Ansteuerung der MobaLedLib mit Tastern, Modellbahnsoftware und Smartphone App

Einleitung

Die MobaLedLib ermöglicht es Beleuchtungs-, Bewegungs- und Soundszenarien auf der Modellbahn zu steuern. Die Szenarien starten, sobald der ARDUINO eingeschaltet wird. Das heißt: Im einfachsten Fall reicht es den programmierten ARDUINO mit Strom zu versorgen und die programmierten Abläufe starten.

Das ist schon mal nicht schlecht, aber schöner wäre es, wenn man als Benutzer auch mit der Steuerung interagieren könnte und bestimmte Beleuchtungsszenarien oder Effekte ein- bzw ausschalten zu können.

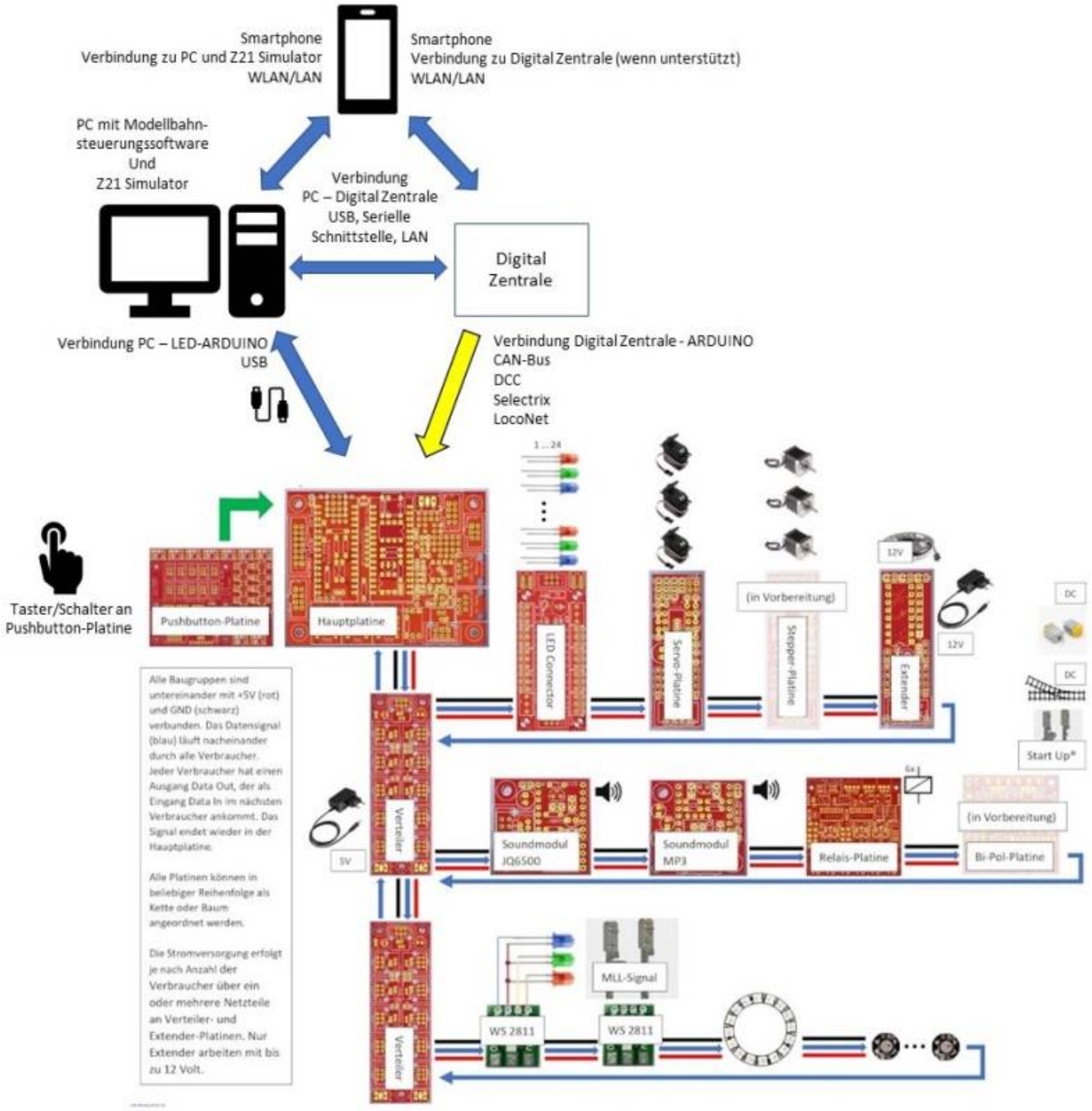
Dazu bietet die MLL mehrere Möglichkeiten:

1. Steuerung über an den MLL-ARDUINO angeschlossene Taster
2. Steuerung über eine Digitalzentrale
3. direkte Steuerung über den USB-Anschluss

Bei den Steuerungsarten 2 und 3 besteht die Möglichkeit die Szenarien auch über eine Modellbahnsoftware oder über ein Smartphone zu steuern.

Alle 3 Steuerungsarten werden in diesem Kapitel vorgestellt, wobei der Fokus auf der Steuerung über eine Modellbahnsoftware oder einer Smartphone App liegen wird.

