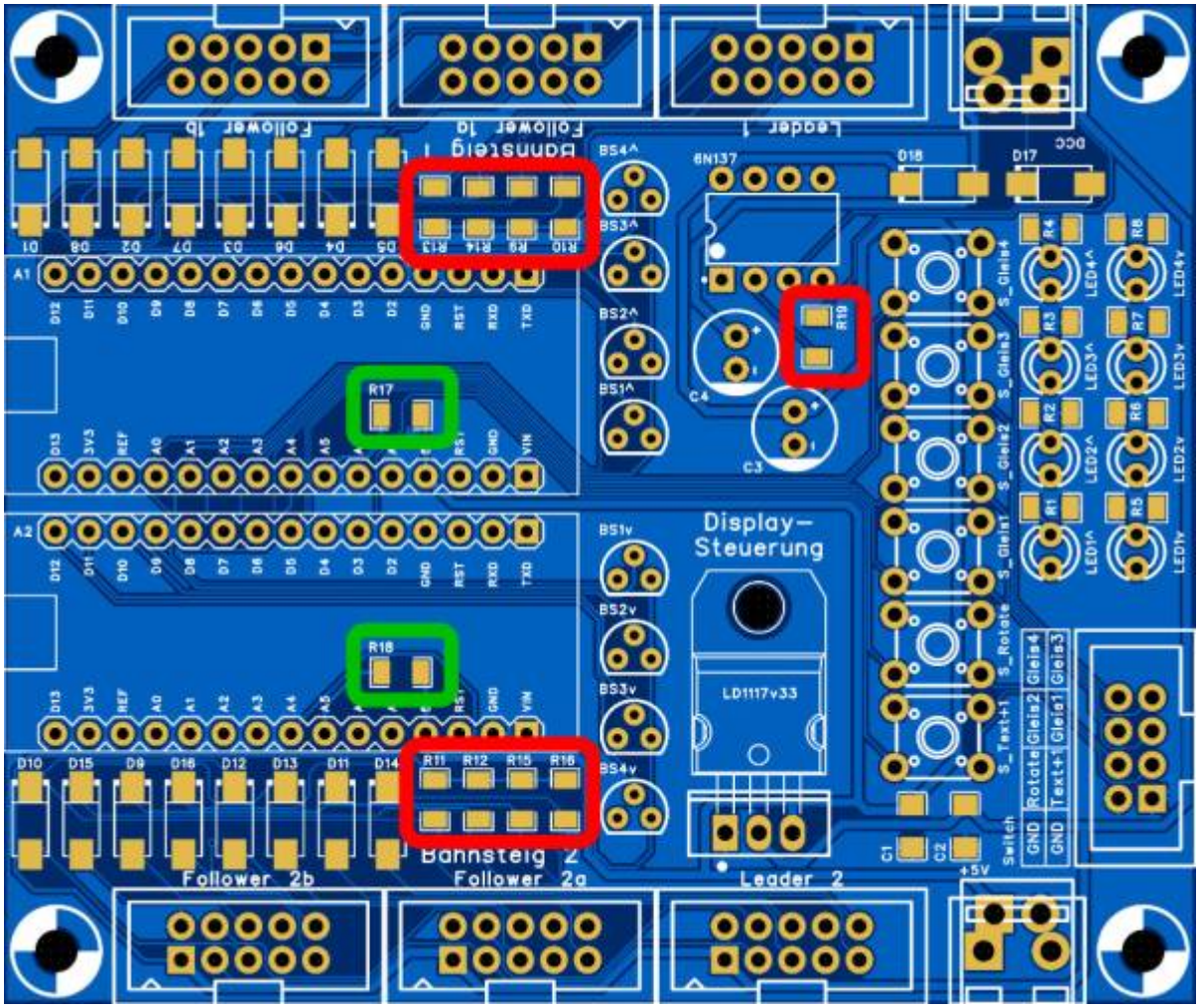
Es empfiehlt sich, folgende Reihenfolge bei der Bestückung einzuhalten:

Sofern nicht die SMD-vorbestückte Platine erworben wurde, müssen zunächst die SMD-Bauteile aufgelötet werden.

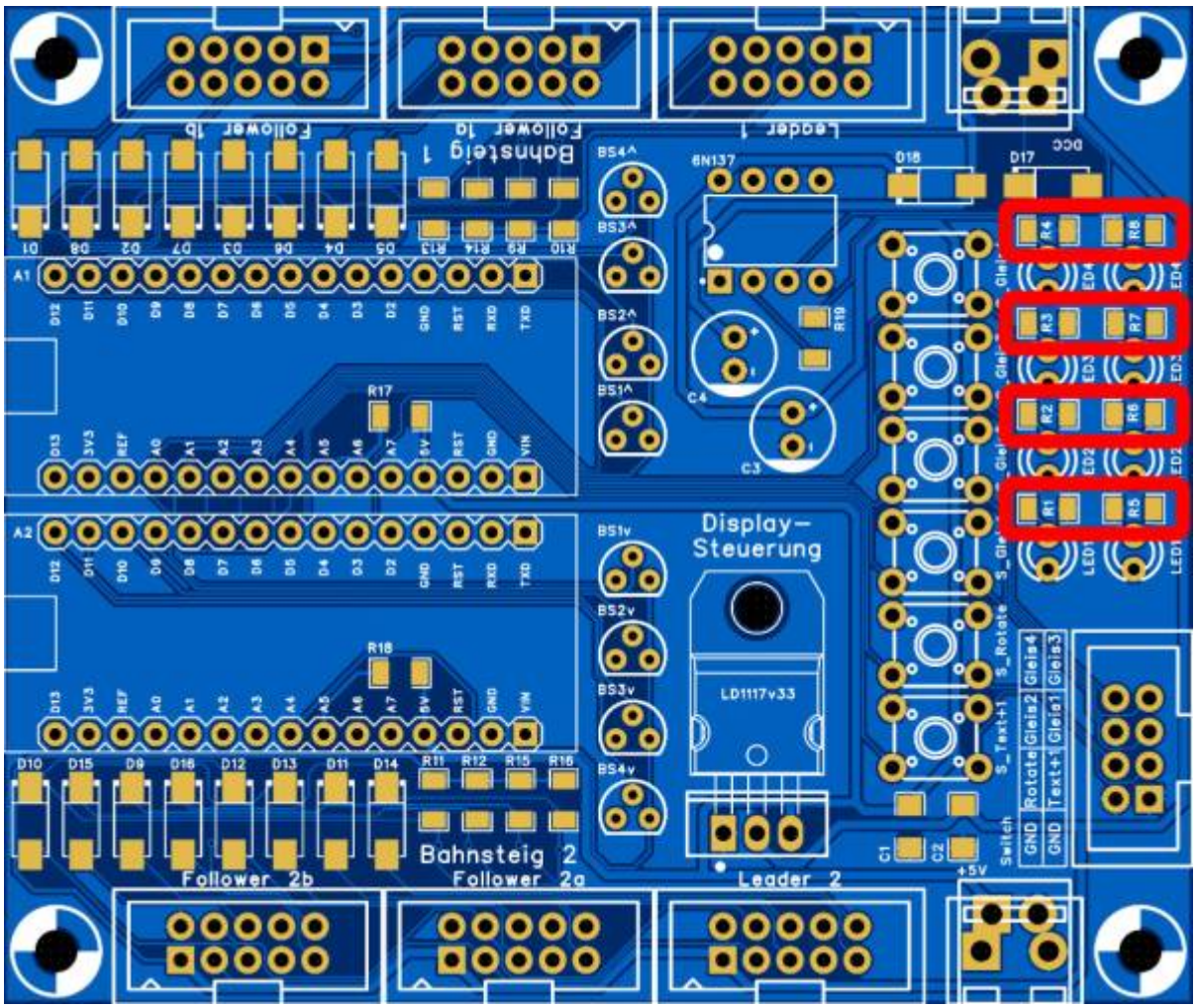
1) Dioden D1 bis D18



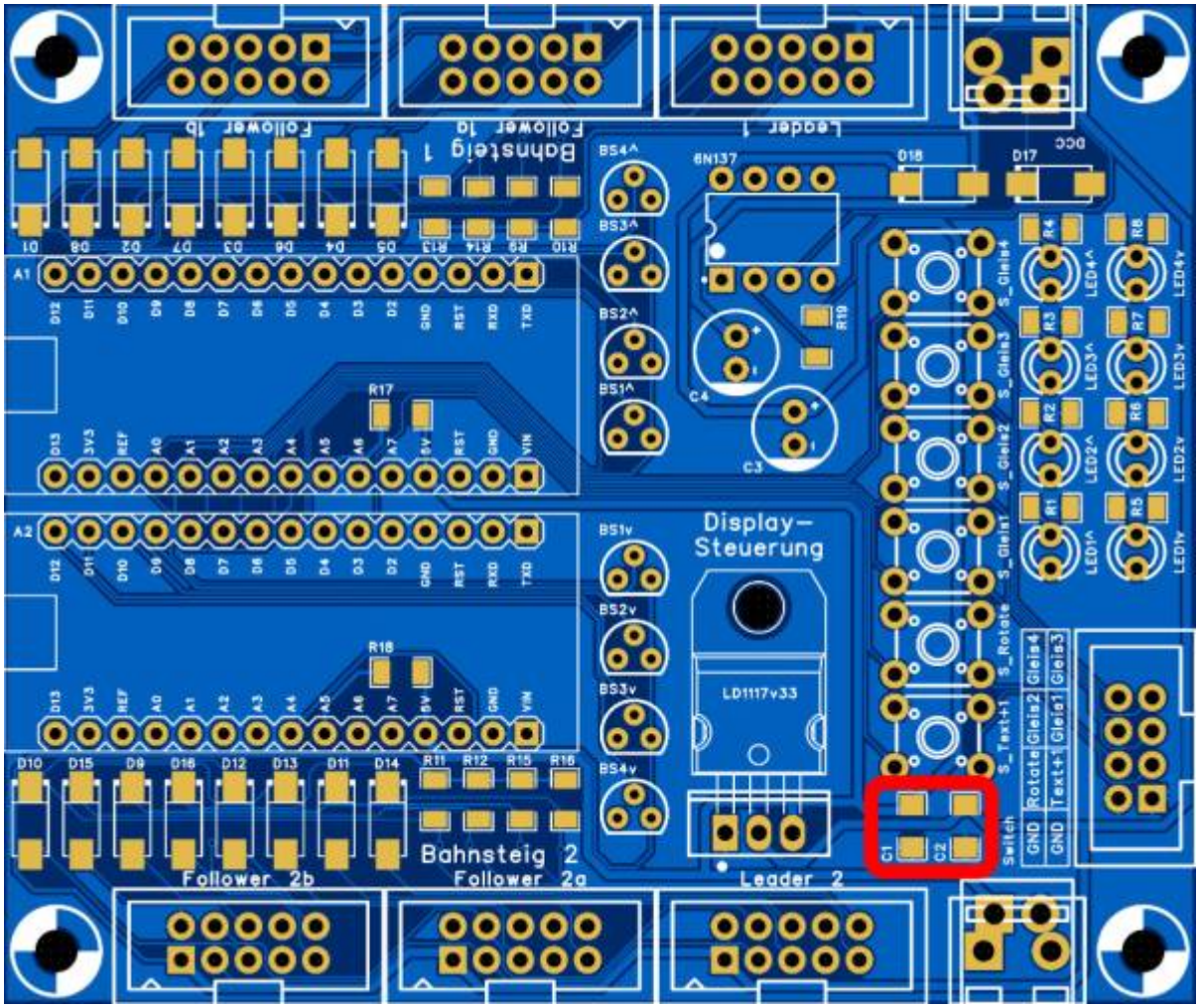
2) Widerstände R9 bis R16 und R19 (rot) sowie Widerstände R17 und R18 (grün)



