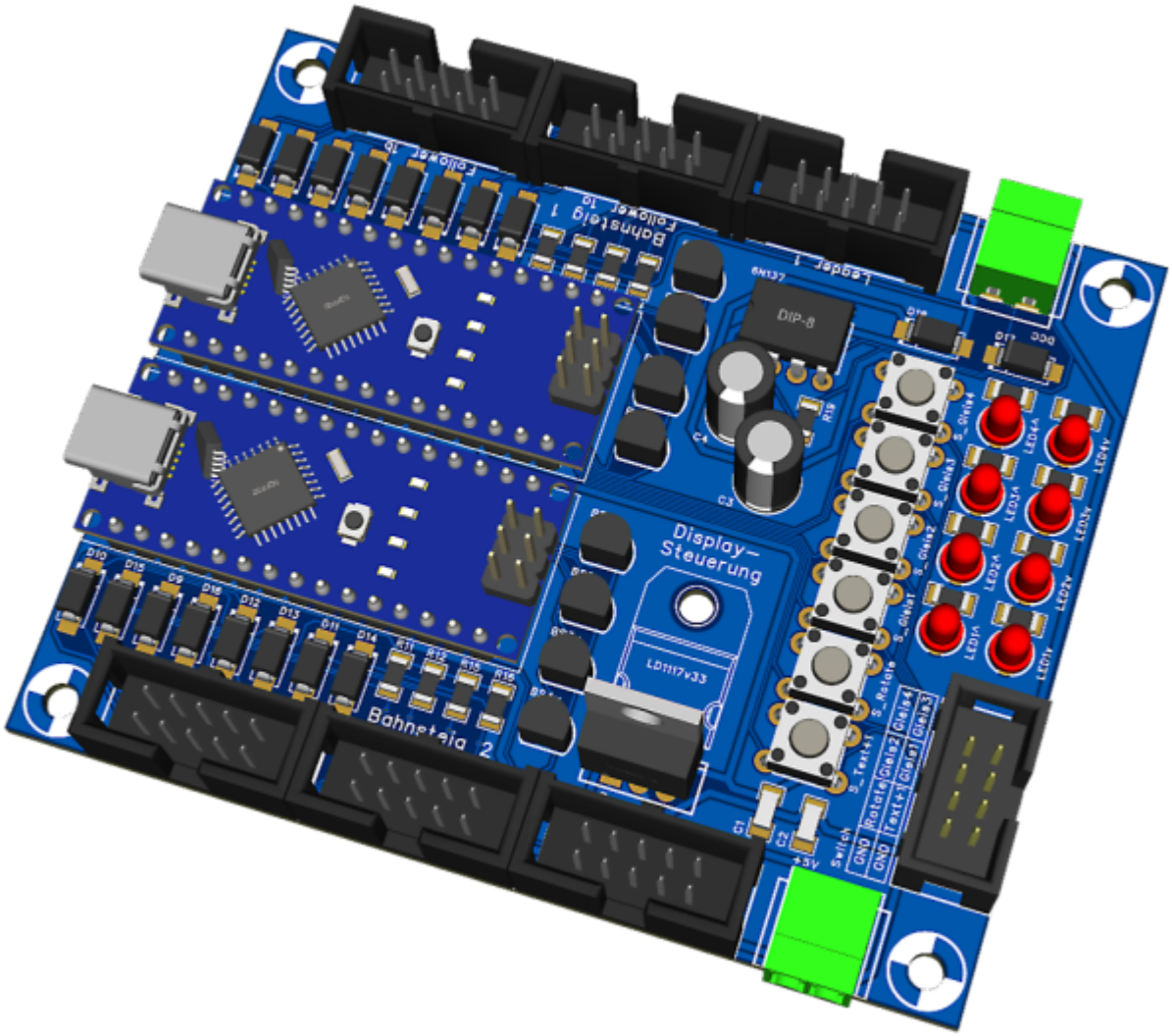
In Arbeit...

740 Display-Steuerung Zugzielanzeiger

Die Zugzielanzeiger haben mit der MobaLedLib an sich nichts zu tun. Sie sind ein völlig eigenständiges Projekt und werden **nicht** über den Programm Generator gesteuert. Initiiert wurden sie bereits im Dezember 2015 im Stummiforum durch TobiBS. Anfang 2019 wurde Hardi auf das Thema aufmerksam und entwickelte den vorhandenen Code zu einem optimierten Sketch weiter. Passend zum Sketch entwickelte er auch die 760 Kofferplatine zur Aufnahme der winzigen 0,87“-Displays sowie einen passenden Schaltplan zur Steuerung der Kofferplatinen. Hier knüpft fünf Jahre später die Display-Steuerung für Zugzielanzeiger an und führt das Projekt zu einem kompletten System.



Funktionsweise der Display-Steuerung

Die Display-Steuerung entstand ursprünglich als Erweiterung zur 760er Kofferplatine. Die Idee der

Platine basiert dabei auf dem ursprünglichen Schaltplan, jedoch erweitert um einen zweiten Arduino.

Warum ein zweiter Arduino?

Der ursprüngliche Schaltplan sowie der im Github zur Verfügung gestellte Sketch können mit nur einem Arduino ohne Weiteres fünf Displays ansteuern. Dabei nutzt die Schaltung mehrere Mosfets (BS170), um das jeweilige Display anzusprechen. Die Displays schalten also um und verharren mit dem zuletzt empfangenen Text. Immer auf dem aktiven Display (dessen Mosfet aktiviert ist) ist der Arduino in der Lage, einen bewegten Lauftext zu generieren. Dieser stoppt, sobald auf eines der anderen Displays umgeschaltet wird. Zugzielanzeiger bestehen häufig aus einem in Fahrtrichtung zeigenden und einem entgegengesetzten Monitor. Bei beiden ist die Gleisnummer jeweils außen angeschlagen und somit ist der Bildschirminhalt beider Monitore nie identisch. Da der Arduino aber nur auf einem Display den Lauftext generieren kann, würde die Verspätung eines einfahrenden Zuges nur auf dem Display in Fahrtrichtung angezeigt, nicht jedoch auf dem am selben Gleis angebrachten Display entgegen der Fahrtrichtung.

Nun kommt der zweite Arduino ins Spiel. Steuert man diesen mit denselben DCC-Befehlen, bzw. nutzt man dieselben Tasten zum Umschalten der Texte, so kann der eine Arduino den Lauftext in Fahrtrichtung generieren und der zweite generiert denselben Lauftext entgegen der Fahrtrichtung. Der Sketch ist dabei bis auf die Seite der Gleisnummer auf beiden Arduinos identisch.

Steuern zwei Arduinos also zehn Displays?

Im Prinzip ja. Auch wenn der Lauftext immer nur auf einem Gleis dargestellt werden kann, so nutzt diese Steuerung immerhin vier der zahlreichen Ausgänge jedes Arduinos zur Steuerung der Displays. Beide Arduinos zusammen liefern also Bildinhalte für acht Displays. Eine Platine versorgt daher zwei Bahnsteige mit jeweils zwei Gleisen. Ein Arduino versorgt vier Displays in die eine Himmelsrichtung (▲) und der zweite Arduino die vier Displays in die andere Himmelsrichtung (▼).

Soll der Lauftext zur gleichen Zeit auf zwei von vier Gleisen angezeigt werden, so muss eine Platine je Bahnsteig zum Einsatz kommen. Die Ausgänge für Bahnsteig 2 bleiben in dem Fall ungenutzt, der zweite Arduino muss trotz allem bestückt werden. Lediglich auf einige Dioden, Mosfets und Wannenstecker kann verzichtet werden (siehe Ende der Bestückungsanleitung).

Ist die Display-Steuerung kompatibel mit den Kofferplatinen?

Jein! Auch wenn die Kofferplatinen nicht dazu geeignet sind, einen beidseitigen Lauftext anzuzeigen, so ist jedoch die Display-Steuerung in der Lage, die Kofferplatinen zu betreiben.

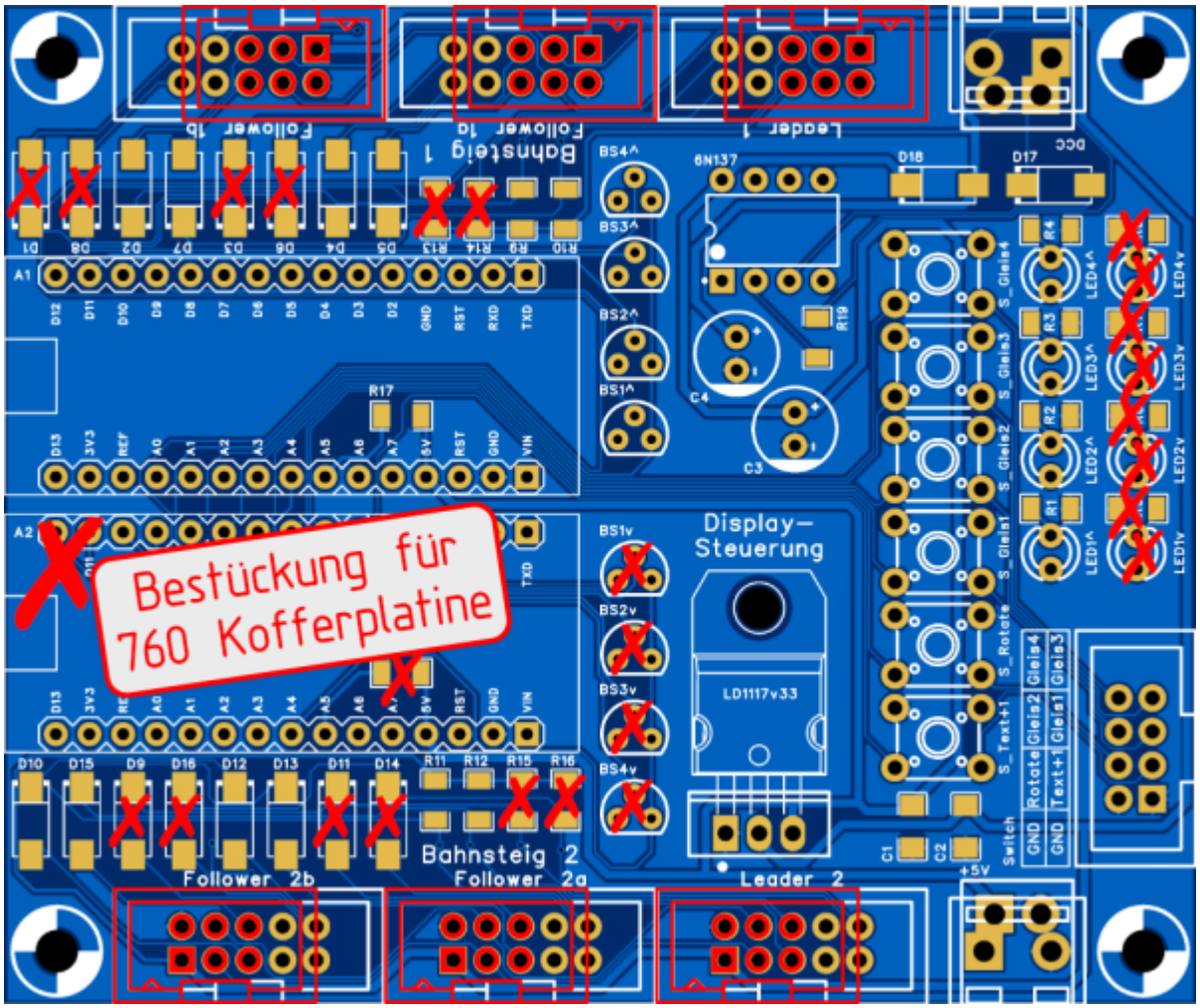
Die Arduinos der Display-Steuerung versorgen jedes Display mit den drei Signalen RST, SCL und SDA, sowie der Versorgungsspannung. Die Kofferplatine führt bis auf SDA alles für Vorder- und Rückseite zusammen. Soll nun das Display auf der Vorderseite von einem anderen Arduino angesteuert werden als das Display auf der Rückseite, so sind die Signalleitungen RST und SCL von beiden Arduinos nötig. Das kann die Kofferplatine nicht abbilden.