Eine Übersicht über alle möglichen Steuerungsarten zeigt das folgende Bild:



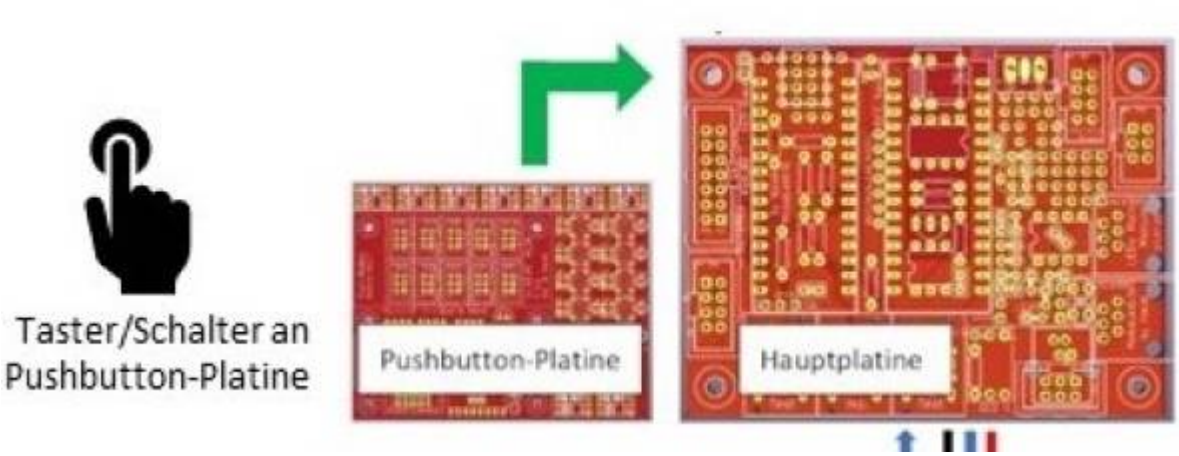
Steuerung über an den MLL-ARDUINO angeschlossene Taster

Mit der MobaLedLib kann man Schalter oder Taster auf vier verschiedene Weisen einlesen. Schalter ist eigentlich ein Sammelbegriff der Kippschalter und Tastschalter umfasst. Der Kippschalter ist ein Schalter der seine Position beibehält. Er ist An- oder Ausgeschaltet. Ein Taster besitzt eine Feder mit der er in die Ruheposition zurück springt, wenn er losgelassen wird. Der Kontakt ist hier nur so lange geschlossen wie der Taster betätigt wird. Hier bezieht sich „Schalter“ in der Regel auf beide Varianten. Von den vier verschiedenen Varianten zum einlesen von Schaltern können drei Kippschalter und Taster einlesen. Dabei ist es egal wie viele Schalter gleichzeitig aktiv sind. Bei der ersten und einfachsten Variante kann immer nur ein Taster (innerhalb einer Gruppe) gleichzeitige

erkannt werden. Wenn mehrere Taster betätigt werden falschen Taster zurückgemeldet. Darum können mit diesem Verfahren auch nur Taster eingelesen werden. Bei den ersten drei Verfahren können mit wenigen Leitungen viele Schalter ausgewertet werden. Das ist zum einen wichtig, weil der Arduino nur sehr wenige Anschlüsse hat. Zum anderen soll die Verkabelung der Eingabeelemente möglichst einfach sein.

Die erste und zweite Methode ist für das einlesen von Tastern am Anlagenrand gedacht. Damit können die aus dem Miniatur Wunderland bekannten „Knopfdruck Aktionen“ realisiert werden. Bei der Methode B sind die Beleuchtungen der Taster bereits mit vorgesehen. Grundsätzlich können aber alle Varianten mit beleuchteten Schaltern betrieben werden. Die LEDs der Schalter können dazu genutzt werden den aktuellen Zustand er Funktion über Farben oder Blinkmuster zu visualisieren. Alle vier Verfahren können gleichzeitig benutzt werden. Die Anzahl der verwendeten Schalter kann fast beliebig erweitert werden. So ist die Auswertung von über hundert Schaltern problemlos möglich. Alle Schalter können komfortabel mit dem Programm_Generator benutzt werden. Dazu wird einfach der Name des Schalters in der Eingangsspalte der Tabelle eingetragen. Ein Schalter kann auch mehrere Aktionen auslösen. Ein Schalter kann mehrfach bei verschieden Funktionen verwendet werden.

Anstelle von manuell betätigten Schaltern können auf diese Weise auch Ereignisse oder Zustände auf der Moba eingelesen werden. So kann man z.B. über einen Reed-Kontakt, oder eine Lichtschranke erkennen, wenn ein Zug an einer bestimmten Position ist. Wichtig ist, dass die Ereignisse einen potentialfreien Ausgang haben. Das ist bei einem Reed-Kontakt oder einem Relais immer der Fall. Bei anderen Schaltungen lässt sich die Potentialtrennung einfach über einen Optokoppler oder ein Relais nachrüsten.



Die verschiedenen Methoden:

1. **Analoge Taster**: Die einfachste Variante ist die Kodierung von Tastern über verschiedene Widerstände. So können über zwei Leitungen und einen analogen Eingang des Arduinos 10 Taster eingelesen werden. Werden mehr Taster benötigt können weitere analoge Eingänge verwendet werden. Auf diese Weise könnten bis zu 80 Taster erfasst werden.
2. **Taster (oder Kippschalter) am Anlagenrand**: Über eine PushButton_4017 Platine können 10 Taster eingelesen werden. Dabei können gleichzeitig betätigte Schalter erkannt werden. Mehrere dieser Platinen können entlang der Anlagenkante platziert werden. Dadurch kann die Anzahl der Taster erhöht werden und die Leitungen zwischen Tastern und Platine kurzgehalten werden. Die Anzahl der möglichen Taster ist (eigentlich) nicht begrenzt. Es können über 100 Taster eingelesen werden. Als Eselsbrücke kann man sich „B“ = Border (Englisch Rand) merken.
3. **Weichenstellpult**: Wenn man viele Schalter an einer Stelle einlesen will, dann kann man die Variante C wählen. Hier können mit einer PushButton_4017 Platine 80 Schalter verarbeitet

werden. Auch hier können mehrere Platinen kaskadiert werden. Bei dieser Variante werden 8 oder mehr Schalter gleichzeitig erfasst. Dadurch wird weniger Rechenzeit benötigt.