3) Widerstände R1 bis R8

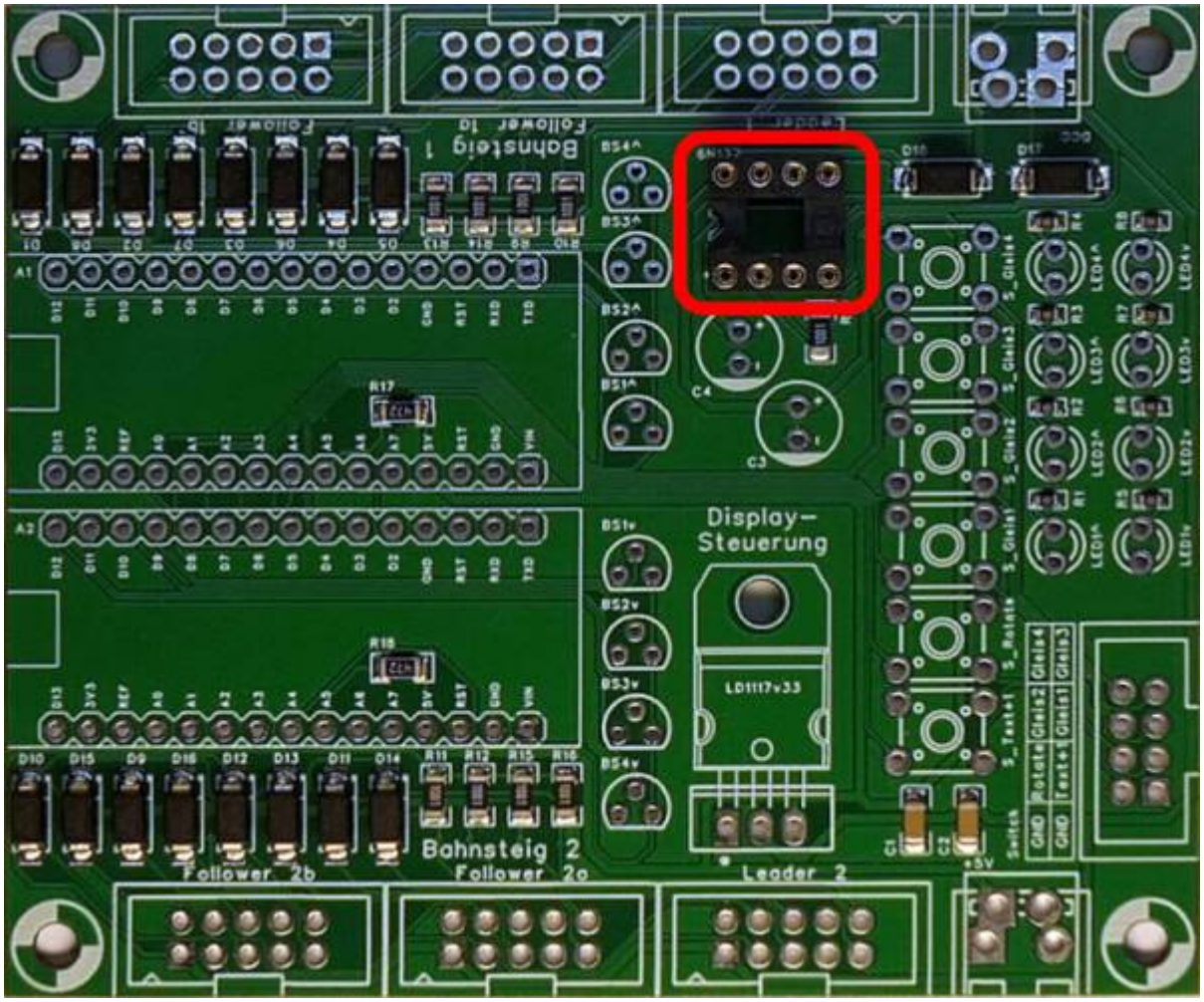


4) Keramikkondensatoren C1 und C2

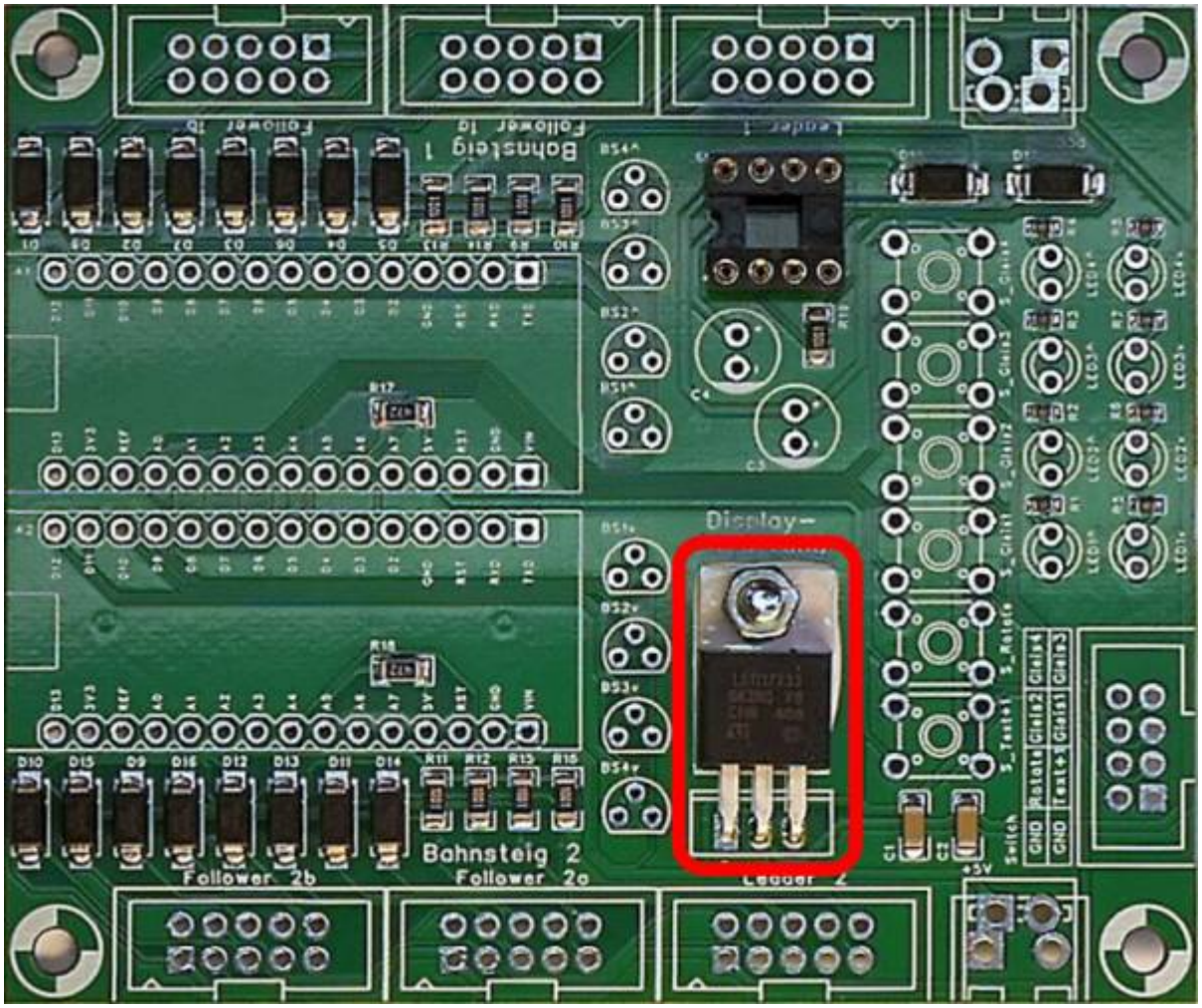


Nun geht es mit der Bestückung der THT-Bauteile weiter:

5) IC-Fassung für Optokoppler 6N137

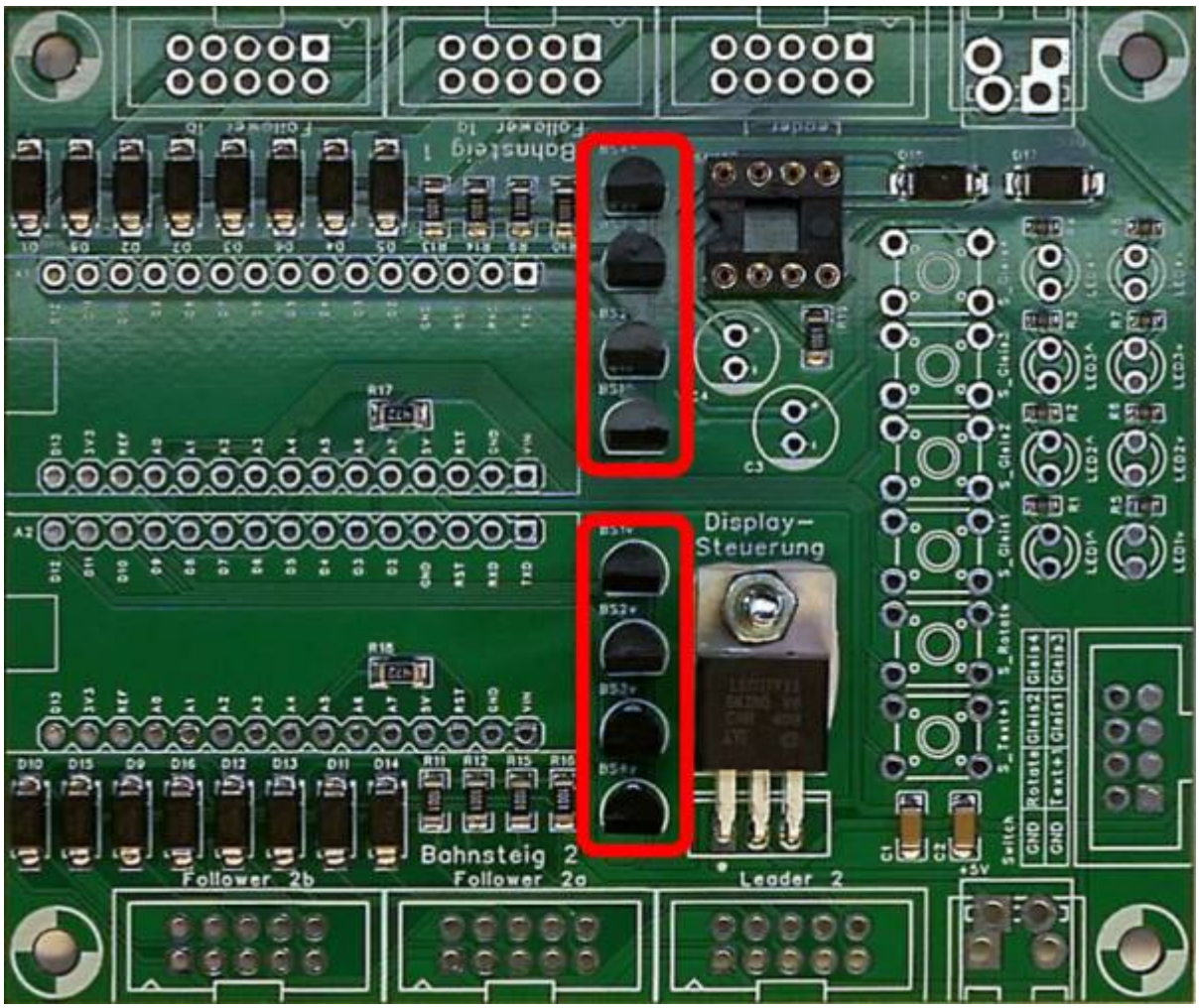


6) Spannungsregler LD1117v33 mit Wärmeleitpaste, Schraube M3x5 und Mutter M3 anbringen, dann löten