Die Kofferplatine ist eine großartige Entwicklung, um das sowieso zu große Display von der noch größeren Platine zu befreien. Speziell für die Display-Steuerung wurde allerdings ein eigener OLED

Adapter entwickelt, der entweder vier oder zwei Displays auf einer doppelseitigen Platine aufnimmt und so den vollen Leistungsumfang der Display-Steuerung ausreizt. Kofferplatine kann dieser Adapter aber nicht mehr genannt werden, da der „Griff“ beim neuen Adapter entfällt. Dazu aber mehr auf der Seite des OLED Adapters.

Für die Nutzung mit Kofferplatinen werden statt der 10-poligen Wannenstecker einfach 6-polige eingelötet und einige Bauteile können entfallen.

Soll die Platine universell eingesetzt werden, sollte sie voll bestückt werden. In dem Fall kann man eine sechsadrige Flachbandleitung wie bei der MobaLedLib gewohnt mit Plus am Pfeil einer zehnpoligen Pfostenbuchse ausrichten.



Stückliste

Nachfolgend findet man die Stückliste der notwendigen Bauteile inkl der Bestellnummern bei Reichelt.

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

Anzahl	Bezeichnung	Beschreibung	Bestellnummer	Alternativen, Bemerkungen
1	Board	Platine	740-Display-Steuerung	
18	D1-D16, D17, D18	Schottkydiode, 40 V, 1 A, DO-214AC/SMA	B 140 F	
8	R1-R8	SMD-Widerstand, 1206, 150 Ohm, 250 mW	WAL WR12X1500FTL	
9	R9-R16, R19	SMD-Widerstand, 1206, 1 kOhm, 250mW	WAL WR12X1001FTL	
2	R17, R18	SMD-Widerstand, 1206, 4,7 kOhm, 250mW	WAL WR12X4701FTL	
2	C1, C2	Vielschicht-Kerko, 100nF, 50V	KEM X7R1206B100N	
2	C3, C4	Elko radial, 100 uF, 25 V, 105°C, low ESR	FM-A 100U 25	
8	BS1▲▼-BS4▲▼	MOSFET, 60 V, 0,5 A, TO-92	BS 170	
1	LD1117v33	LDO-Spannungsregler, 15 Vin, 3,3 Vout	LD1117V33	
8	LED1▲▼-LED4▲▼	LED 3mm, bedrahtet, gelb	3004Y1D-EHB-A	
1	6N137	Optokoppler	6N 137	
1	Fassung für 6N137	IC-Sockel, 8-polig	GS 8	
2	+5V, DCC	Stiftleiste - 2-pol, RM 3,5 mm, 90°	CTB932HD-2	Alternative: siehe nächste Zeile
2	+5V, DCC	Anschlussklemme, 2-pol, Ø 2 mm, RM 5,08	AKL 101-02	
6	Leader, Follower	Wannenstecker, 10-polig	WSL 10G	
1	Switch	Wannenstecker, 8-polig	BKL 10120552	Günstiger bei Pollin oder AliExpress
6	S_Text+1, S_Rotate, S_GleisX	Kurzhubtaster 6x6mm, Höhe: 20,0mm	Schalter Dip 6x6x20	
4	A1, A2	Buchsenleiste, 15-pol	BL 1X20G8 2,54	**Diese Buchsenleiste muss leider geteilt werden. Bei Conrad ist auch die 15-polige Variante erhältlich.**
2	A1, A2	Arduino Nano	ARDUINO NANO	Günstiger bei AliExpress

Buchsenleiste teilen

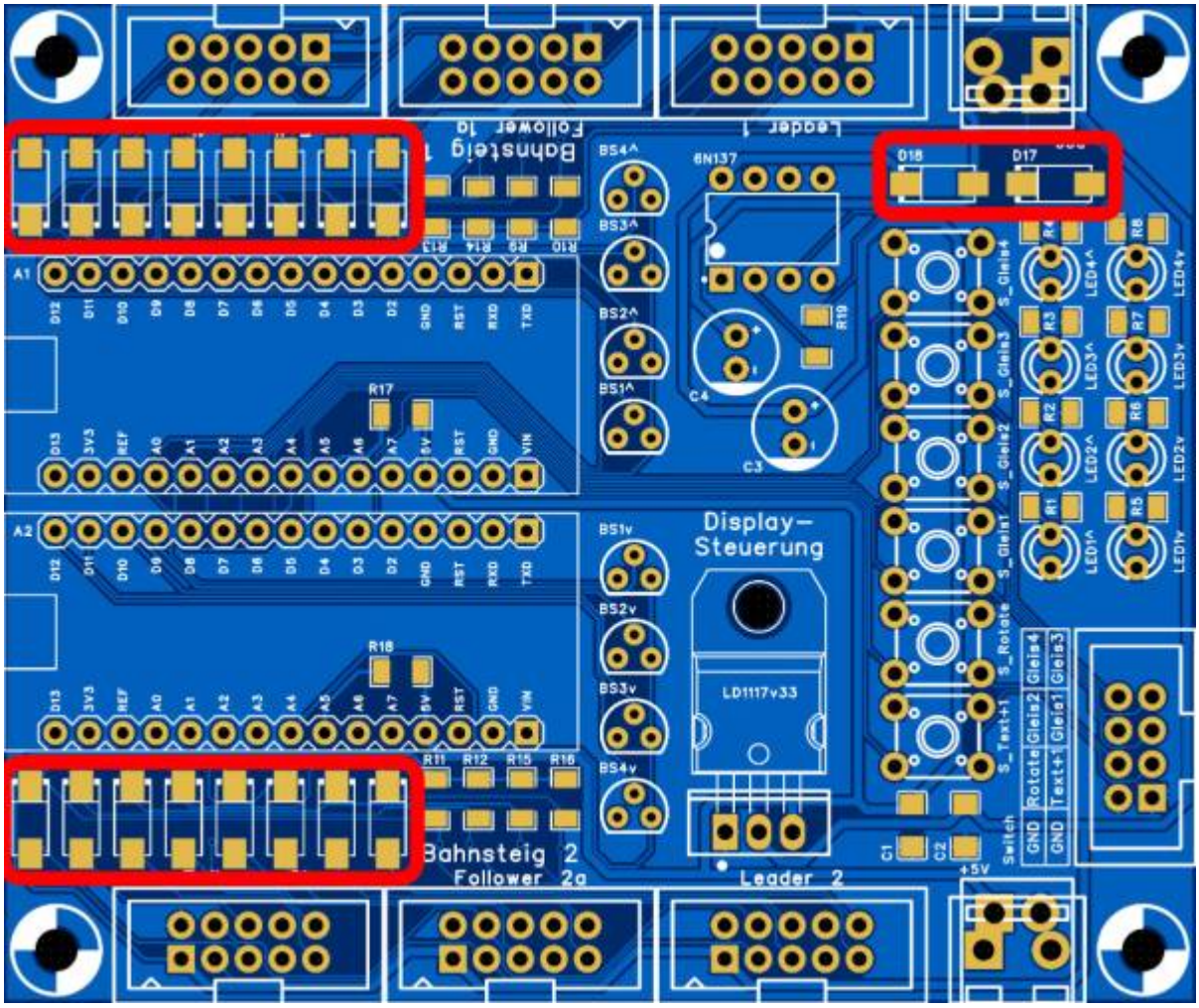
Die 20poligen Buchsenleiste für A1 und A2 wird per Säge auf die notwendigen Teilstücke abgelängt (jeweils etwa 1mm hinter dem letzten benötigten Bein absägen). Aus einer 20poligen Leiste wird eine 15-polige Buchsenleisten erstellt.

Bestückung der Display-Steuerung

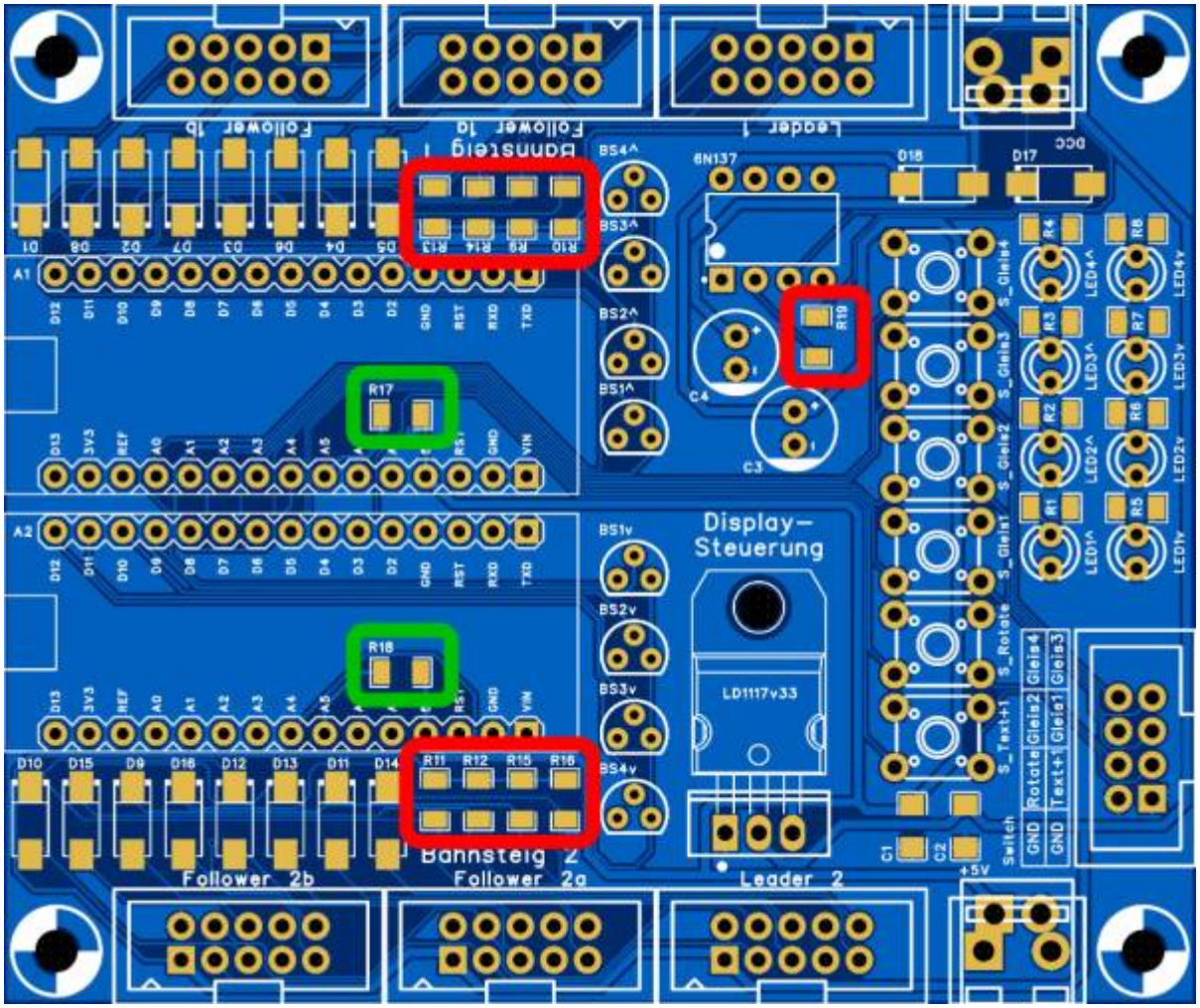
Es empfiehlt sich, folgende Reihenfolge bei der Bestückung einzuhalten:

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

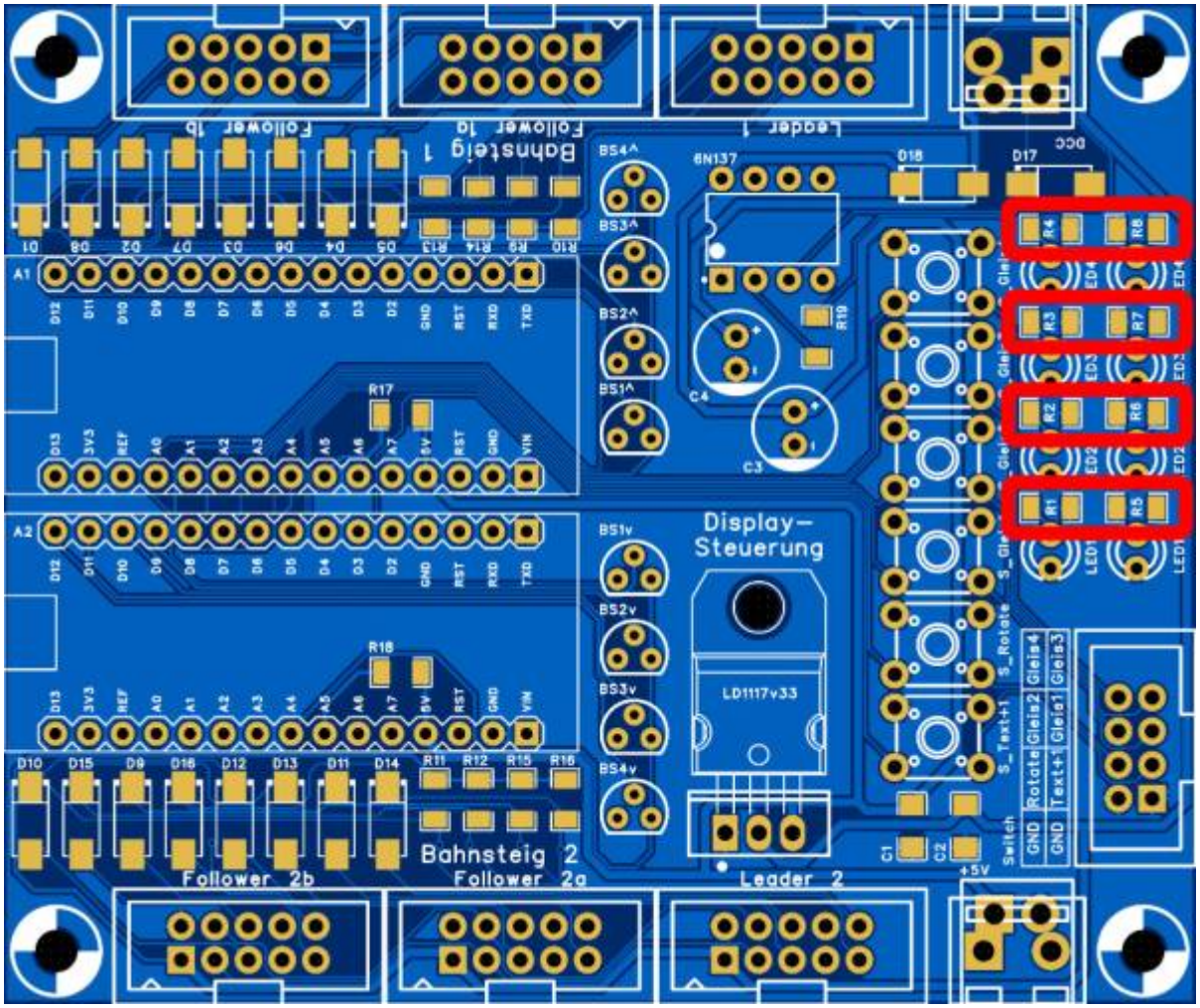
1) Dioden D1 bis D18



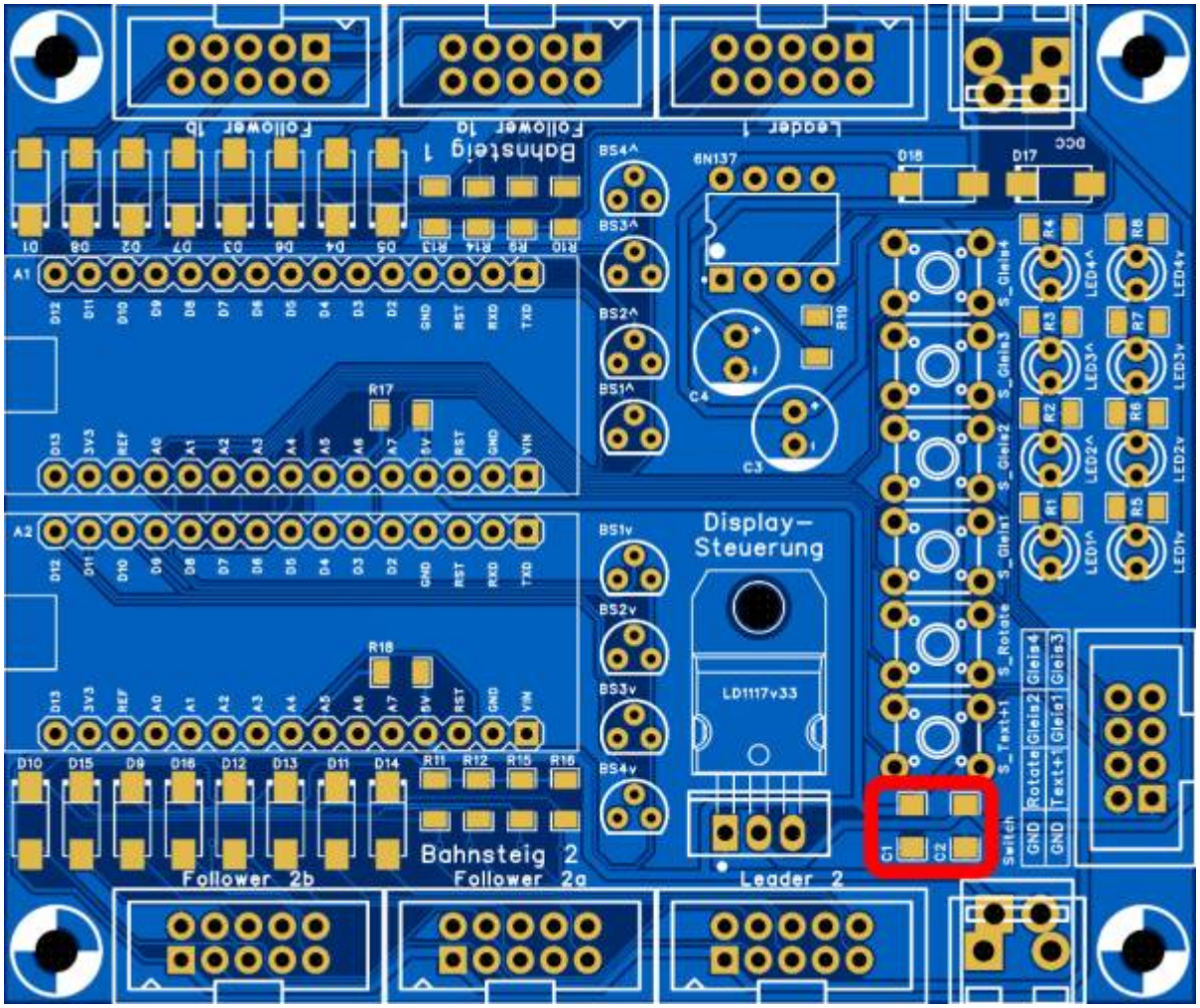
2) Widerstände R9 bis R16 und R19 (rot) sowie Widerstände R17 und R18 (grün)