4. **Direkt angeschlossene Schalter**: Diese Methode kommt ganz ohne zusätzliche Bauteile aus. (Eselsbrücke „D“ = Direkt). Sie nutzt die drei auf der Hauptplatine vorhandenen Taster. Es ist aber auch möglich externe Schalter über die vorhandenen Stecker anzuschließen. Die Anzahl der direkt an die Hauptplatine angeschlossenen Schalter kann auch erhöht werden. Allerdings wird hier für jeden Schalter ein eigener Pin das Arduinos belegt.

Eine ausführliche Anleitung, wie Schalter direkt an die MLL angeschlossen werden können findet Ihr hier:

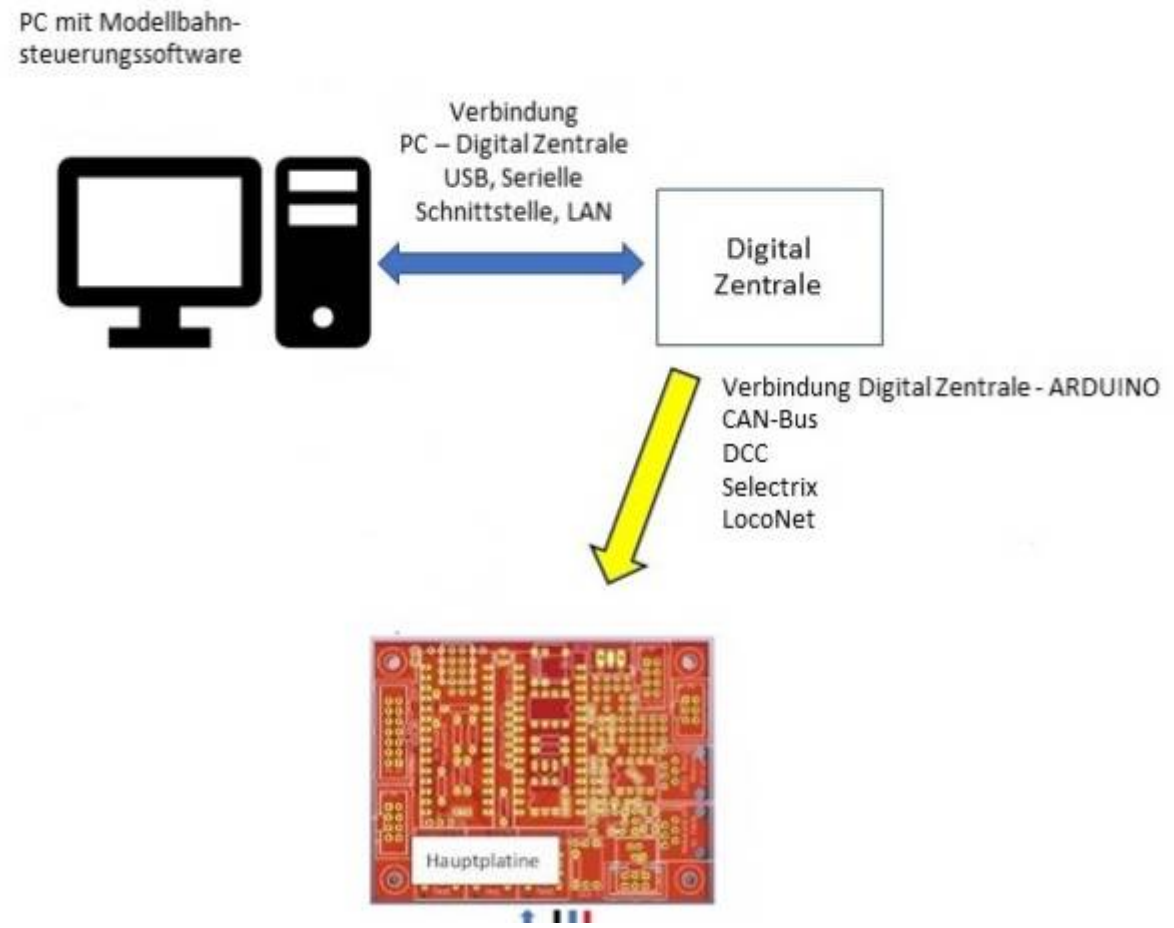
Ausführliche Anleitung

Bis jetzt können mit den Schalter nur Aktionen in der MobaLedLib ausgelöst werden. Die Möglichkeit die Schalterstellungen auch an eine Modellbahnsteuerungssoftware als Rückmelder zu melden ist in Vorbereitung - erste Tests waren sehr vielversprechend. Der ARDUINO muss dazu aber immer direkt per USB mit dem PC verbunden sein.

Steuerung über eine Digitalzentrale

Für die Steuerung der Beleuchtung über eine Digitalzentrale bietet die MobaLedLib mehrere Schnittstellen an:

1. CAN-Bus
2. DCC
3. Selectrix (HW auf Platine ab Version 1.5, vorher mit Zusatzplatine)
4. Loconet (HW auf der Basisplatine bis Version 1.4, aber bis jetzt nicht getestet)



Weiter unten sind einige Beispiele für die Steuerung der MLL über verschiedene Digitalzentralen zu finden. Beispiele für MLL mit Digitalzentralen

CAN-Bus

Die Märklin Digital Zentralen CS2 und CS3 verwenden einen CAN-Bus zur Kommunikation zwischen verschiedenen Geräten. Die MobaLedLib unterstützt den CAN-Bus mit der Hauptplatine 100. Die notwendige Bestückung der Hauptplatine ist hier beschrieben:

Hauptplatine Version 1.6 & 1.7 - Grundversion für MCAN-Bus (DE)

Zum Programmieren muss im Programmgenerator dass Tabellenblatt „CAN“ verwendet werden. Die CAN-Adresse mit der ein Makro geschaltet werden soll, wird in der Spalte „D“ - „Adresse oder Name“ eingetragen. Das Makro kann dann in der CS2 oder CS3 als Schalter oder Weiche oder Signal mit der gewählten CAN-Adresse eingetragen werden.