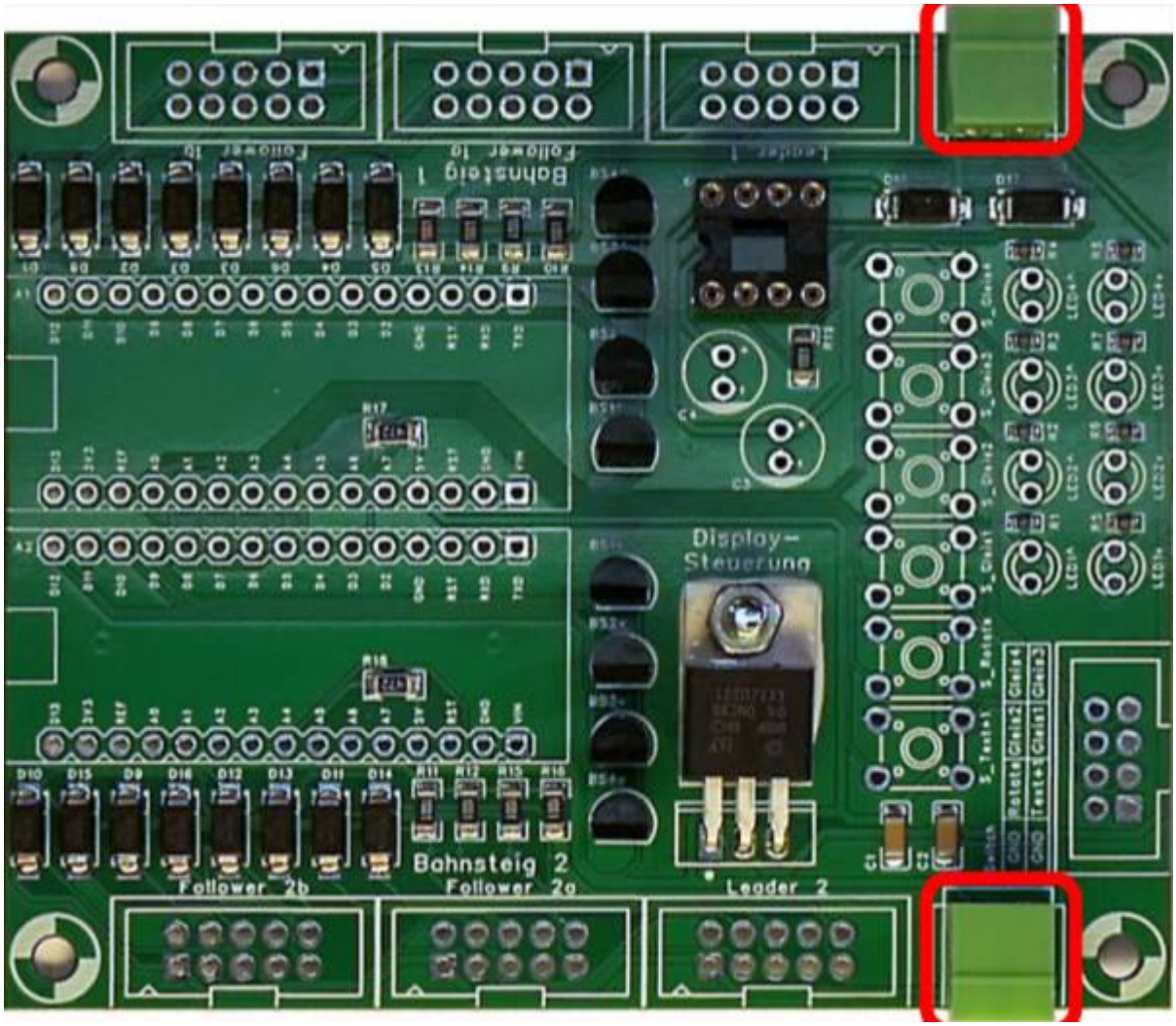


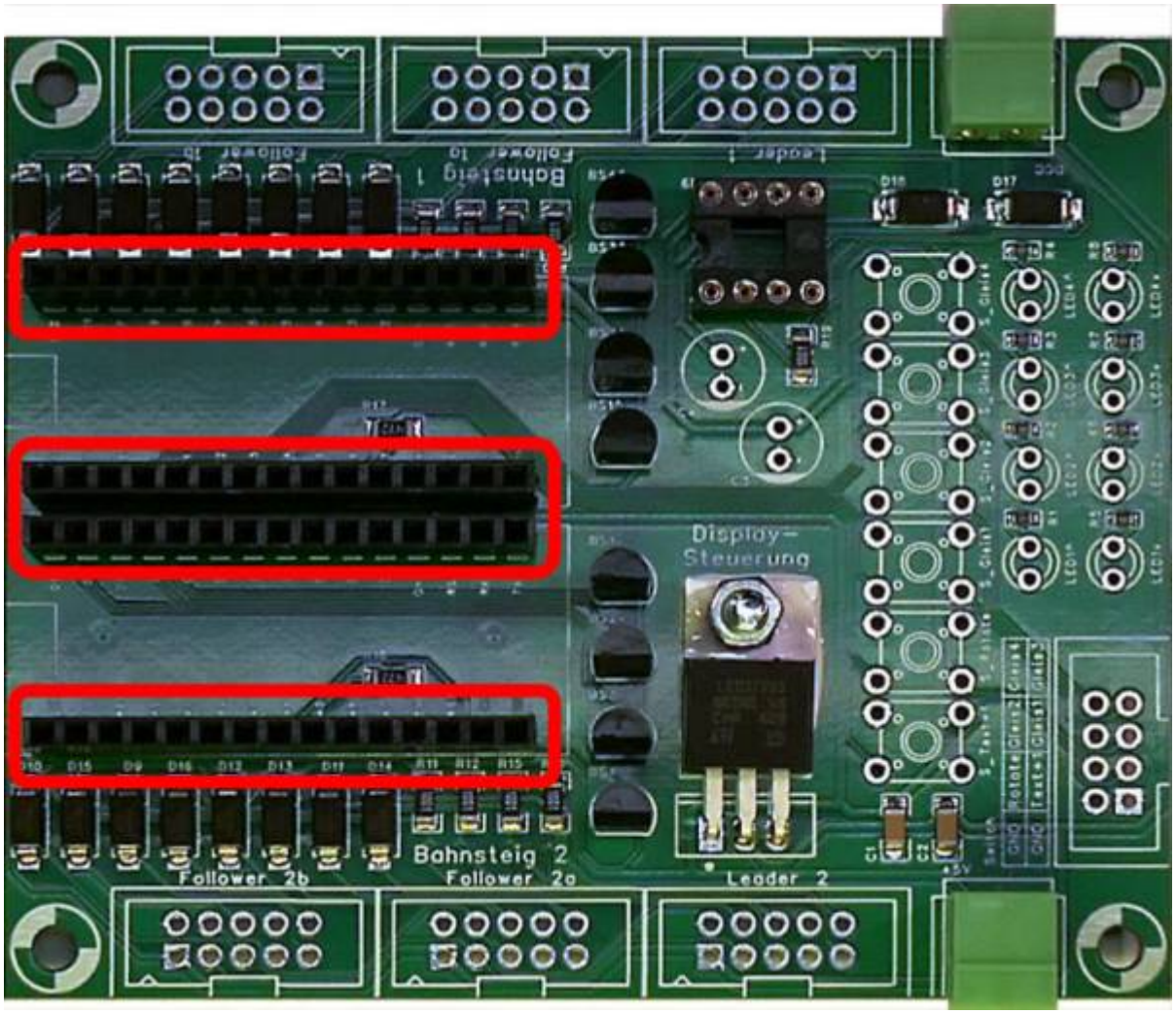
7) Mosfets BS170

Alle, die das Bauteil-Set bei Frank bestellt haben, müssen die Mosfets entgegen der Platinenbeschriftung mit der flachen Seite nach oben einlöten. Alle anderen müssen vor der Montage die Ausrichtung des erworbenen Mosfets mit einen Bauteiltester überprüfen oder es ausprobieren und die Mosfets ggf. nochmal tauschen. Vielleicht empfiehlt es sich, zunächst die Mosfets BS1^ und BS2^ einzulöten und zwei Displays anzustecken. Zeigen beide den gleichen Text, sind die Mosfets verdreht.

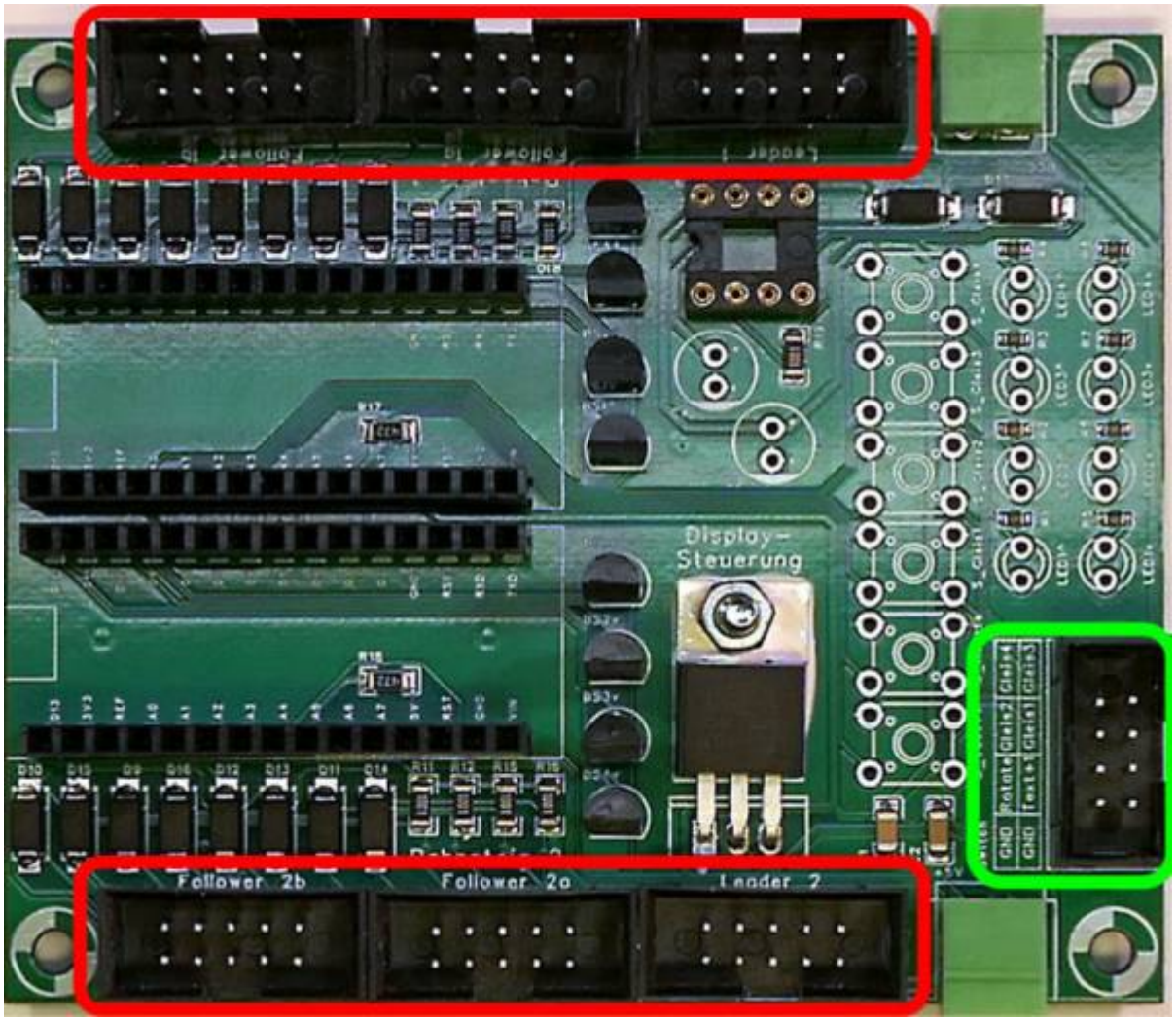


8) Anschlussklemmen +5V und DCC (Bild 1) und Buchsenleisten A1 und A2 (Bild 2)