3) Widerstände R1 bis R8

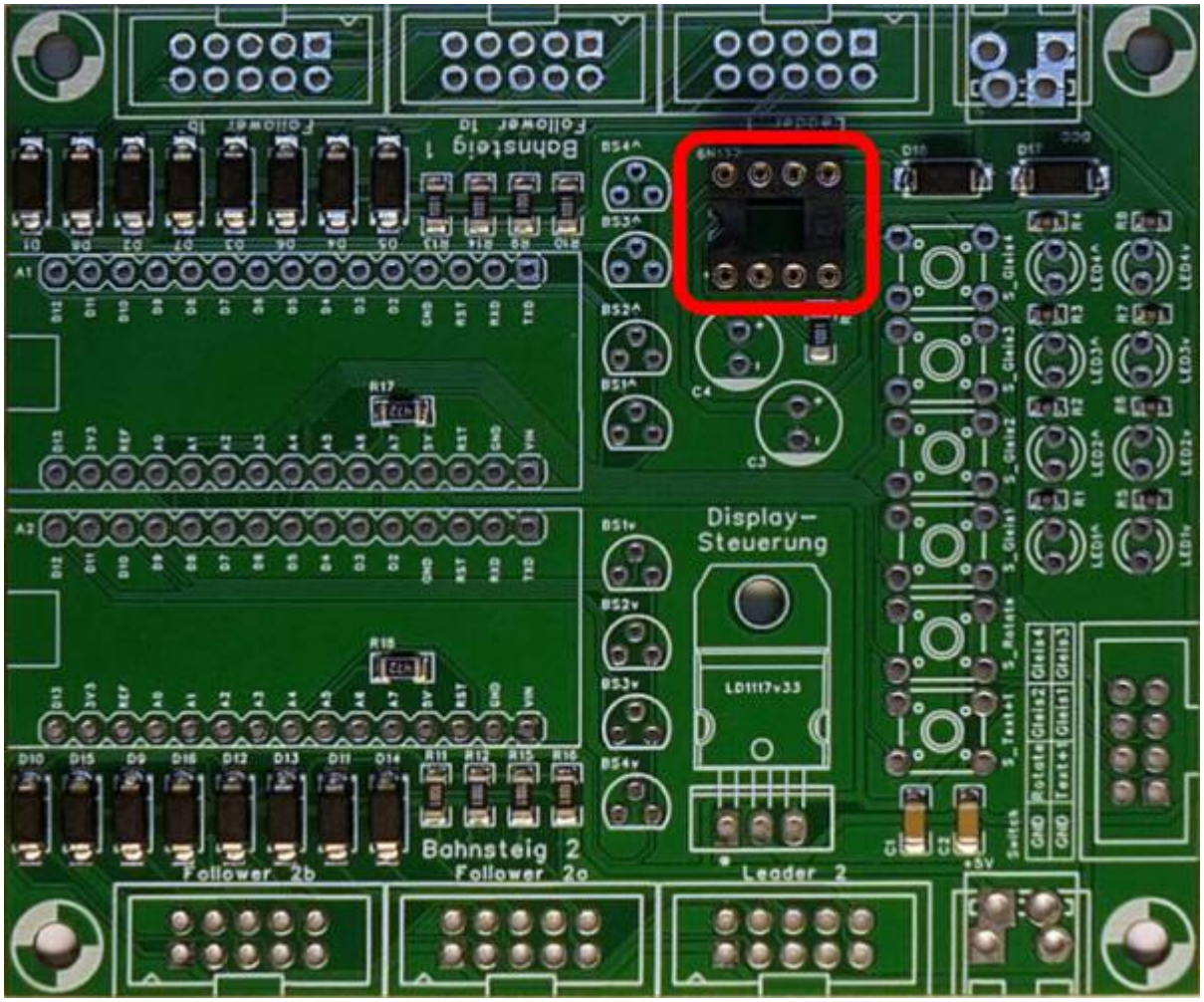


4) Keramikkondensatoren C1 und C2

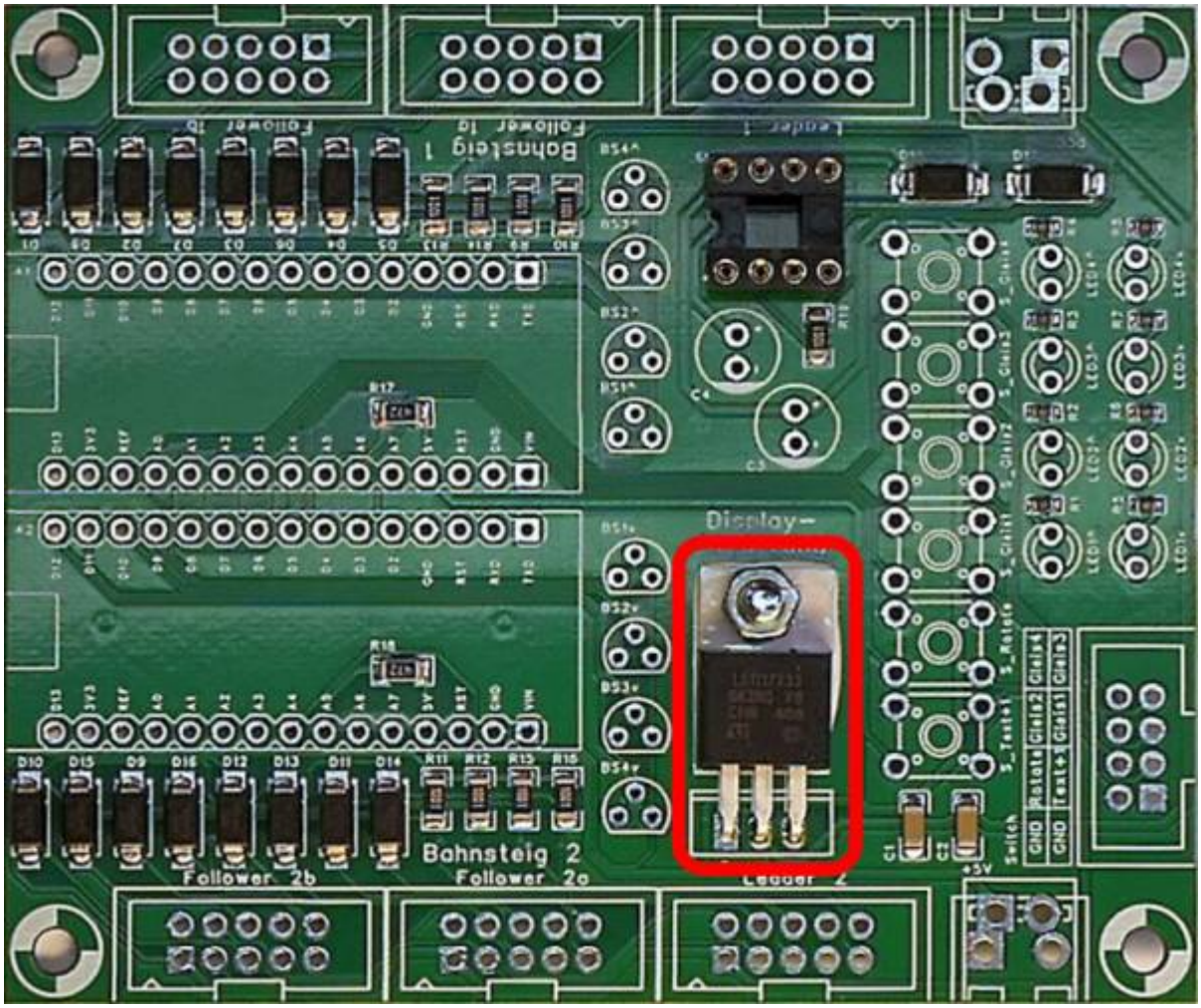


Nun geht es mit der Bestückung der THT-Bauteile weiter:

5) IC-Fassung für Optokoppler 6N137

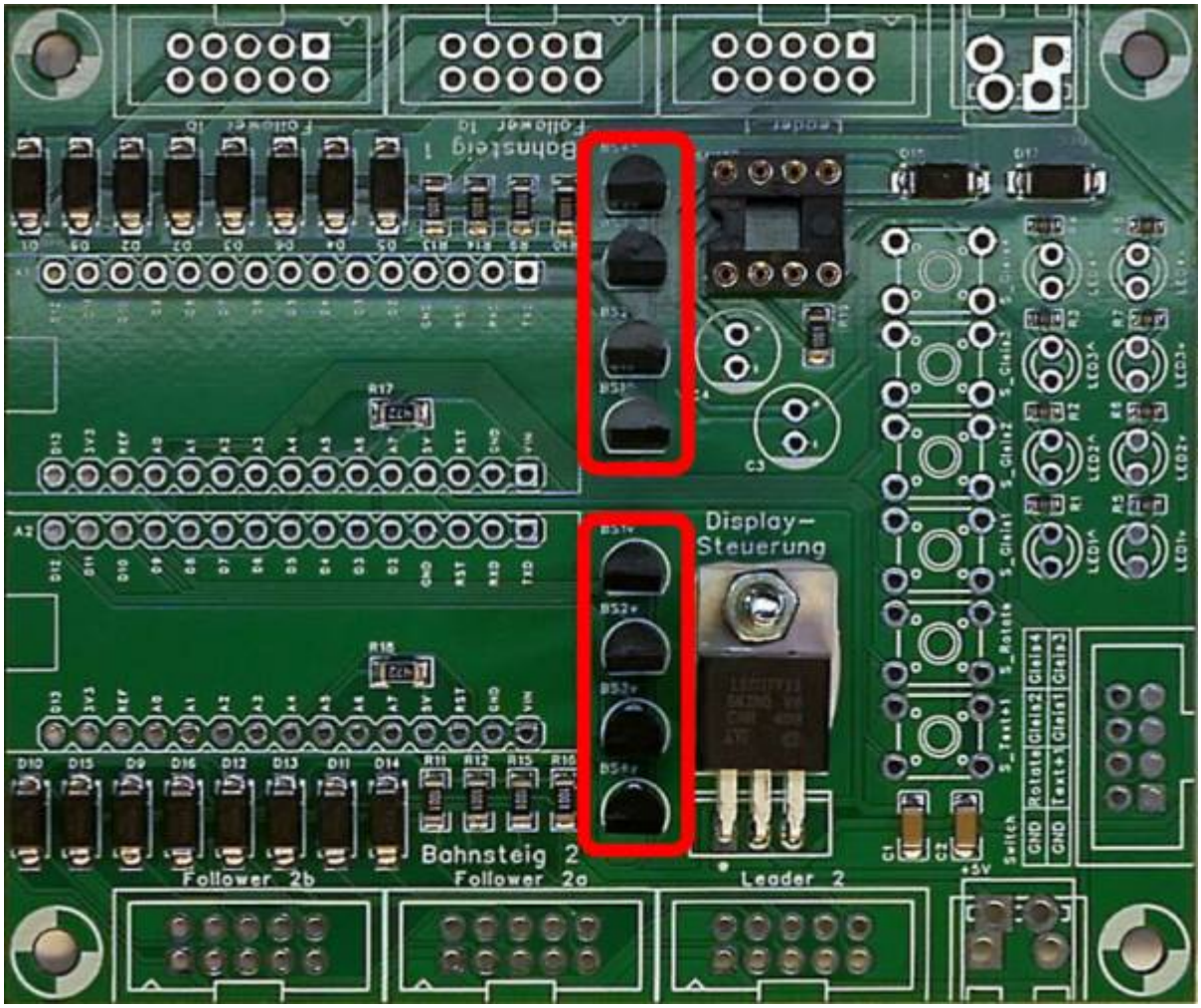


6) Spannungsregler LD1117v33 mit Wärmeleitpaste, Schraube M3x5 und Mutter M3 anbringen, dann löten