DCC

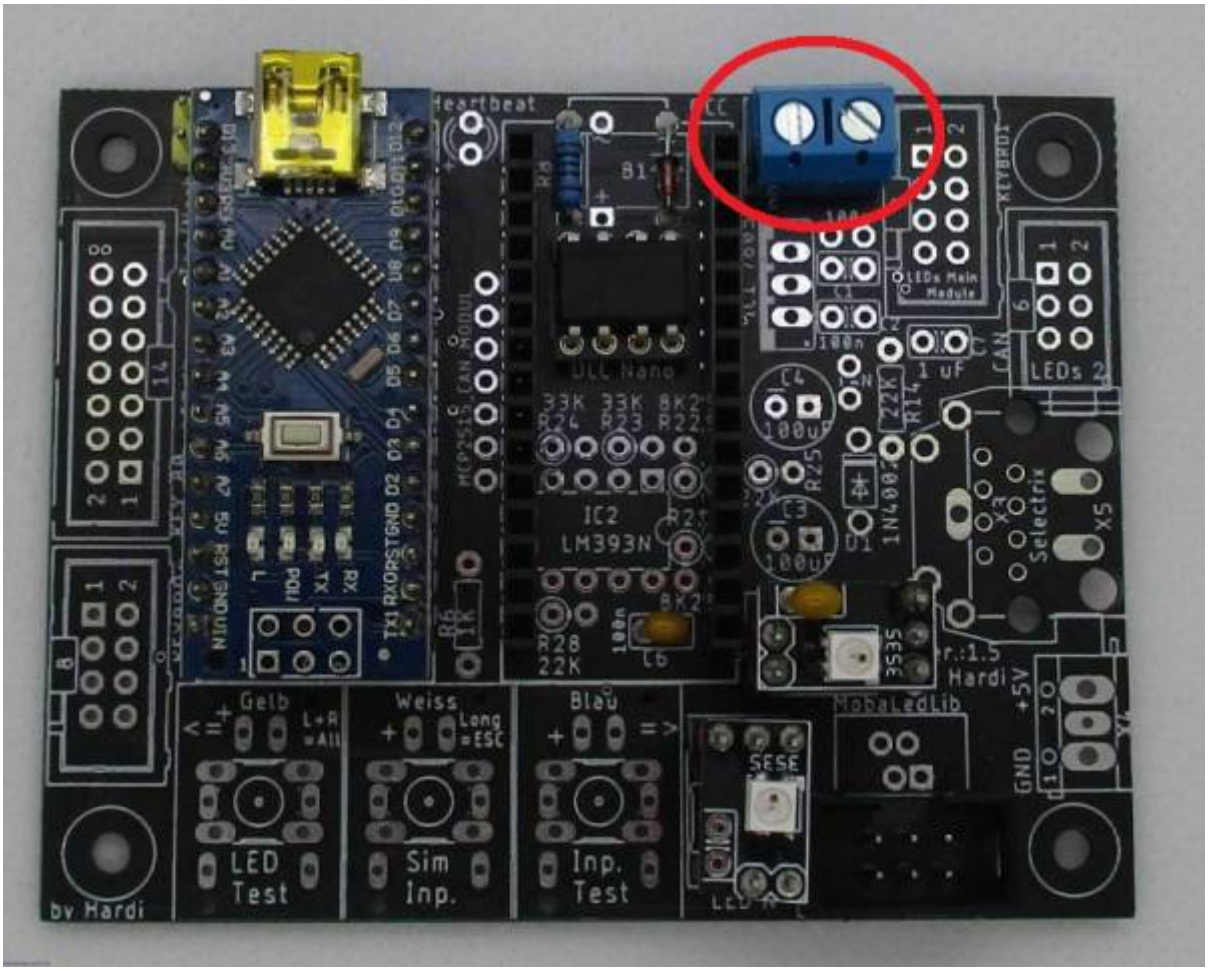
Alle modernen Digital Zentralen können das DCC Protokoll erzeugen. Mit diesem Protokoll werden die Lok- und Zubehör-Decoder angesteuert. Die Digital Zentralen haben dafür einen sogenannten „Schienen-Ausgang“, der mit den Schienen der Anlage verbunden ist. Parallel zu den Schienen oder von den Schienen abgezweigt werden die Zubehör-Decoder an diese Leitung angeschlossen.

Die MobaLedLib Basisplatine 100 kann ebenfalls an diese Leitung angeschlossen werden und auf DCC

Weichensteuerbefehle reagieren. Die notwendige Bestückung der Hauptplatine ist hier beschrieben:

Hauptplatine Version 1.7 - Grundversion für DCC (DE)