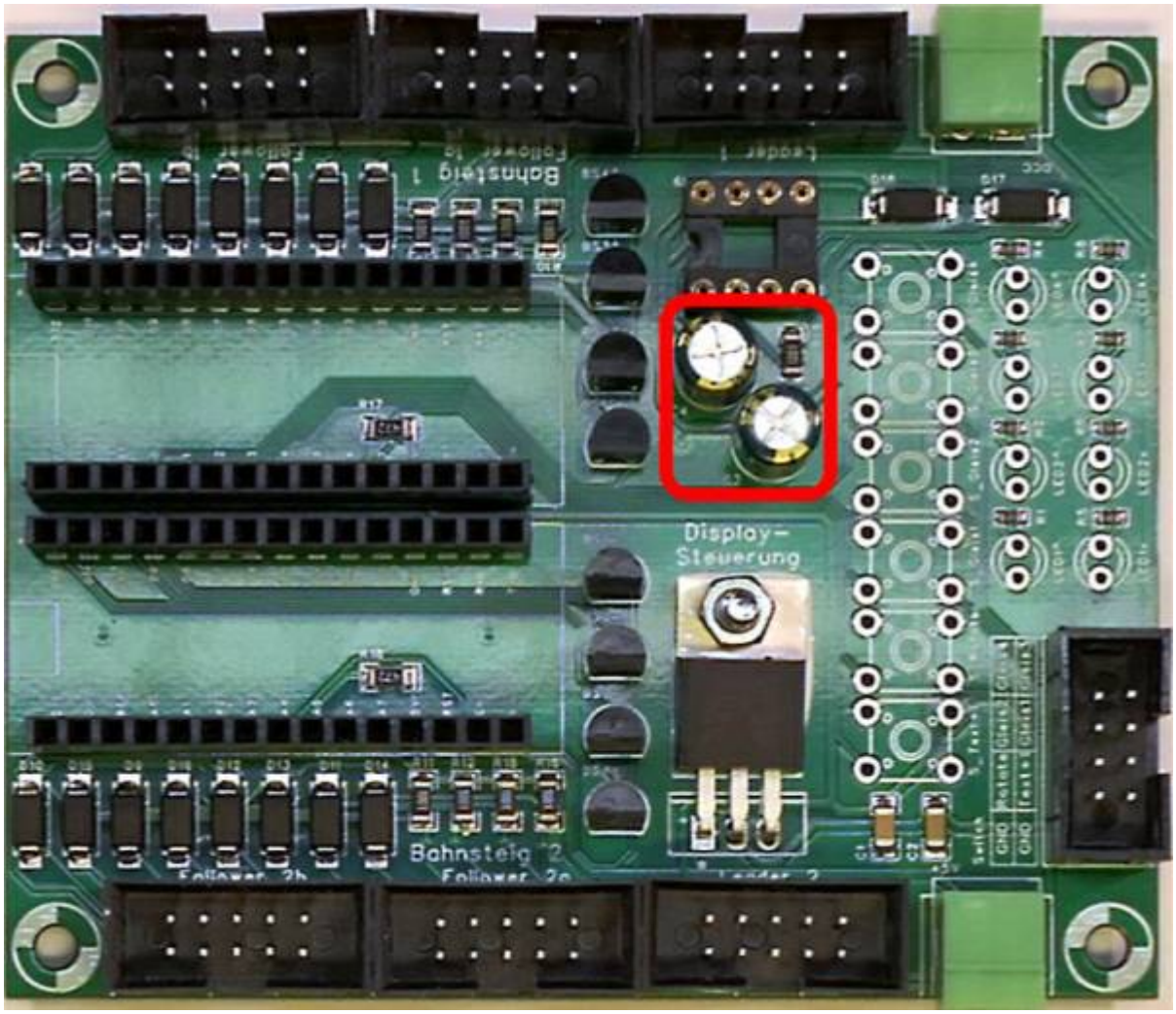

Bahnsteig 1
Follower 1b
Follower 1a
Leader 1
Display-
Steuerung
Bahnsteig 2
Follower 2b
Follower 2a
Leader 2
Chain4
Chain2
Chain1
Chain3
Rotate
Texte
GND
GND



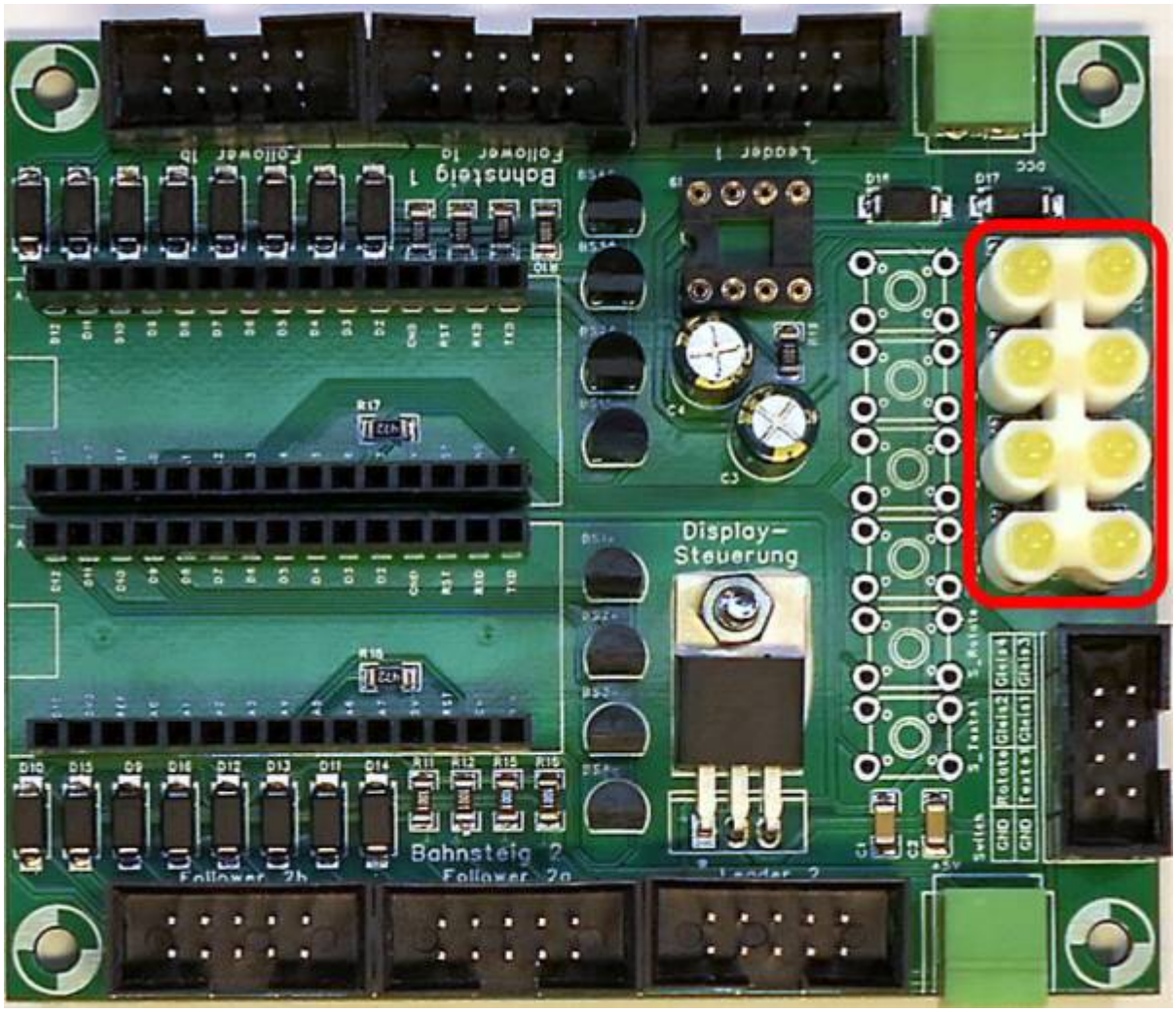
9) 10-polige Wannenstecker Leader + Follower (rot), 8-poliger Wannenstecker Switch (grün)



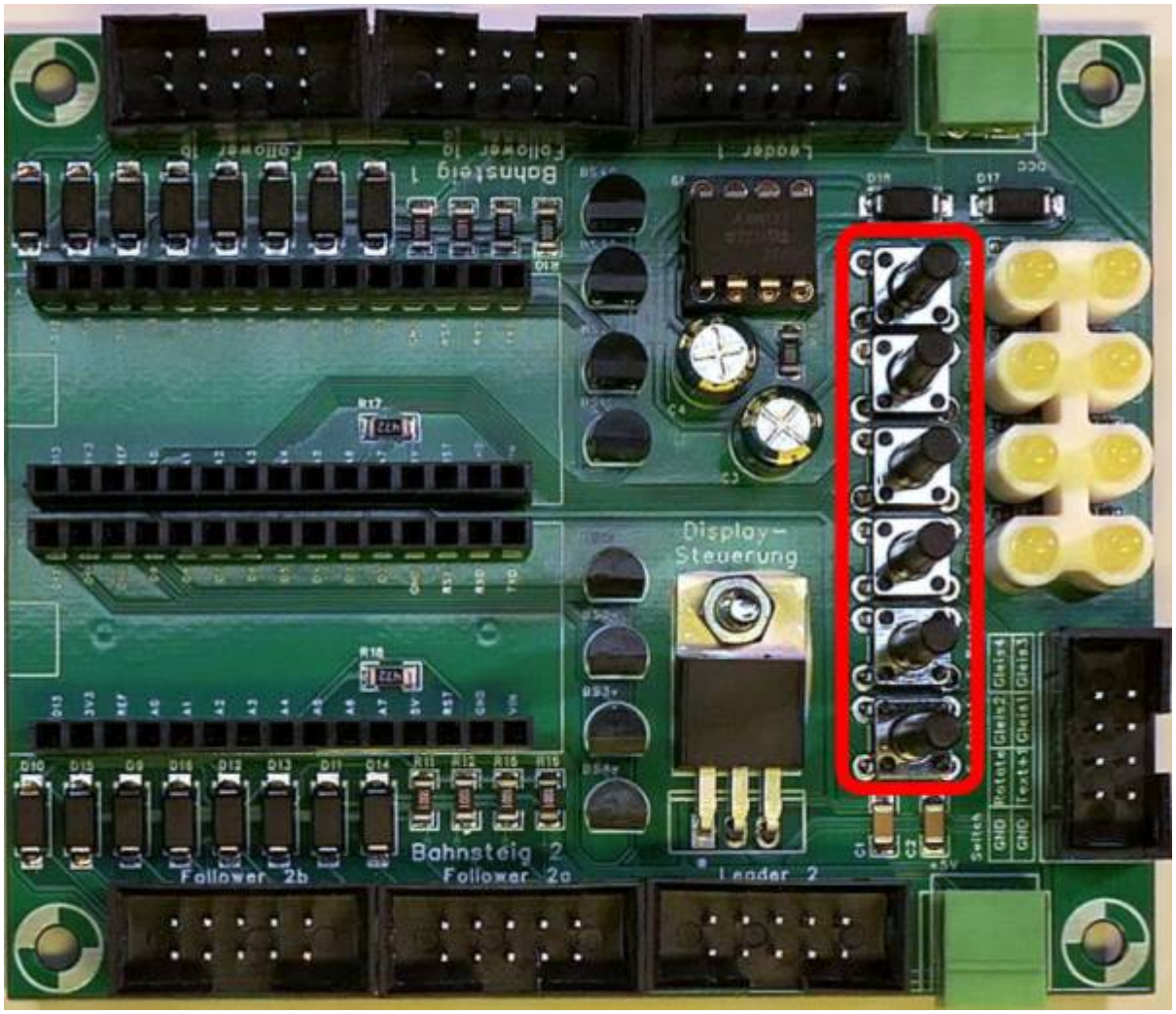
10) Kondensatoren C3 und C4 (Bild 1), LEDs 1-8 (Bild 2) inkl. 3D Abstandshalter



Achtung: Auf richtige Polung der LEDs achten. Die abgeflachte Seite ist auf der Platine nur zu erahnen, sie zeigt nach oben, wenn man die Platine wie hier gezeigt betrachtet. Der LED-Abstandhalter hat ebenfalls eine abgeflachte Seite, die wie hier im Bild gezeigt montiert werden muss.