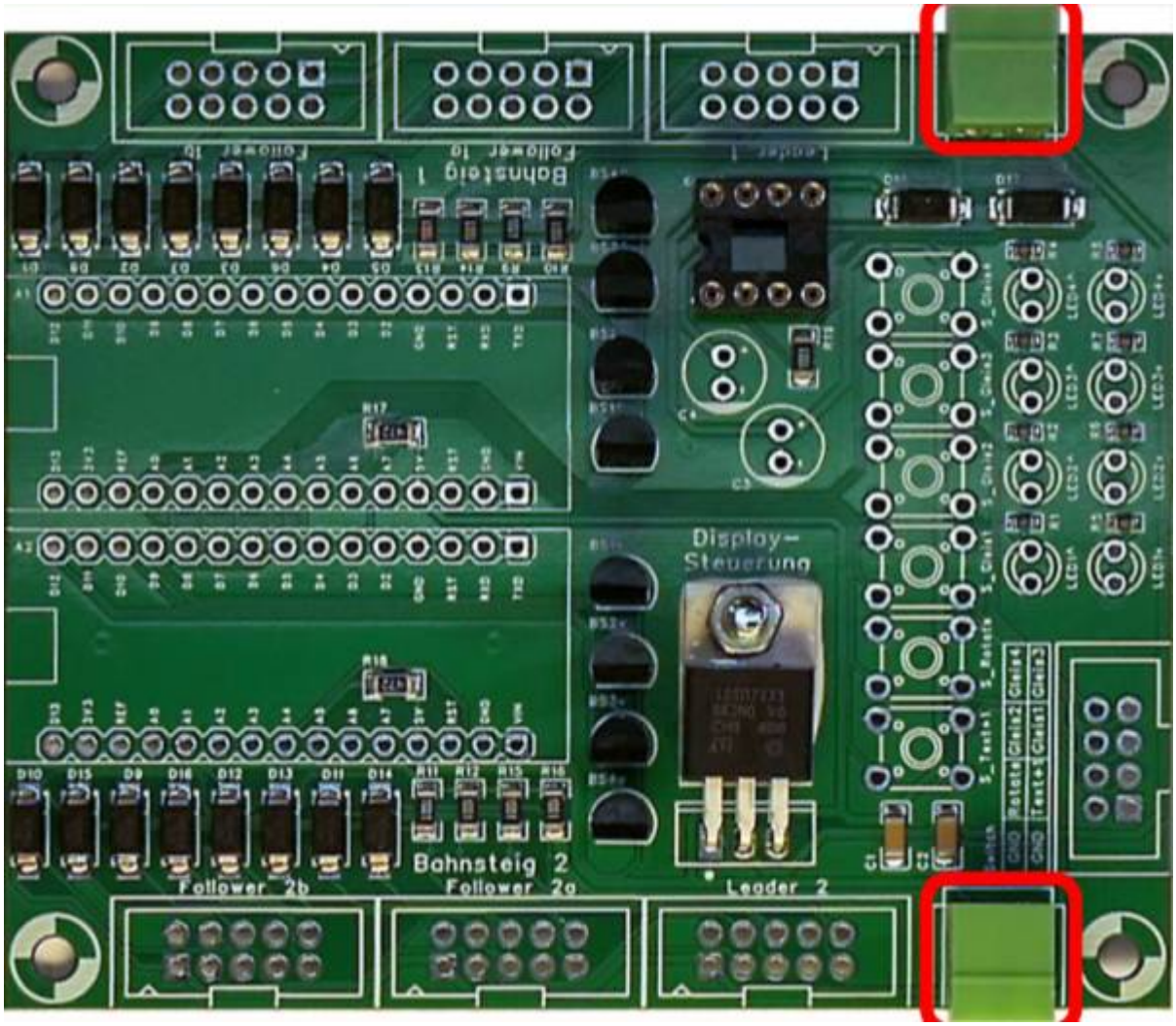


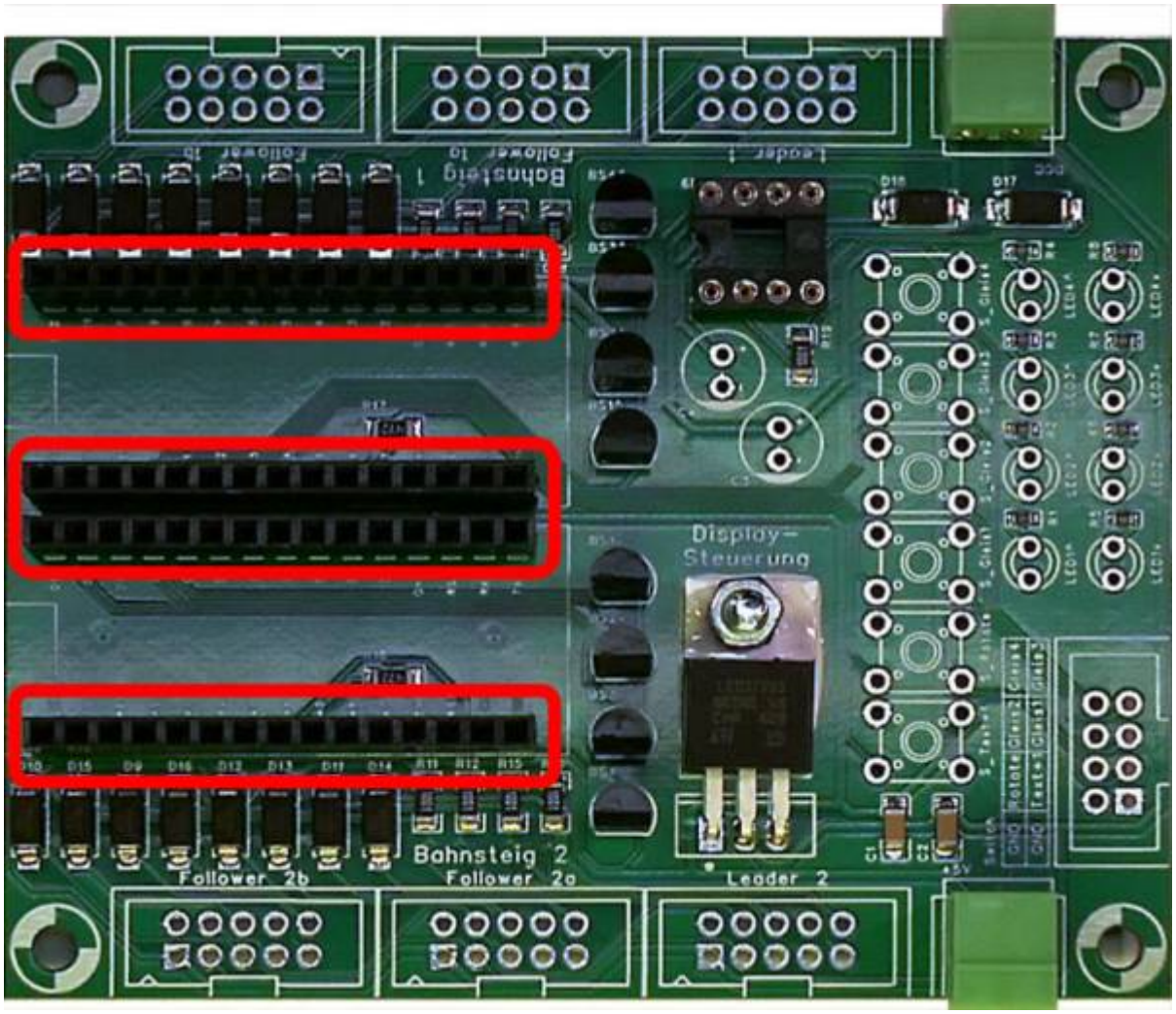
7) Mosfets BS170

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.

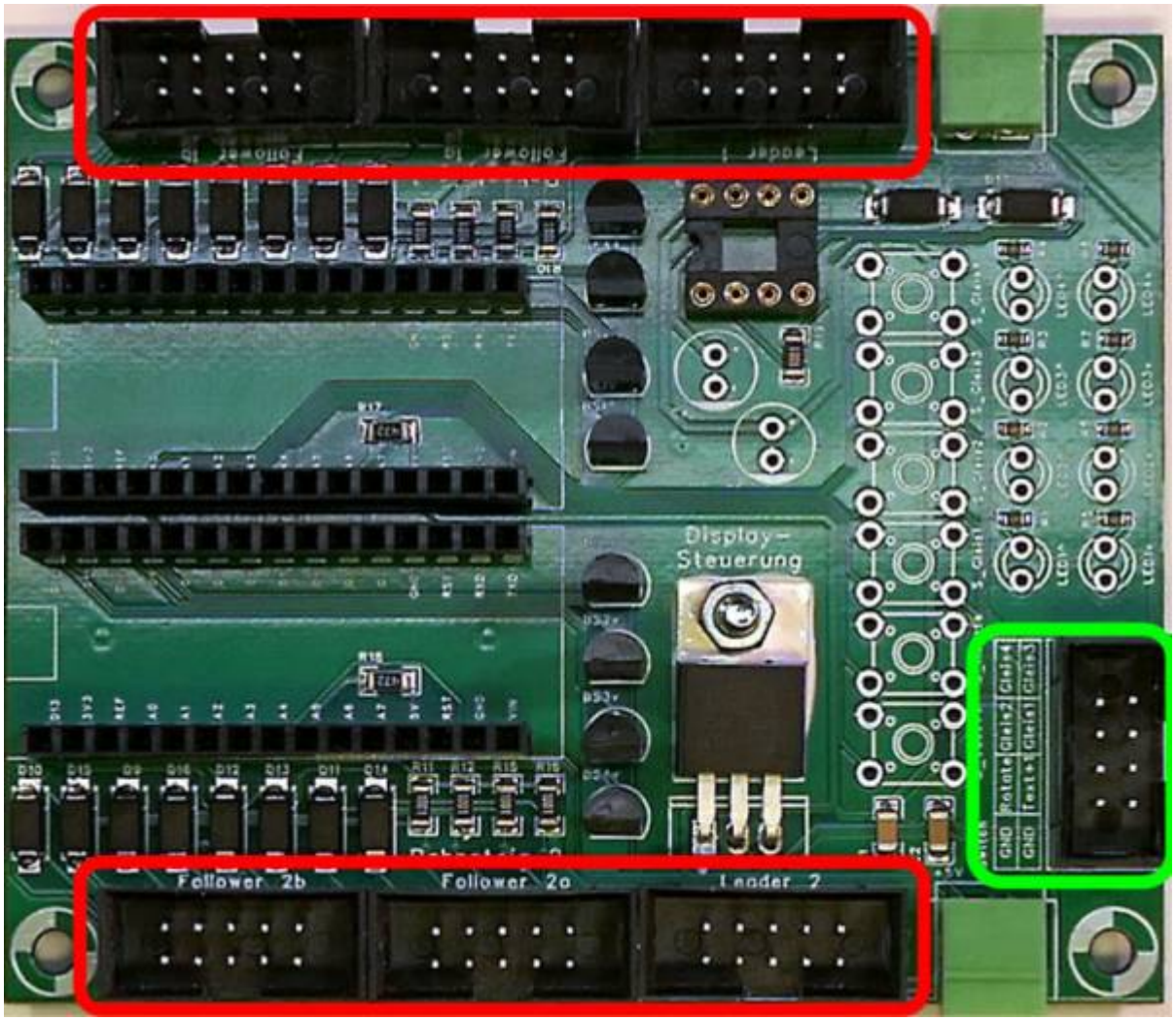


8) Anschlussklemmen +5V und DCC (Bild 1) und Buchsenleisten A1 und A2 (Bild 2)