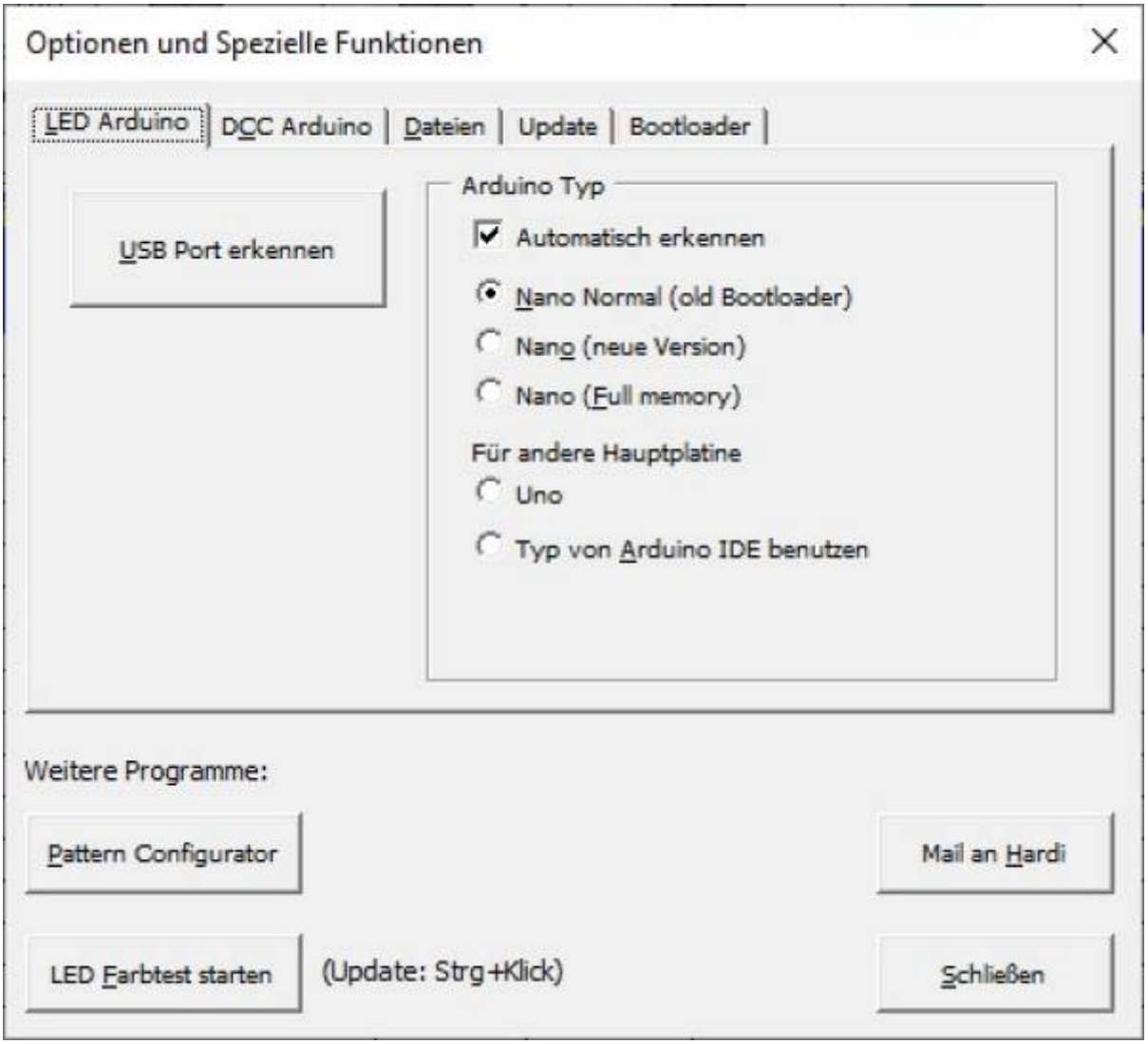
Es werden 2 ARDUINOs für diese Konfiguration benötigt. Einer steuert die LEDs und der andere überwacht das DCC-Signal von der Zentrale und schickt eine Nachricht an den LED ARDUINO, wenn eine Weichensteuerbefehl erkannt wurde. Das DCC-Signal (Schienenanschluss der Zentrale) muss an der Basisplatine an dem rot umrandeten Anschluss angeschlossen werden.



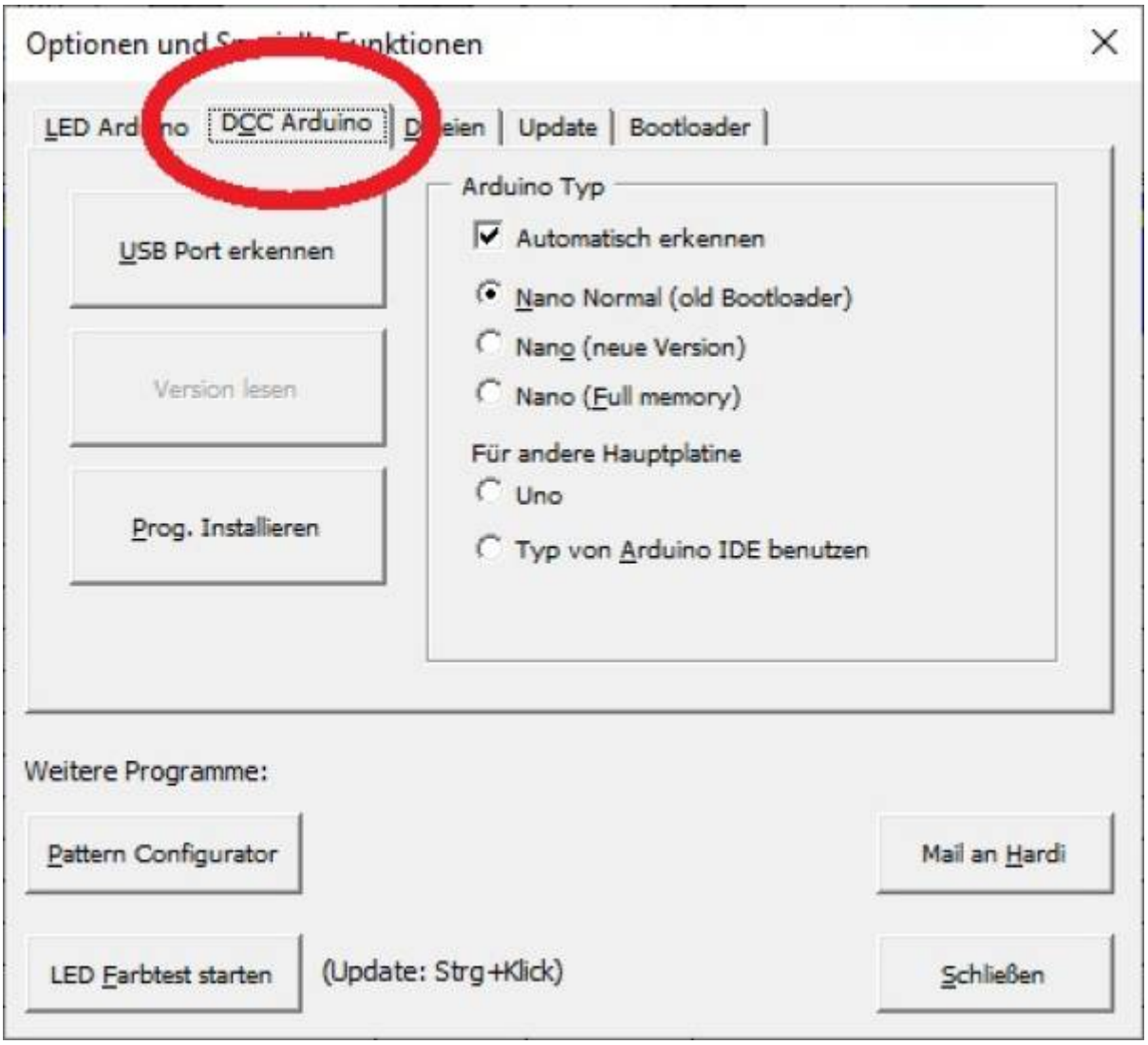
Achtung: In dem Bild ist der DCC-ARDUINO noch nicht eingesteckt worden, um die Bauteile, die unter dem ARDUINO liegen zu zeigen.