Bahnsteig 1
Follower 1a
Follower 1b
Leader 1
Display-
Steuerung
Bahnsteig 2
Follower 2b
Follower 2a
Leader 2
GND
Rotate
Gleis2
Gleis1
GND
Taster
Gleis1
Gleis2
Switch
+5V

11) Taster



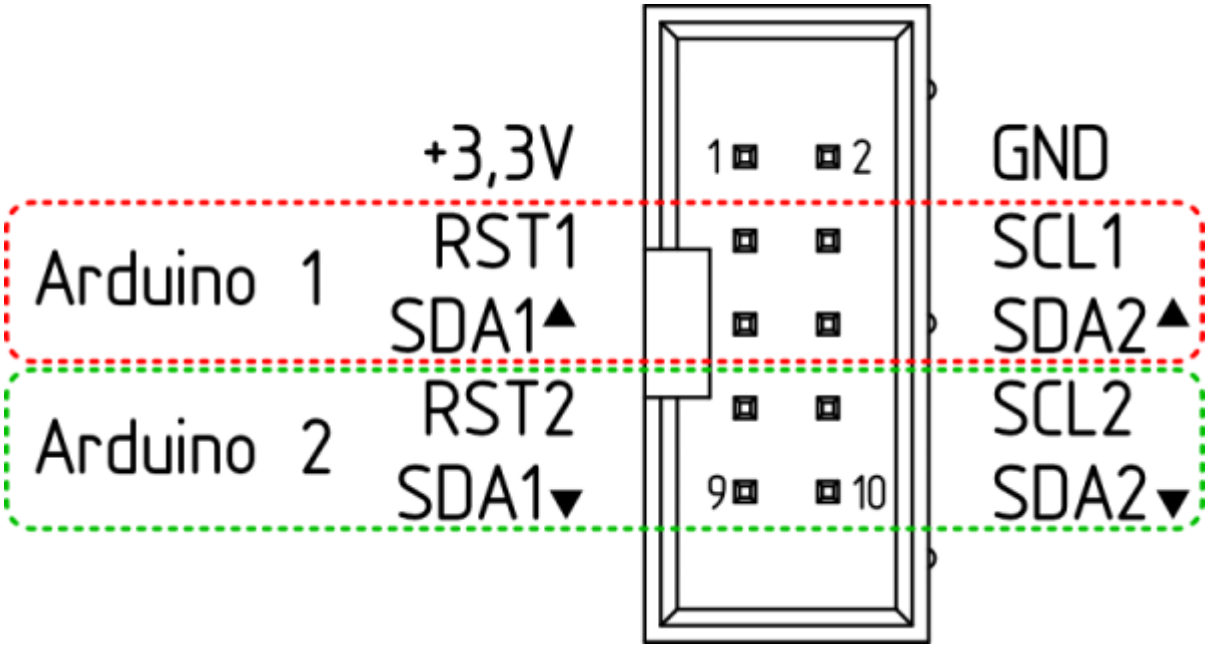
Vielen Dank an Frank für die Bereitstellung der einzelnen Bilder der jeweiligen Bauschritte!

Der erste Funktionstest

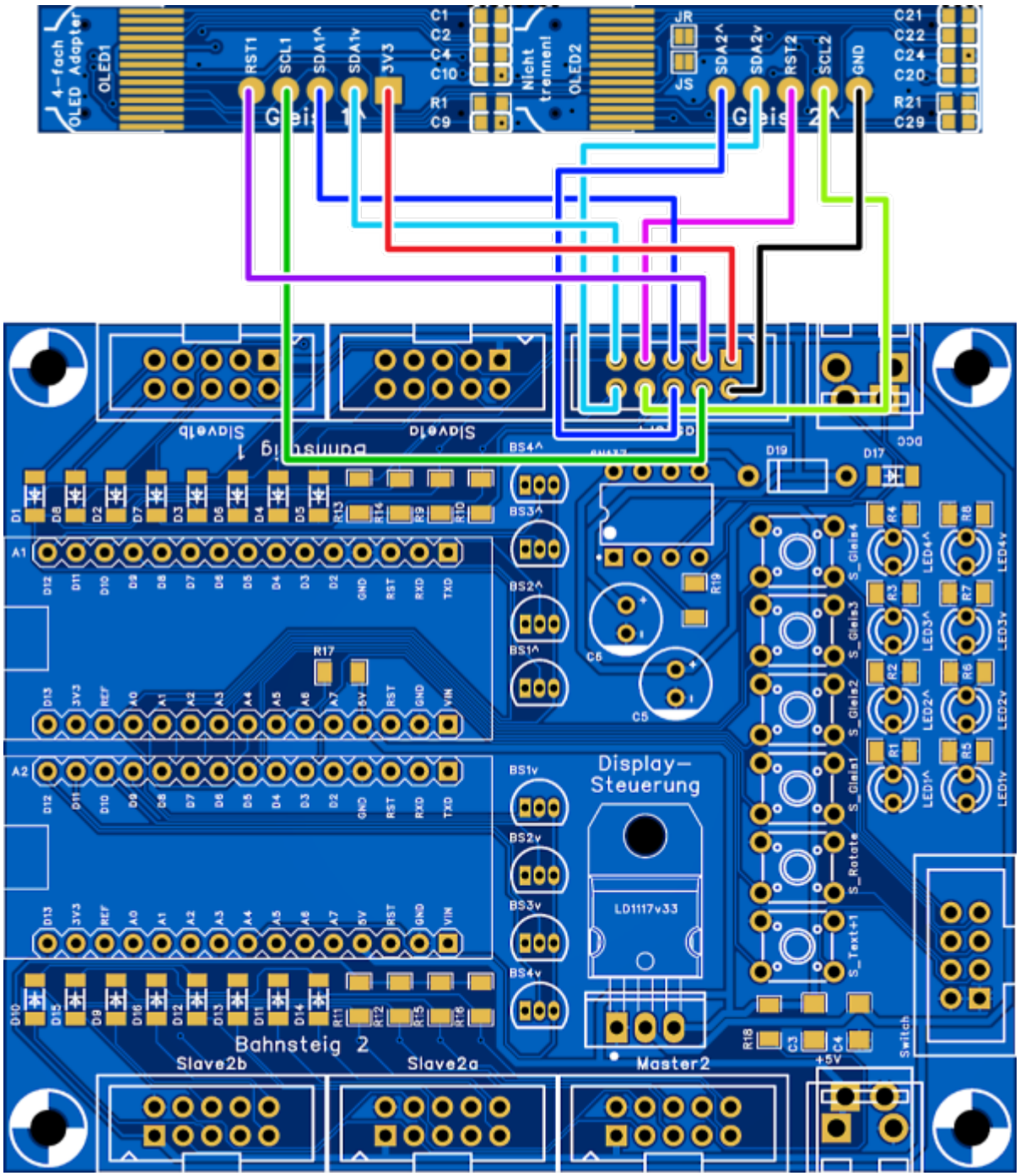
Noch bevor die OLEDs abgeschlossen werden, sollte an dieser Stelle der erste Funktionstest durchgeführt werden. Das Programm schaltet nach dem Anlegen der Versorgungsspannung die acht LEDs nacheinander an, beginnend mit dem Pärchen für Gleis 4, dann 3, 2 und schließlich 1. Die beiden LEDs für Gleis 1 bleiben am Ende an. Nun sollten sich die LEDs mit dem Taster „R“ rotierend durchschalten lassen (1>2>3>4>1>...).

Anschluss-Schema an OLED-Adapter

Pin-Belegung des Wannensteckers

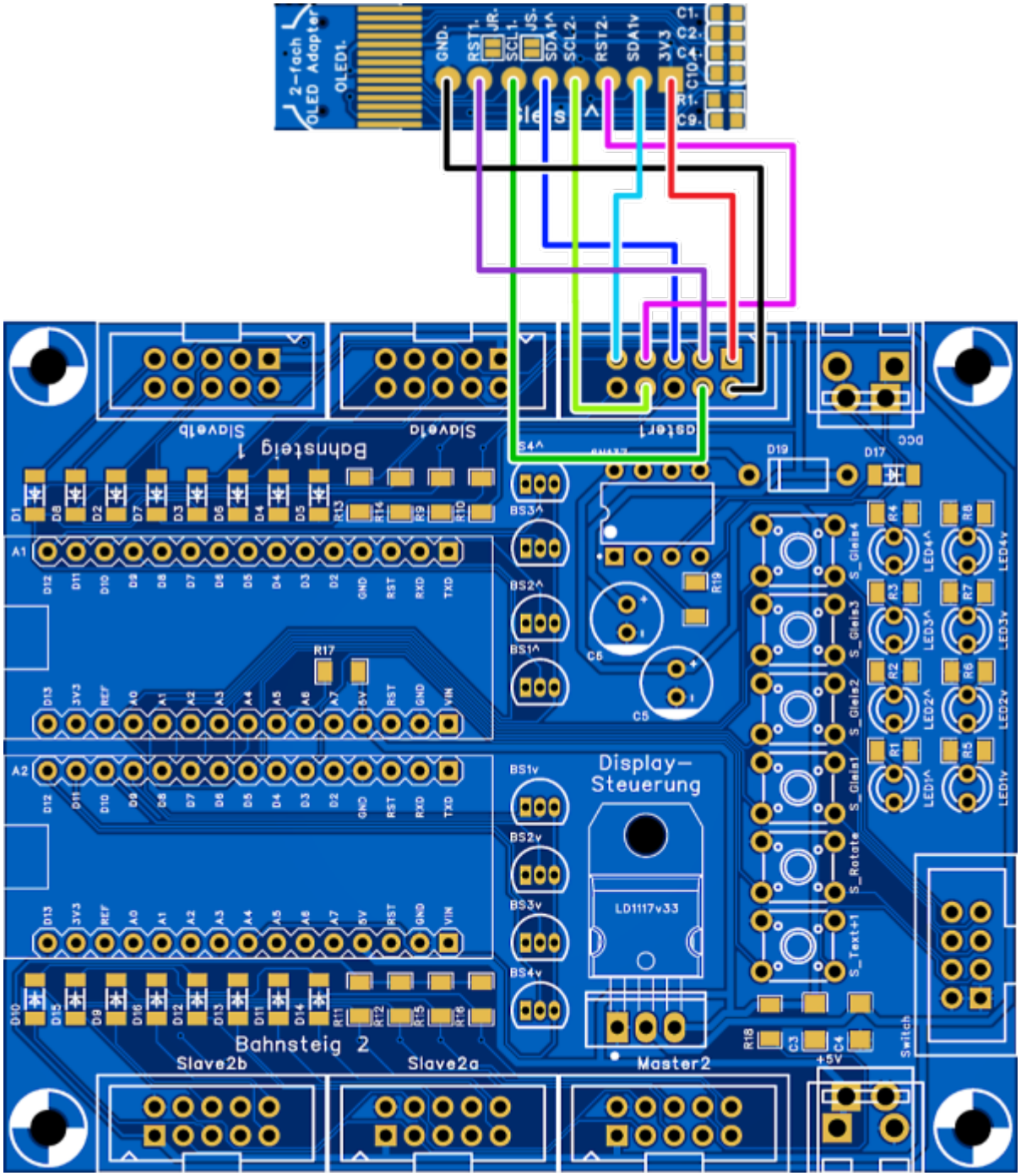


Zweigleisiger Zugzielanzeiger



Das oben gezeigte Anschluss-Schema gilt für alle sechs Wannenstecker. Auf diese Weise können bis zu 24 Displays an die Steuerung abgeschlossen werden.