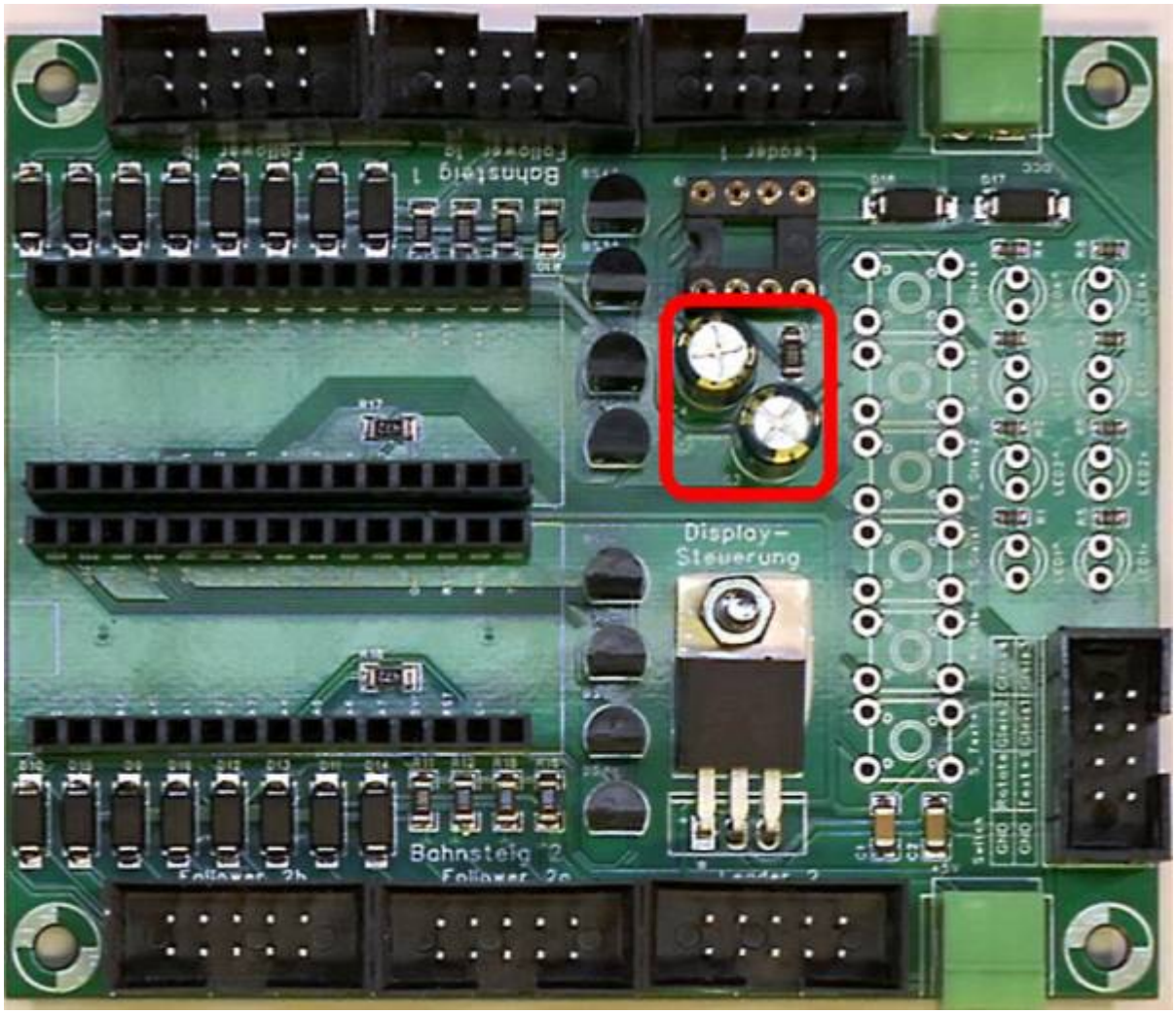

Display-
Steuerung
Bahnsteig 2
Follower 2b
Follower 2a
Leader 2



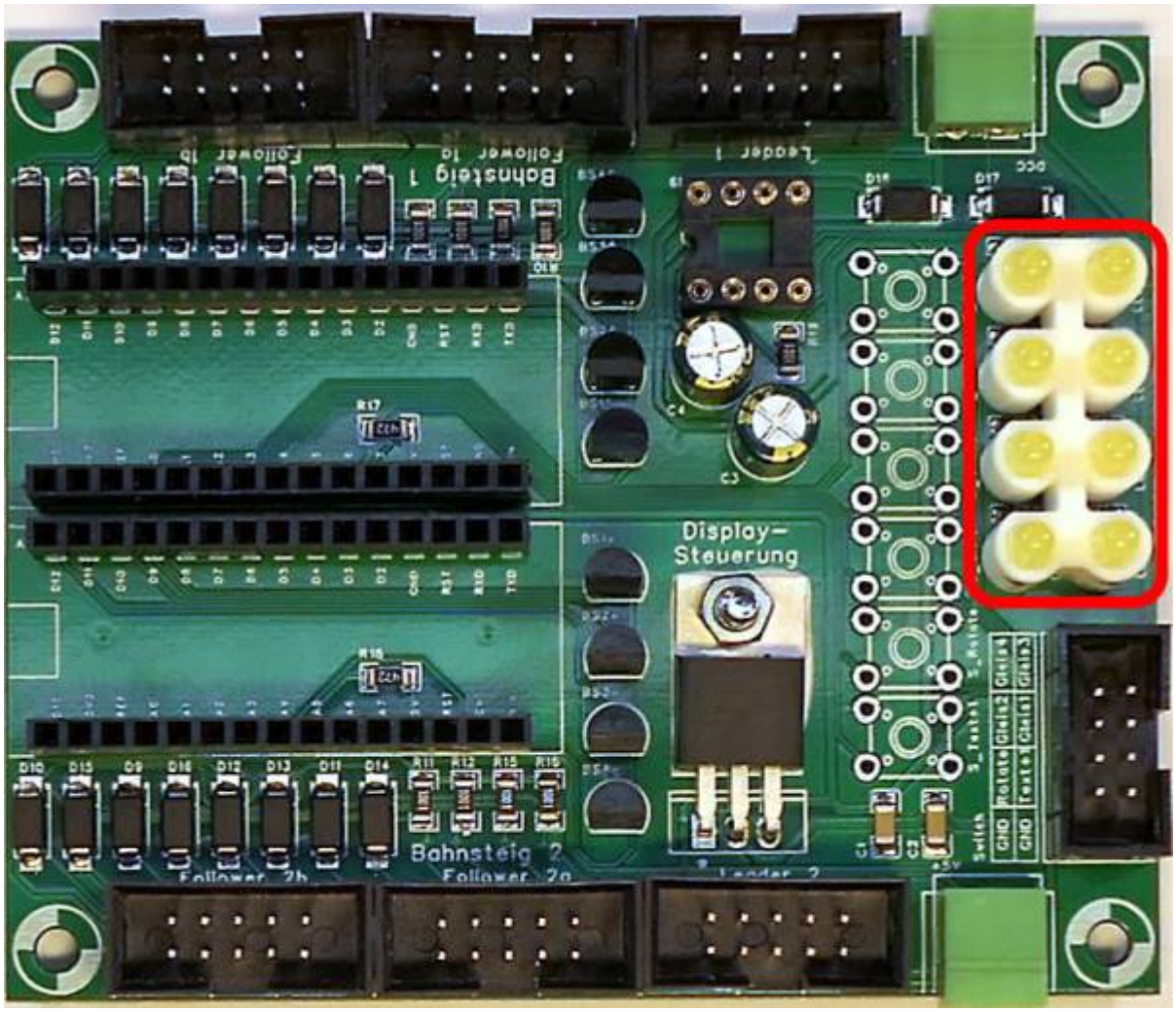
9) 10-polige Wannenstecker Leader + Follower (rot), 8-poliger Wannenstecker Switch (grün)



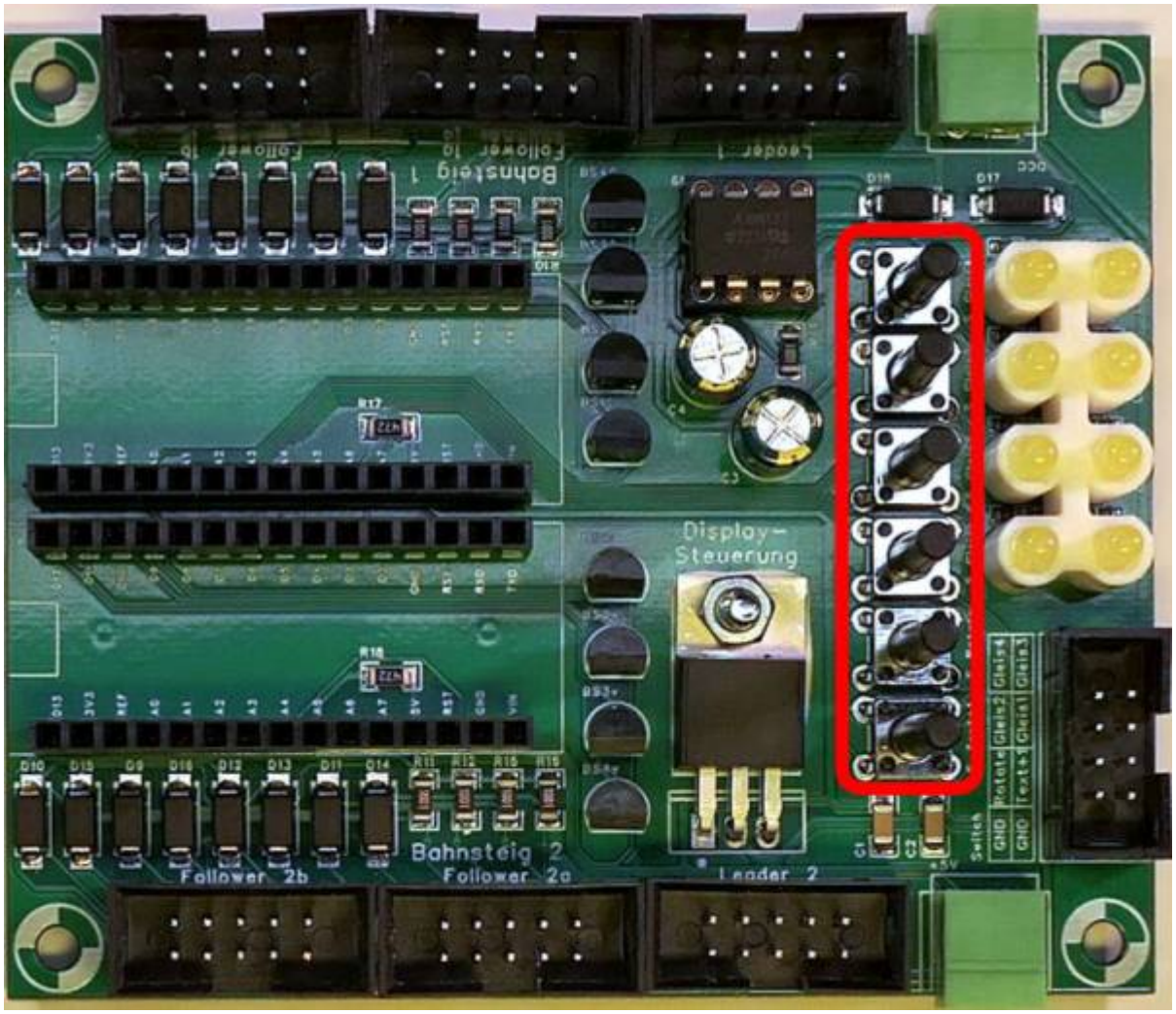
10) Kondensatoren C3 und C4 (Bild 1), LEDs 1-8 (Bild 2) inkl. 3D Abstandshalter



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.