Der DCC-ARDUINO muss mit dem speziellen DCC-Programm programmiert werden. Dies geht am einfachsten mit dem Programmgenerator.

1. Nur den DCC ARDUINO mit dem USB Kabel an den PC anschließen
2. Den Excel Programmgenerator starten.
3. „Optionen“ Schaltfläche anklicken
4. Es erscheint das Optionen Fenster



5. Den Reiter DCC ARDUINO auswählen



6. „USB Port erkennen“ ausführen das Programm sucht dabei nach einem angeschlossen ARDUINO, bestimmt den Typ des ARDUINO und trägt die korrekten Daten in die Felder ein)
7. „Prog. Installieren“ anklicken. Der DCC-ARDUINO wird jetzt programmiert.
8. Danach den DCC ARDUINO wieder vom PC trennen und den LED ARDUINO mit dem PC verbinden.

Für die Programmierung muss im Programmgenerator dass Tabellenblatt „DCC“ verwendet werden. Die DCC-Adresse mit der ein Makro geschaltet werden soll, wird in der Spalte „D“ - „Adresse oder Name“ eingetragen. Das Makro kann dann in der Digitalzentrale als Schalter oder Weiche oder Signal mit der gewählten DCC-Adresse eingetragen werden.

Probleme mit DCC Adressen bei verschiedenen DCC Zentralen

Einige Hersteller und Programmierer von DCC-Zentralen und deren Software nummerieren die Weichen ab Modul 0 (mit jeweils 4 Weichen), andere DCC-Zentralenhersteller erst ab Modul 1. Diese unterschiedliche Zählweise ist historisch aus einer Schwäche der Spezifikation NMRA S-9.2.1 gewachsen, wobei keine der beiden Zählweisen grundsätzlich als „falsch“ bezeichnet werden konnte.

Details zu diesem Problem und der Behebung findet Ihr hier:

Probleme bei DCC

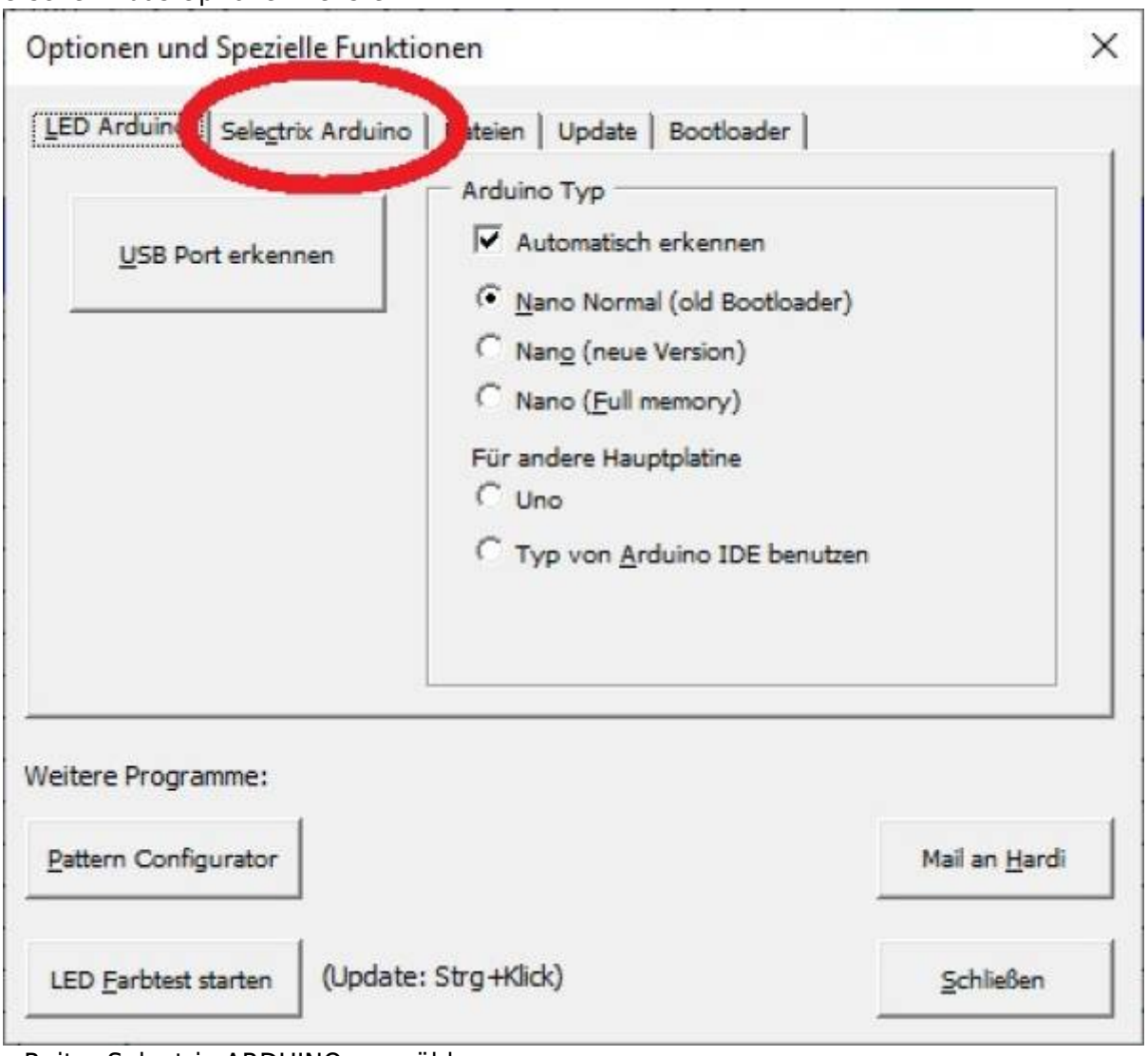
Selectrix

Einige Digital Zentralen können das Selectrix Protokoll erzeugen. Mit diesem Protokoll werden die Lok- und Zubehör-Decoder angesteuert. Die Digital Zentralen haben dafür einen sogenannten SX-Bus. Die MobaLedLib Basisplatine 100 kann ebenfalls an den SX-Bus angeschlossen angeschlossen werden und auf Selectrix Weichensteuerbefehle reagieren.

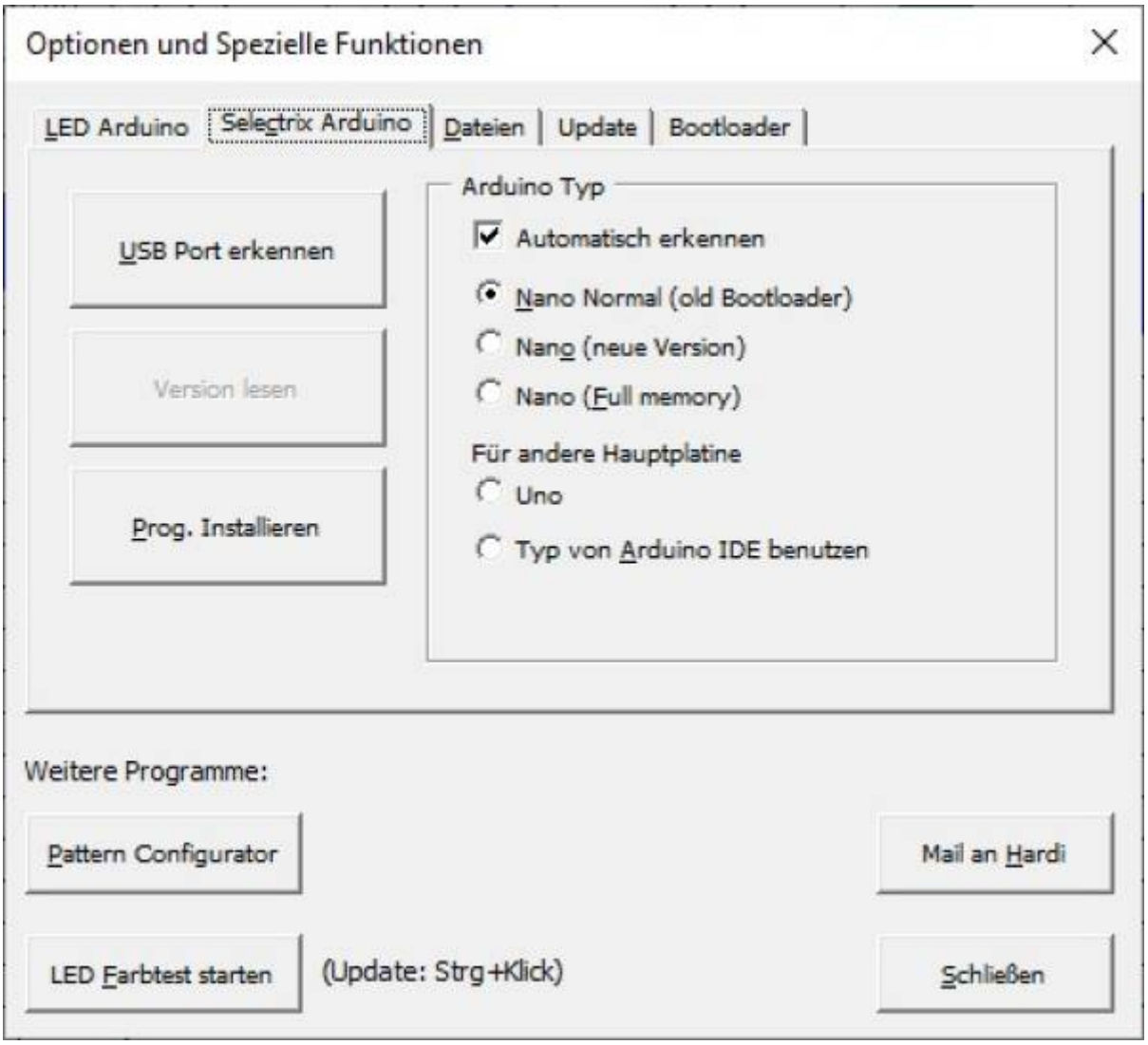
Eine Anleitung dazu findet Ihr hier: Selectrix und MobaLedLib

Es werden 2 ARDUINO für diese Konfiguration benötigt. Einer steuert die LEDs und der andere überwacht das SX-Signal von der Zentrale und schickt eine Nachricht an den LED ARDUINO, wenn eine Weichensteuerbefehl erkannt wurde. Der SX-ARDUINO muss mit dem speziellen Selectrix-Programm programmiert werden. Dies geht am einfachsten mit dem Programmgenerator.