Eingleisiger Zugzielanzeiger



Download des Arduino Sketches

Der Sketch ist hier zu finden:
https://github.com/raily74/MobaLedLib/blob/main/OLED/Zugzielanzeiger/Sketch/Bahnsteiganzeige_Michael.zip

Erste Schritte mit dem Sketch

Es empfiehlt sich, Hardis ursprünglichen Beitrag im Stummiforum zu lesen, bevor man in das Editieren des Sketchs einsteigt.
Zugzielanzeiger für den Bahnsteig mit Arduino

Gleise definieren

Der ursprüngliche Sketch wurde für die Verwendung mit der Display-Steuerung leicht modifiziert. Statt der ursprünglichen fünf Taster verwendet die hier gezeigte Platine sechs Taster und vier statt drei Displays.
Zudem arbeitet die Platine direkt mit zwei Arduinos, welche unterschiedliche Sketches benötigen. Da dieser modifizierte Sketch speziell auf die hier gezeigte Platine abgestimmt ist, steht er alternativ (bald) im Github zum Download zur Verfügung.

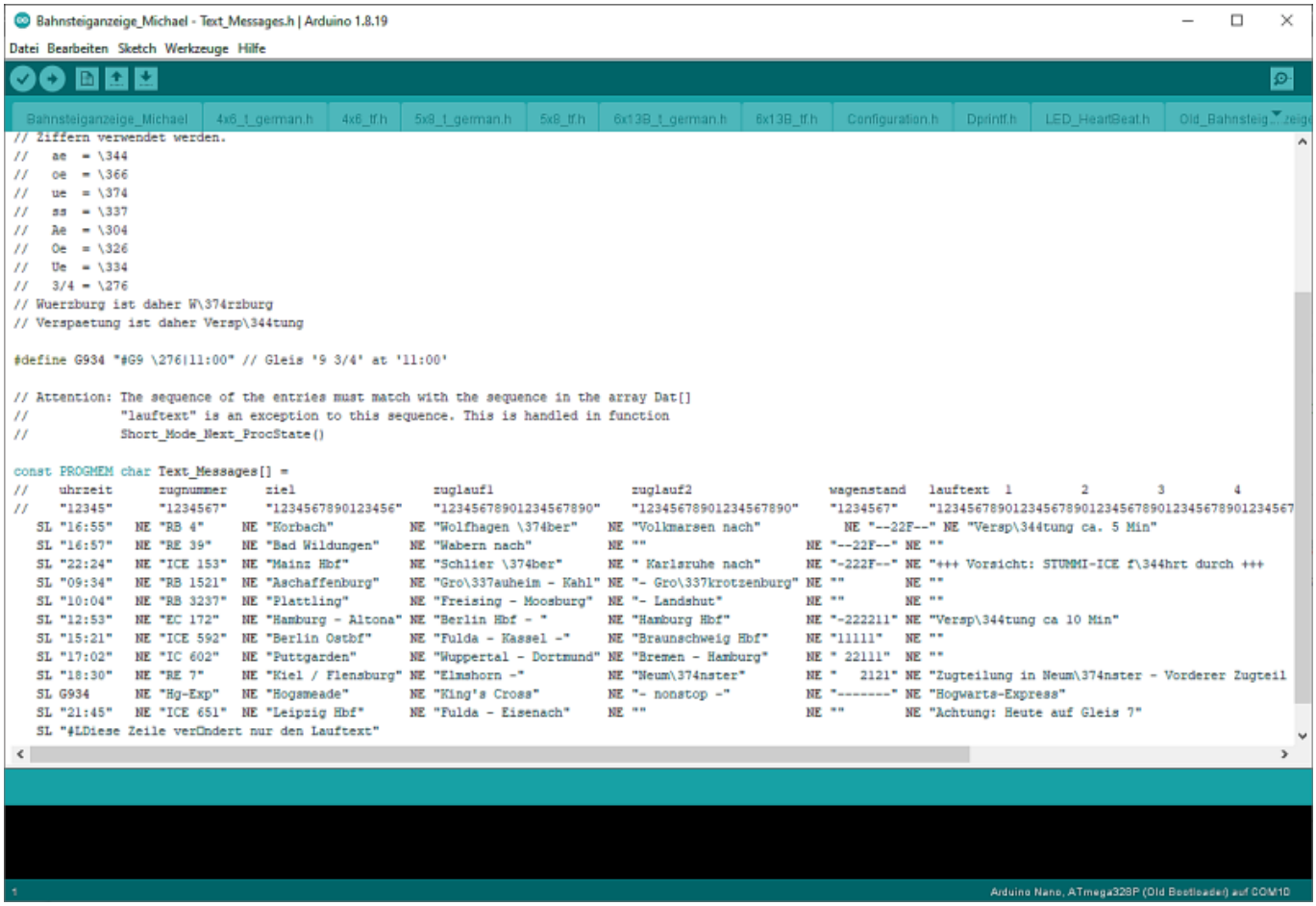
Ohne Anpassung des Sketches hätten die anzuzeigenden Texte (TextMessages.h) zudem immer in zwei Programmen gepflegt werden müssen - einmal für die vorwärts und einmal für die rückwärts gerichteten Displays. In der Praxis wird man nicht alle 100 Texte auf einmal pflegen, sondern immer dann, wenn einem ein besonderer Zug in die Finger gerät und man mit diesem spielt. Das ist der richtige Zeitpunkt zur Anlage seines Textes. Das hätte das Einpflegen der Texte und das Programmieren der Arduinos zum Geduldsspiel gemacht.

Hier greift der Sketch nun auf eine Besonderheit der Platine zurück. Beide Arduinos sind bis auf einen Unterschied identisch angeschlossen. Arduino 1 wird an Pin „A0“ auf GND gezogen. Das fragt der Sketch ab und setzt die Gleisnummer automatisch nach links oder nach rechts.

Bei der Ersteinrichtung müssen die auskommentierten Zeilen nach Wunsch aktiviert werden, indem die beiden Schrägstriche in der jeweiligen Zeile entfernt werden (1). Der Sketch ist zunächst für ein OLED Panel konfiguriert und eignet sich in der Form für einen ersten Test mit einem über ein Breadboard angeschlossenen Display. Dies ist in der Regel der erste Test, den man nach dem Zusammenbau durchführt.

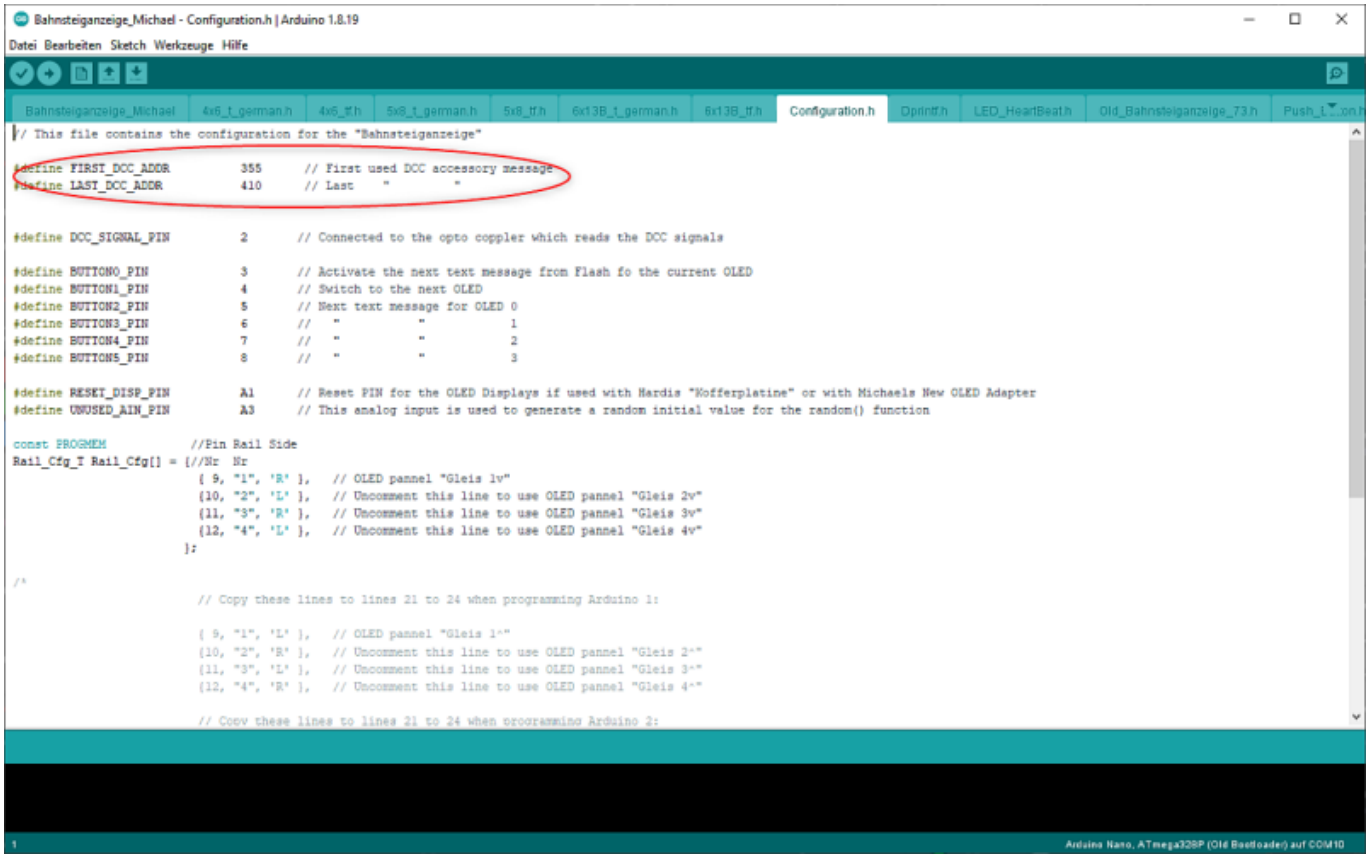
Texte für die Züge definieren

Wenn man den Arduino Sketch offen hat, sieht man oben die Reiter für die einzelnen Bestandteile. Einer davon ist die Sammlung der Anzeigentexte (TextMessages.h). Hier legt man einen passenden Text für den Zug seiner Wahl an. Dabei empfiehlt es sich, die Beispieltexte auszukommentieren oder bei Nichtgebrauch zu löschen. Texte sollten unbedingt von 1 bis 100 fortlaufend ergänzt werden und nicht im Nachgang sortiert werden. In den meisten Fällen wird man den soeben angelegten Text mit seiner Zeilennummer einem Zug zuweisen. Ergänzt man nachträglich oberhalb dieser Zeile einen weiteren Text oder sortiert die Texte um, gehen diese direkten Verknüpfungen aus Zug und Zeilennummer verloren.