11) Taster



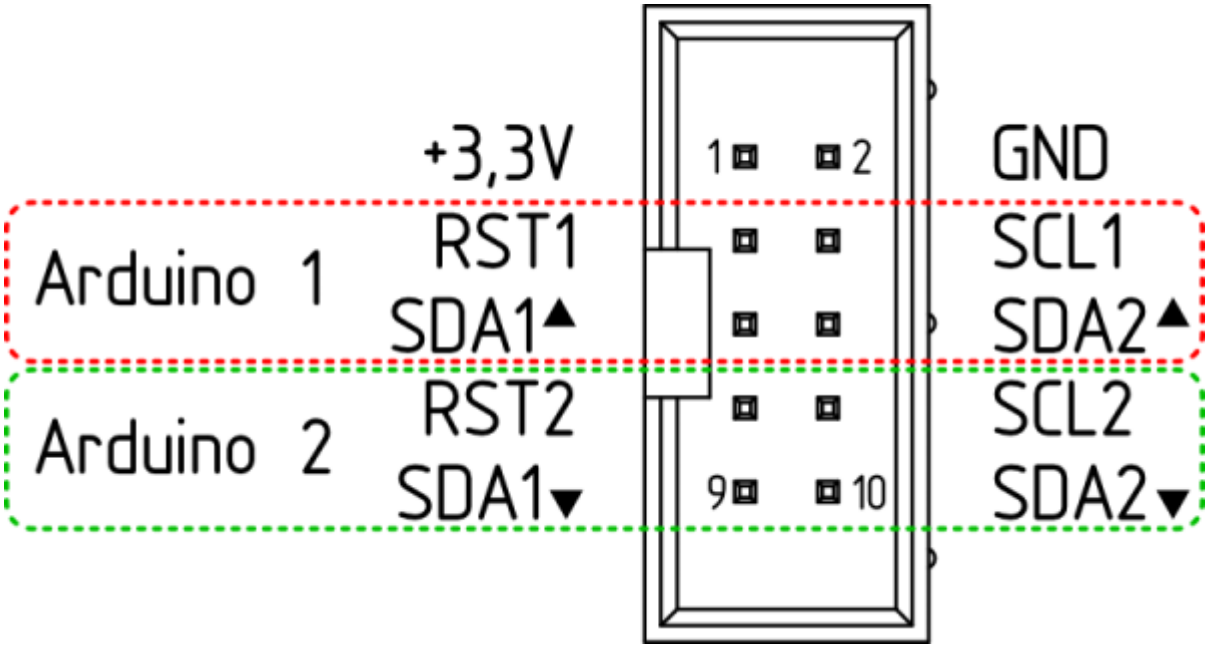
Vielen Dank an Frank für die Bereitstellung der einzelnen Bilder der jeweiligen Bauschritte!

Der erste Funktionstest

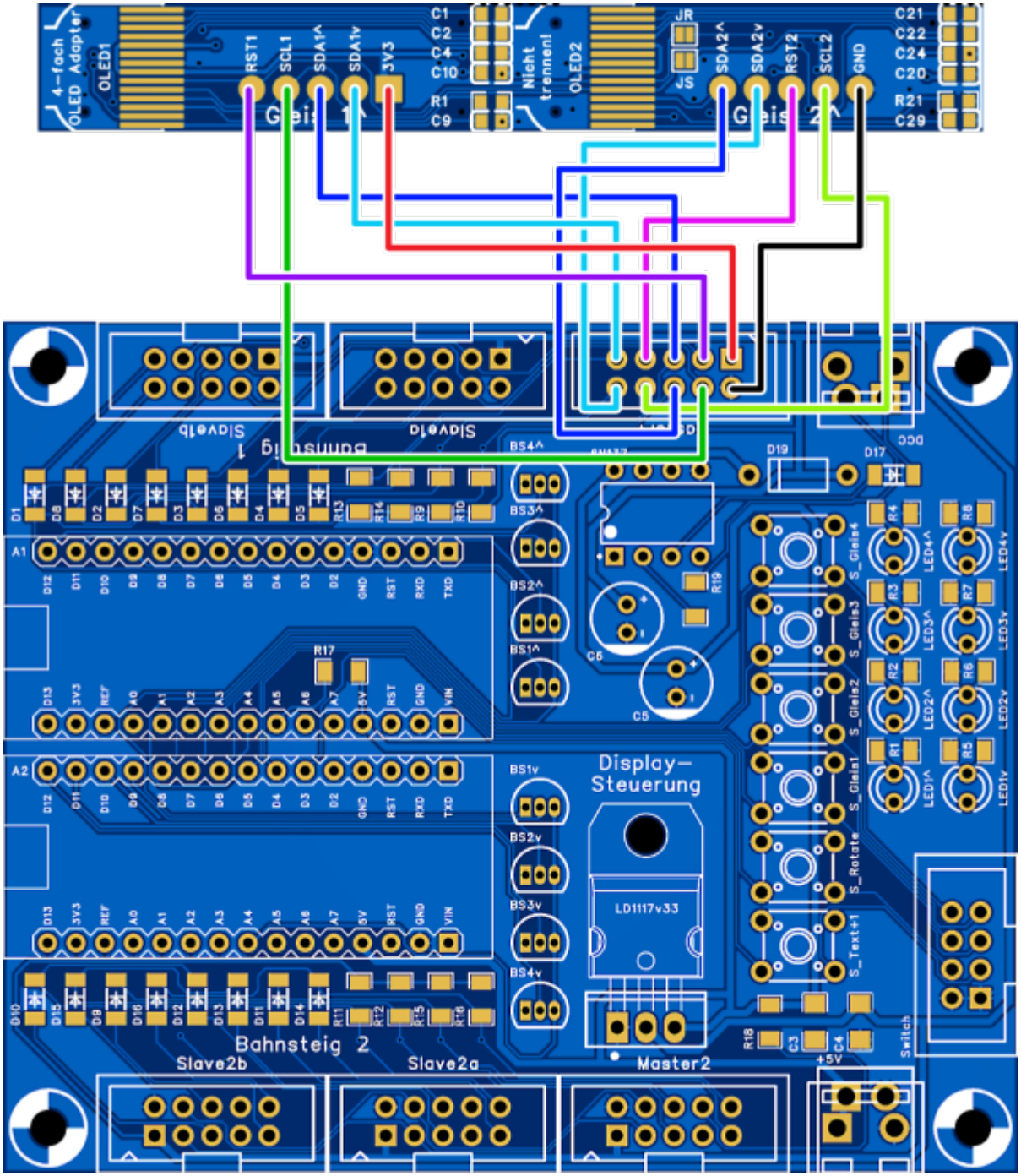
Noch bevor die OLEDs abgeschlossen werden, sollte an dieser Stelle der erste Funktionstest durchgeführt werden. Das Programm schaltet nach dem Anlegen der Versorgungsspannung die acht LEDs nacheinander an, beginnend mit dem Pärchen für Gleis 4, dann 3, 2 und schließlich 1. Die beiden LEDs für Gleis 1 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

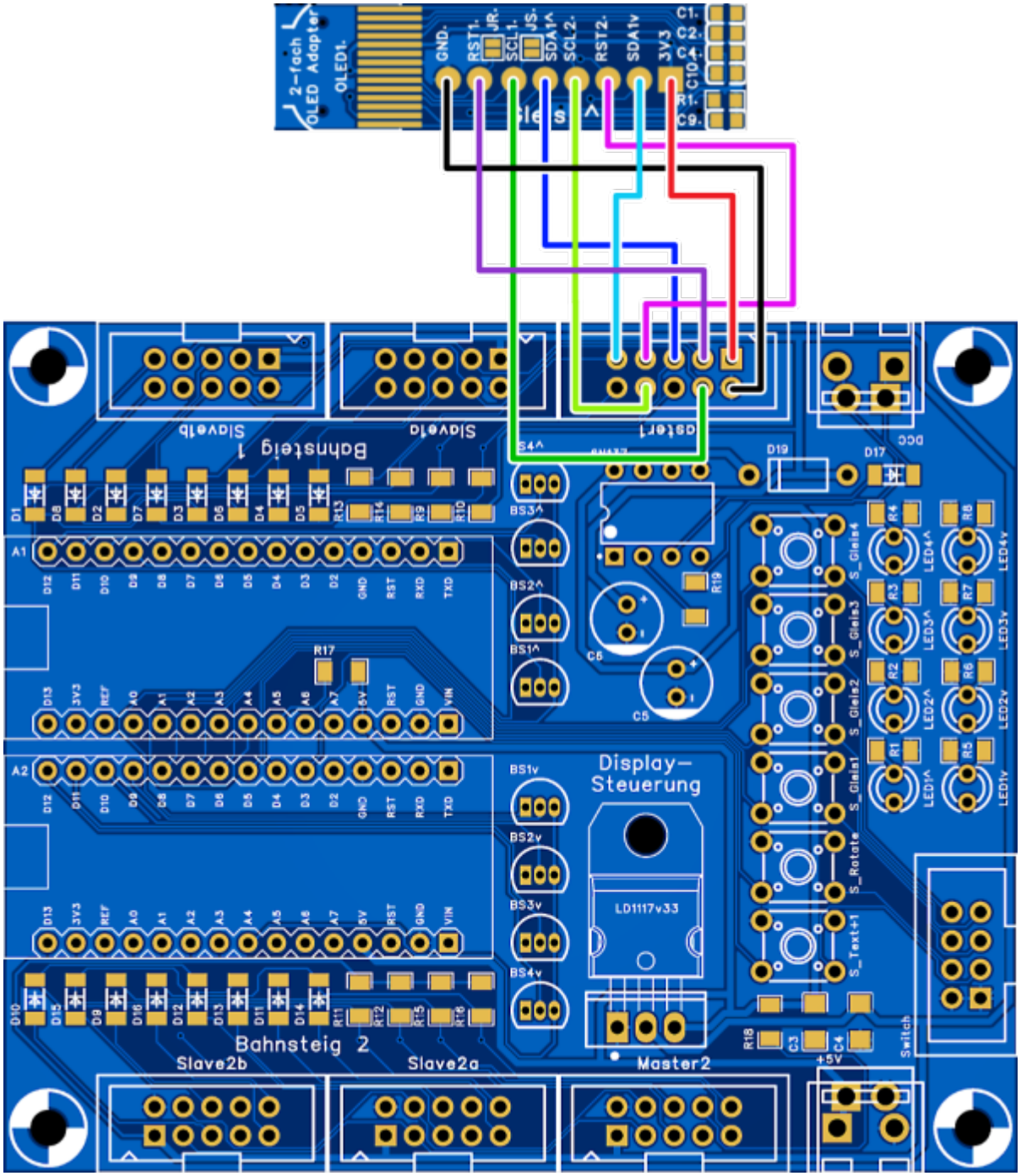


Zweigleisiger Zugzielanzeiger



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.