DCC Adressbereich definieren

Als nächstes den Adressbereich in der „Configuration.h“ über die Arduino IDE einstellen und den Sketch auf beide Arduinos laden. Für den vollen Funktionsumfang (100 Texte) werden 56 DCC-Adressen benötigt (im gezeigten Beispiel ⇒ 355 bis 410 = 56).



Fehlerbehebung

Die Displays sind empfindlich und man muss beim Zusammenbau sehr sorgfältig arbeiten. Hier eine Auflistung der häufigsten Fehlerursachen.

Fehlerbild	Ursache	Behebung
Zwei Displays nebeneinander zeigen gleichen Text	Mosfet BS170 ist verdreht	Mit Bauteiltester wie oben beschrieben prüfen und ggf. drehen.
Display flackert	Kurzschluss zwischen den hauchdünnen Leiterbahnen des Flexkabels	Unter der Lupe mit viel Licht auf Kurzschlüsse prüfen und ggf. neu löten.
Hieroglyphen auf dem Display	Gekreuzte Signalleitungen SDA1+2⇑ und SDA1+2⇓	Kabel möglichst parallel und ohne Kreuzen verlegen, SDA1⇑ und SDA2⇑ nicht zu dicht an SDA1⇓ und SDA2⇓ verlegen.
Display-Beleuchtung sehr schwach	Schlechte Verbindung zwischen Platine und Display	Lötverbindungen am Flexkabel prüfen und ggf. Flexkabel auf Bruch untersuchen.

Steuerung per DCC

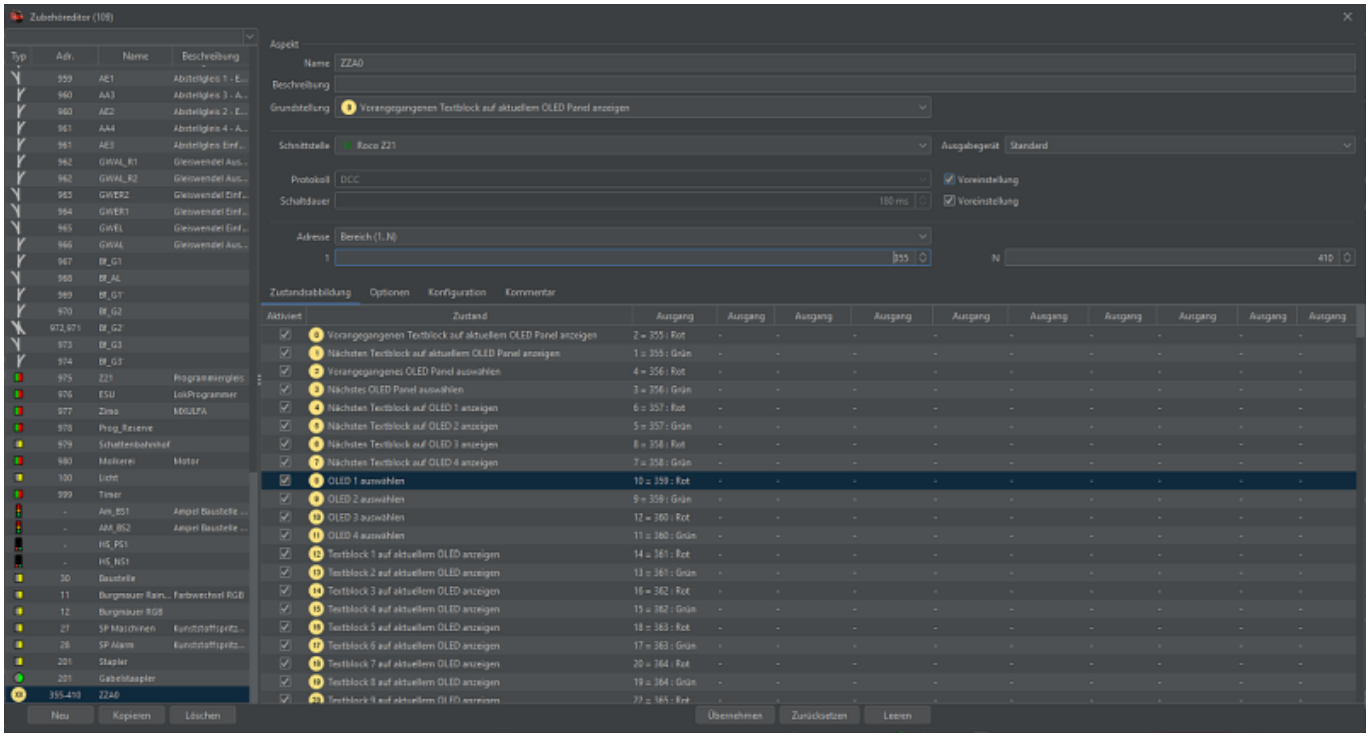
Der größte Clou der Zugzielanzeiger ist die Möglichkeit, den Anzeigentext vom einfahrenden Zug steuern zu lassen.

Aspekt*	DCC-Adr.	Zustand	Funktion	Bemerkungen
0	n+0	Rot	Vorangegangenen Textblock auf aktuellem OLED Panel anzeigen	wird zur Automatiksteuerung nicht benötigt
1	n+0	Grün	Nächsten Textblock auf aktuellem OLED Panel anzeigen	wird zur Automatiksteuerung nicht benötigt
2	n+1	Rot	Vorangegangenes OLED Panel auswählen	wird zur Automatiksteuerung nicht benötigt
3	n+1	Grün	Nächstes OLED Panel auswählen	wird zur Automatiksteuerung nicht benötigt
4	n+2	Rot	Nächsten Textblock auf OLED 1 anzeigen	Wichtige Funktion nach Verlassen von Gleis 1
5	n+2	Grün	Nächsten Textblock auf OLED 2 anzeigen	Wichtige Funktion nach Verlassen von Gleis 2
6	n+3	Rot	Nächsten Textblock auf OLED 3 anzeigen	Wichtige Funktion nach Verlassen von Gleis 3
7	n+3	Grün	Nächsten Textblock auf OLED 4 anzeigen	Wichtige Funktion nach Verlassen von Gleis 4
8	n+4	Rot	OLED 1 auswählen	Wird als erstes von einfahrendem Zug auf Gleis 1 ausgelöst
9	n+4	Grün	OLED 2 auswählen	Wird als erstes von einfahrendem Zug auf Gleis 2 ausgelöst
10	n+5	Rot	OLED 3 auswählen	Wird als erstes von einfahrendem Zug auf Gleis 3 ausgelöst
11	n+5	Grün	OLED 4 auswählen	Wird als erstes von einfahrendem Zug auf Gleis 4 ausgelöst
12	n+6	Rot	Textblock 1 auf aktuellem OLED anzeigen	Wird anschließend mit Verzögerung (<1s) an das aktuelle Display gesendet
13	n+6	Grün	Textblock 2 auf aktuellem OLED anzeigen	
14	n+7	Rot	Textblock 3 auf aktuellem OLED anzeigen	
15	n+7	Grün	Textblock 4 auf aktuellem OLED anzeigen	
16	n+8	Rot	Textblock 5 auf aktuellem OLED anzeigen	
17	n+8	Grün	Textblock 6 auf aktuellem OLED anzeigen	
18	n+9	Rot	Textblock 7 auf aktuellem OLED anzeigen	
19	n+9	Grün	Textblock 8 auf aktuellem OLED anzeigen	
…				
110	n+55	Rot	Textblock 99 auf aktuellem OLED anzeigen	
111	n+55	Grün	Textblock 100 auf aktuellem OLED anzeigen	

Mit der oben gezeigten Tabelle ist das Erstellen einer Regel ganz einfach. Die Verknüpfung wird hier am Beispiel von iTrain gezeigt. Zur Erstellung einer „Wenn/Dann-Regel“ in anderen Programmen muss deren Anleitung zu Rate gezogen werden.

Neuen Aspekt* als Zubehör in iTrain definieren

- Im Zubehöreditor von iTrain (Strg+F8) wird ein neues Zubehör vom Typ „Aspekt“ erstellt. Diesem gibt man einen frei wählbaren, sinnvollen Namen (z.B. „ZZA Hauptbahnhof“ oder wie im Bild „ZZA0“).
- Als Adresse wählt man „Bereich (1…N)“ und trägt bei 1 die zuvor in der Arduino IDE definierte Adresse „355“ und bei N die Adresse 410 ein. Es werden automatisch 56 DCC Adressen reserviert.
- Als nächstes müssen alle 112 Aspekte aktiviert werden. Ich kenne keinen Weg, wie man alle auf einmal aktivieren kann. Am schnellsten geht es daher, mit den Pfeiltasten zur zweiten Zelle in der Spalte „Aktiviert“ zu navigieren und die Leertaste zu drücken. Dann kann man im Wechsel den Pfeil nach unten und die Leertaste drücken, bis alle 112 Aspekte aktiviert sind.
- Hat man bis hierher alles richtig gemacht, sollte der Zustand „Aspekt A0“ auf Ausgang „1 = 355 : Grün“ liegen und „Aspekt A1“ auf Ausgang „2 = 355 : Rot“. Wir benötigen es aber genau umgekehrt. „Aspekt A0“ muss auf Ausgang „2 = 355 : Rot“ und „Aspekt A1“ auf Ausgang „1 = 355 : Grün“. Das Vertauschen ist einfach. Man wählt zunächst den „Aspekt A0“ per Mausklick aus, dann bei gedrückter Umschalt-Taste (Shift) den letzten Aspekt („Aspekt A111“) und drückt die Taste „S“. Alternativ kann man über die rechte Maustaste das Kontextmenü aufrufen und „Vertausche die Ausgänge (S)“ wählen.
- Wer jetzt noch die Muse hat, kann die vorgegebenen Bezeichnungen „Aspekt A0“ bis „Aspekt A111“ in der Spalte „Zustand“ per Doppelklick umbenennen. Hier empfiehlt es sich, die Namen der Funktion zu verwenden. Diese befinden sich im MobaLedLib-Wiki. Das Umbenennen muss nur einmal gemacht werden. Für eine zweite Display-Steuerung mit anderem Adress-Bereich kann das Zubehör „ZZA0“ kopiert werden. Im Anschluss kann einfach ein anderer Adressbereich für das Duplikat definiert werden.