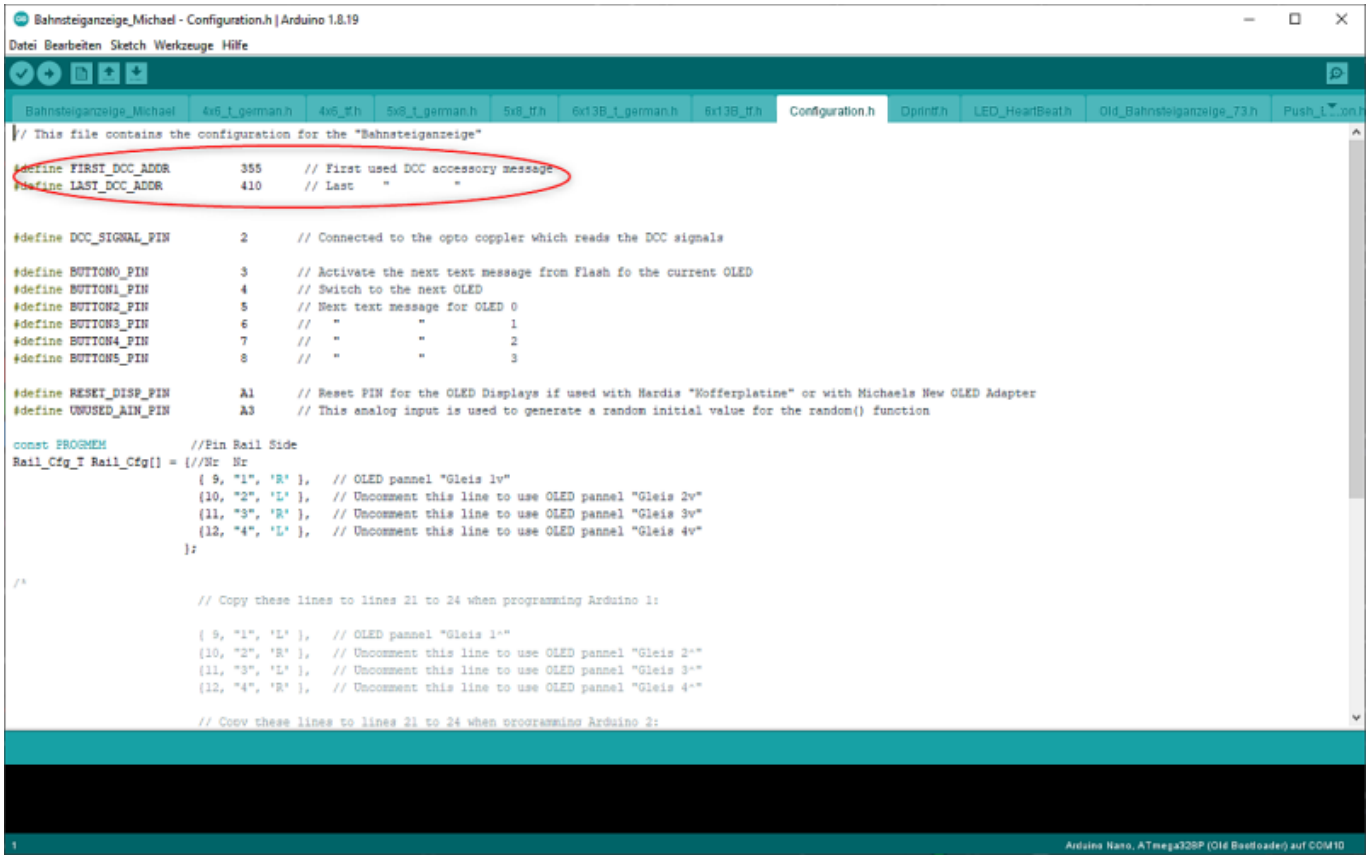
Bei der Ersteinrichtung müssen die auskommentierten Zeilen nach Wunsch aktiviert werden, indem die beiden Schrägstriche in der jeweiligen Zeile entfernt werden (1). Der Sketch ist zunächst für ein OLED Panel konfiguriert und eignet sich in der Form für einen ersten Test mit einem über ein Breadboard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

DCC Adressbereich definieren

Als erstes 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).



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

Aspekt*	DCC-Adr.	Zustand	Funktion	Bemerkungen
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. Zunächst legt man im Arduino Sketch im Reiter „TextMessages.h“ 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.

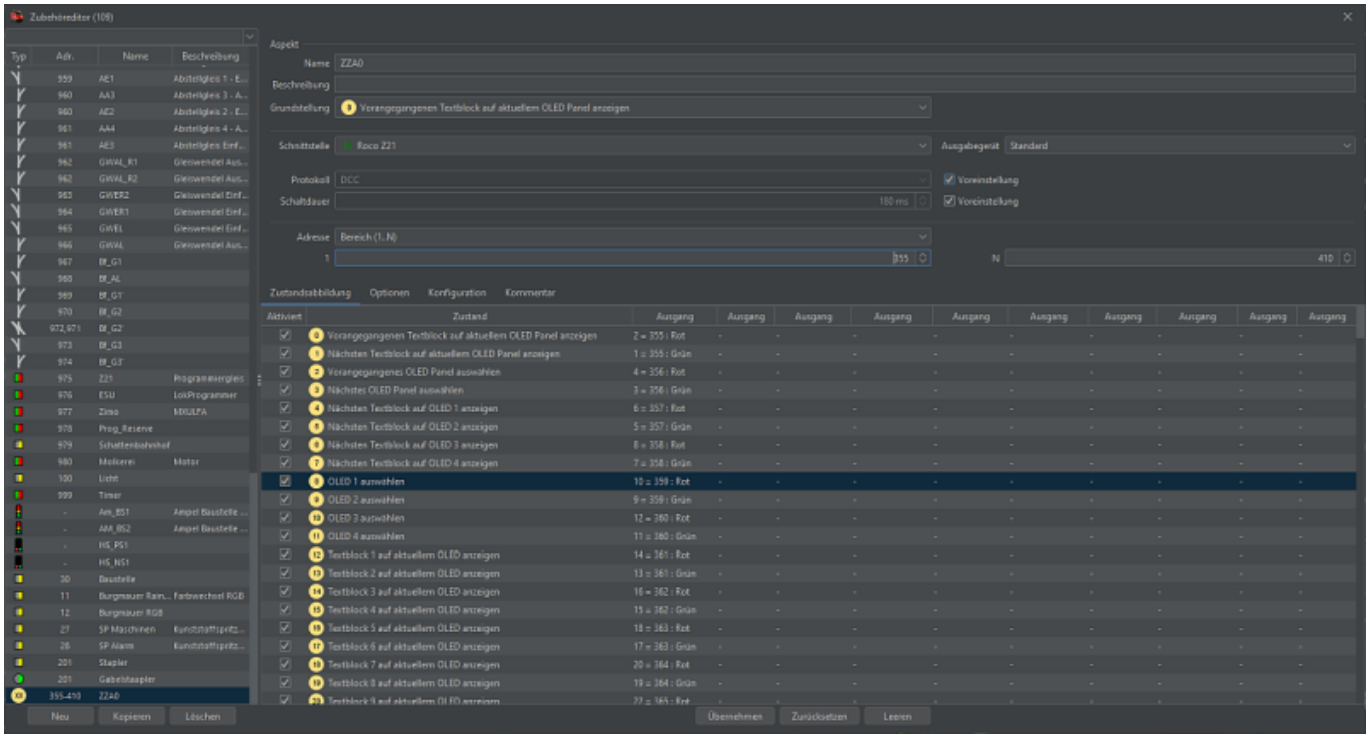
Nun geht es an die eigentliche Verknüpfung. Die Verknüpfung wird hier am Beispiel von iTrain gezeigt. Zur Erstellung einer „Wenn/Dann-Regel“ in anderen Programmen muss deren Anleitung zu Rate gezogen werden.

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 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. 6 angezeigt werden. Dazu wird zunächst die DCC Adresse n+5 auf Rot gesetzt (entspricht Aspekt 10) und mit einer kurzen Verzögerung die Adresse n+8 auf Grün (entspricht Aspekt 17).

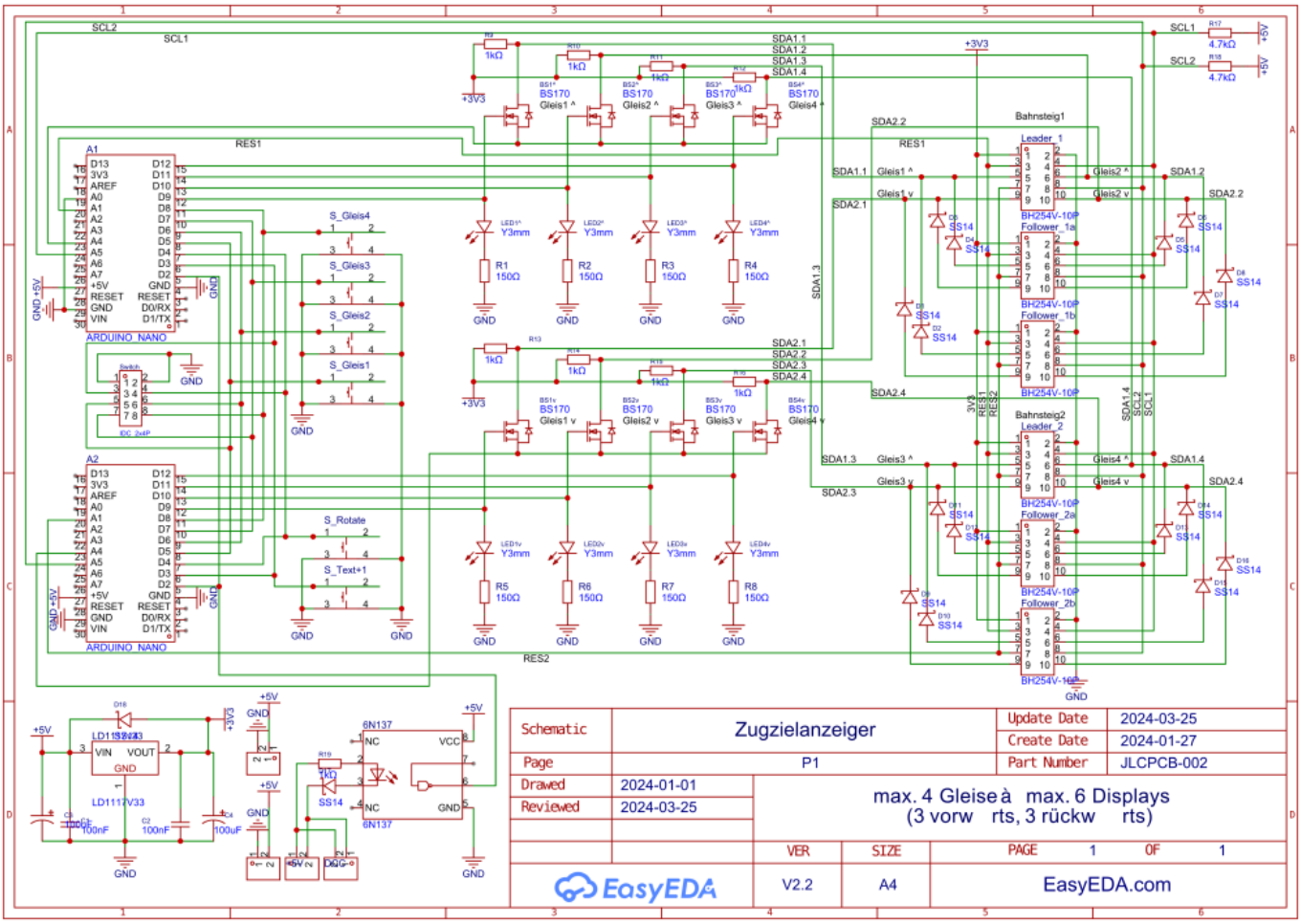
*) Der Aspekt ist eine spezielle Funktion in iTrain. Dazu wird ein beliebiger DCC Adressbereich mit maximal 128 Adressen in einen virtuellen „Drehschalter“ mit bis zu 256 Schaltstellungen verwandelt. Das macht das Senden der Display- und Textwahl einfacher. Die Programme WinDigipet, TrainController und RocRail werden ähnliche Möglichkeiten bieten, wenn auch unter anderem Namen. 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

Schaltplan



From:
https://wiki.mobaledlib.de/ - **MobaLedLib Wiki**

Permanent link:
https://wiki.mobaledlib.de/anleitungen/oled/display-steuerung?rev=1716492162

Last update: **2024/05/23 19:22**