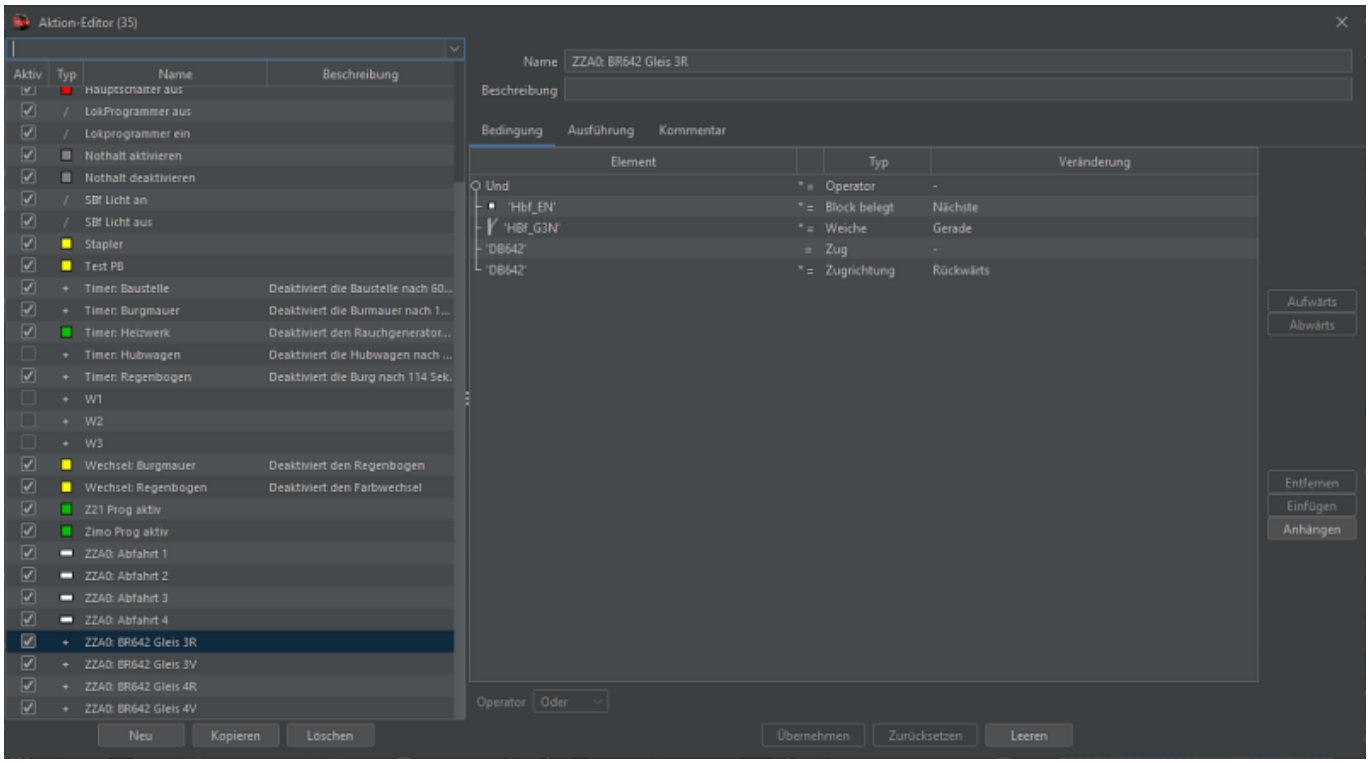
In einer Aktion werden nun die Bedingungen und die Ausführung miteinander verknüpft.

Beispiele für die Bedingung:
- ein bestimmter Block ist belegt
- im Block befindet sich ein bestimmter Zug (DCC-Adresse)

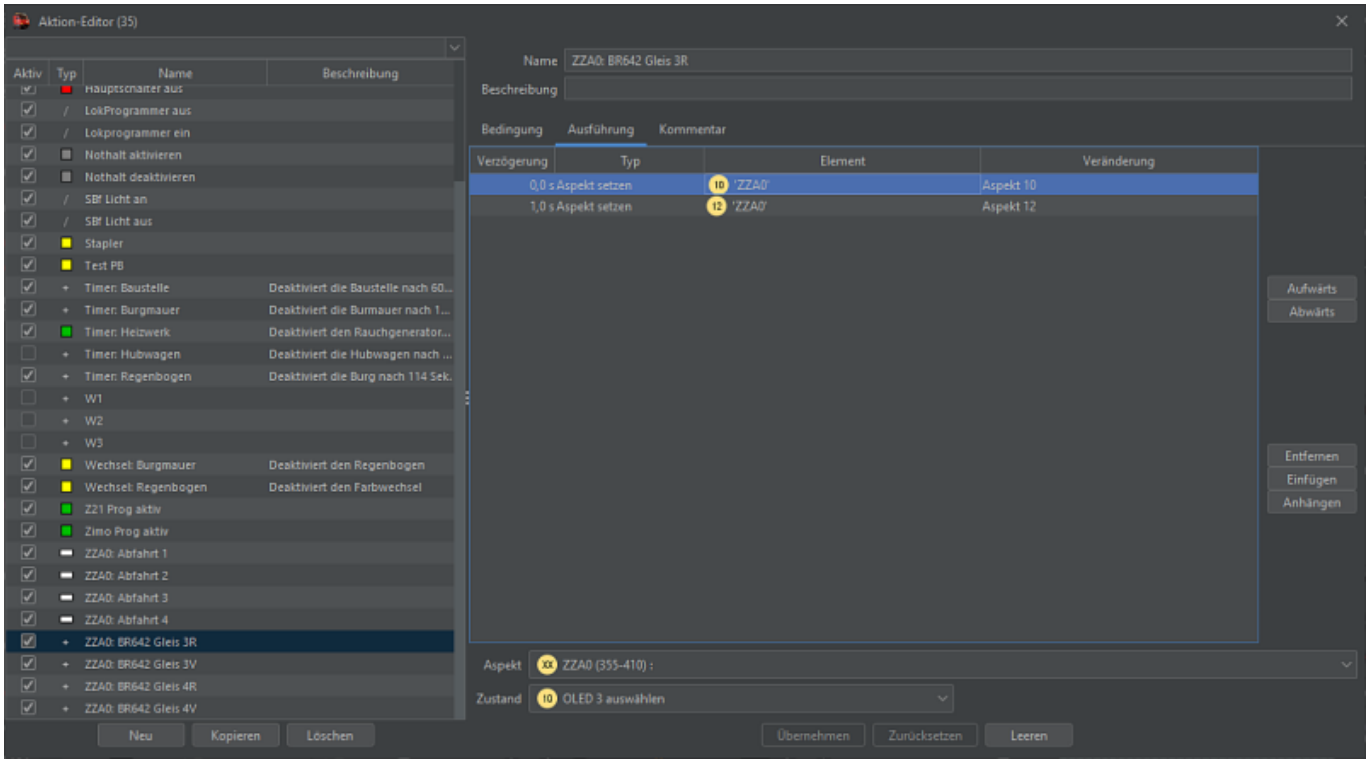
- die Fahrtrichtung ist vorwärts/rückwärts
- die folgenden Weichen führen zu einen bestimmten Gleis

Alle oben genannten Bedingungen sollten in einer „Und-Bedingung“ erfasst werden, sodass nur ausgeführt wird, wenn alle Anfragen mit „Ja“ beantwortet werden.

Die Ausführung schaltet lediglich zwei DCC Adressen. Auch hier ein Beispiel: An Gleis 3 soll der Text Nr. 1 angezeigt werden. Dazu wird zunächst die DCC Adresse n+5 auf Rot gesetzt (entspricht Aspekt 10) und mit einer kurzen Verzögerung die Adresse n+7 auf Rot (entspricht Aspekt 12). Im folgenden Beispiel wird für den Zug „DB642“ die Bedingung definiert, dass der Zug rückwärts fahrend den Block „Hbf_EN“ in Richtung Bahnhof belegt und die Weiche „Hbf_G3N“ auf gerade steht.



Sind alle vier Bedingungen erfüllt, wird der Aspekt „ZZA0“ auf den Zustand „10 (OLED3 auswählen)“ und mit einer Sekunde Verzögerung auf den Zustand „12 (Textblock 1 auf aktuellem OLED anzeigen)“ eingestellt. Das sorgt im Hintergrund dafür, dass die angeschlossene Zentrale die DCC-Adresse 360 auf Rot setzt und eine Sekunde später die DCC-Adresse 361 auf Rot. Der Arduino interpretiert diese beiden Signale, schaltet auf Display 3 um und zeigt dort Text 1 an. Für den vorwärts fahrenden „DB642“ wählt man beispielsweise Zustand „13 (Textblock 2 auf aktuellem OLED anzeigen)“. Auf Gleis 4 wählt man Zustand 14 und 15. So fährt derselbe Zug nicht immer zum selben Ziel.

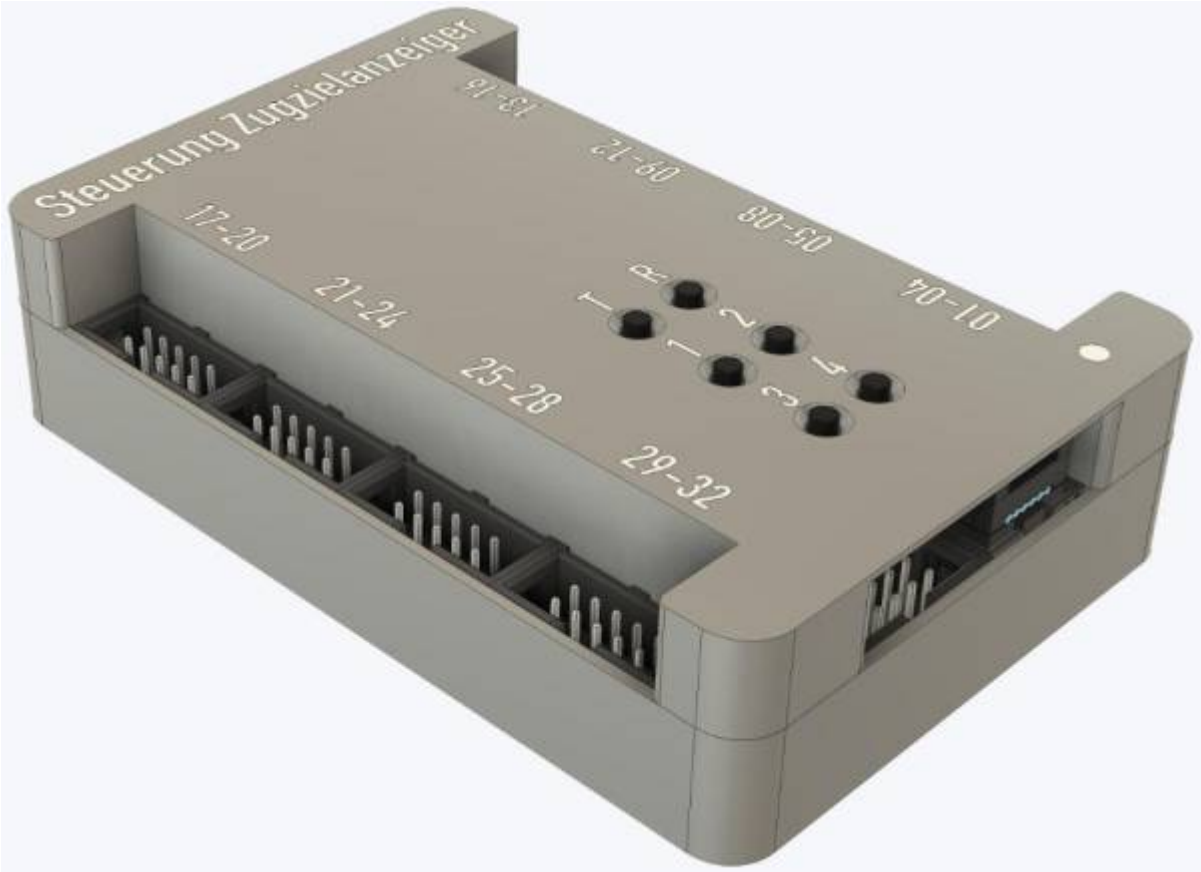


*) Der Aspekt ist eine spezielle Funktion in iTrain. Dazu wird ein beliebiger DCC Adressbereich mit maximal 128 Adressen in einen virtuellen „Drehschalter“ mit bis zu 256 Schaltstellungen verwandelt. Das macht das Senden der Display- und Textwahl einfacher. Die Programme WinDigipet, TrainController und RocRail werden ähnliche Möglichkeiten bieten, wenn auch unter anderem Namen. Zur Not kann auch einfach die jeweilige DCC-Adresse als Ausführung gesendet werden.
Hinweise zur Vorgehensweise dieser Programme bitte gern im Forum posten.

3D-Gehäuse - Display-Steuerung

Eignung für 3D-Drucker: FFF / FDM ★★★★★ SLA / STL ★★★☆

Die Druckdaten sind hier zu finden:
https://github.com/Hardi-St/MobaLedLib_Docu/tree/master/3D_Daten_fuer_die_MobaLedLib/Gehaeuse-740_Display-Steuerung_Zugzielanzeiger
Platzhalter - Datei folgt



Schaltplan