1. Nur den SX-ARDUINO mit dem USB Kabel an den PC anschließen
2. Den Excel Programmgenerator starten.
3. Selectrix Arbeitsblatt auswählen
4. „Optionen“ Schaltfläche anklicken
5. Es erscheint das Optionen Fenster



6. Den Reiter Selectrix ARDUINO auswählen



7. „USB Port erkennen“ ausführen /das Programm sucht dabei nach einem angeschlossen ARDUINO, bestimmt den Typ des ARDUINO und trägt die korrekten Daten in die Felder ein)
8. „Prog. Installieren“ anklicken. Der Selectrix ARDUINO wird jetzt programmiert.
9. Danach den Selectrix ARDUINO wieder vom PC trennen und den LED ARDUINO mit dem PC verbinden.

Zum Programmieren muss ilm Programmgenerator dass Tabellenblatt „Selectrix“ verwendet werden. Die Selectrix-Adresse mit der ein Makro geschaltet werden soll, wird in der Spalte „D“ - „Adresse oder Name“ eingetragen. Das Makro kann dann in der Digitalzentrale als Schalter oder Weiche oder Signal mit der gewählten Selectrix-Adresse eingetragen werden.

Loconet

noch nicht freigegeben, Bei Bedarf bitte Hardi fragen.

Beispiele für MLL mit Digitalzentralen

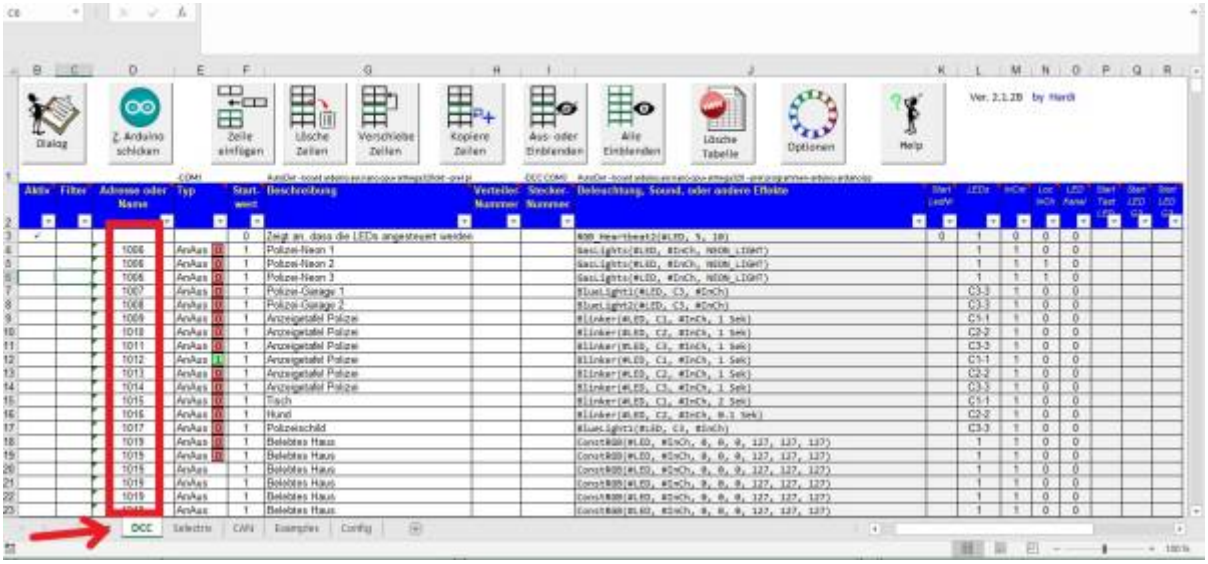
Märklin CS2/CS3 und MS2

Die Märklin-Zentralen CS2, CS3 und MS2 (mit Gleisbox) können mit MobaLedLib wahlweise über den DCC-Standard oder über den M-CAN-Bus kommunizieren. Der M-CAN-Bus ist ein Märklin-Standard, über den beispielsweise Märklin-Zentralen untereinander Daten austauschen. Sowohl über DCC wie auch über eine Anbindung per M-CAN-Bus können gleichermaßen alle MobaLedLib-Funktionen genutzt werden. Je nach gewählter Anschlussart muss die Hauptplatine aber unterschiedlich bestückt werden (siehe Bestückung DCC und Bestückung M-CAN-Bus) Die Kommunikation mit Mobaledlib ist unidirektional. Es können also nur Befehle von der Zentrale an die Hauptplatine gesendet, aber keine Rückmeldesignale von der Hauptplatine an die Zentrale zurückgeliefert werden.

Für die Kommunikation per DCC werden zwei Adern vom DCC-Anschluss der Hauptplatine mit der Märklin-Zentrale verbunden. Sie können wahlweie am Gleisausgang der Zentrale oder Gleisbox, aber auch direkt am Gleis angeschlossen werden. Die Hauptplatine unterscheidet hierbei nicht zwischen Bahnstrom (schwarz) und Masse (braun). Es ist also nicht releveant, welches der beiden Kabel an welche DCC-Anschlussklemme der Hauptplatine angeschlossen wird und entsprechend sind diese auch nicht farblich markiert. Um Digitalstrom zu sparen bietet es sich an, nur das Steuersignal von der Zentrale zu übernehmen und MobaLedLib zusätzlich mit einer eigenen Stromversorgung zu versehen. Dies kann bei neueren Hauptplatinen ab Version 1.7 über ein an die Hauptplatine angeschlossenes Netzteil geschehen oder über eine Verteilerplatine, die ebenfalls per Netzteil versorgt werden kann.



Damit MobaLedLib-Funktionen über eine Märklin-Zentrale per DCC angesteuert werden können, müssen diese im Programm-Manager als DCC-Elemente angelegt sein (roter Pfeil) und in der Spalte „Adresse oder Name“ eine DCC-Adressen erhalten haben.



In der Märklin-Zentrale werden die entsprechenden Adressen anschließend als

to be continued ...

ESU ECOS

in Arbeit

ROCO Z21

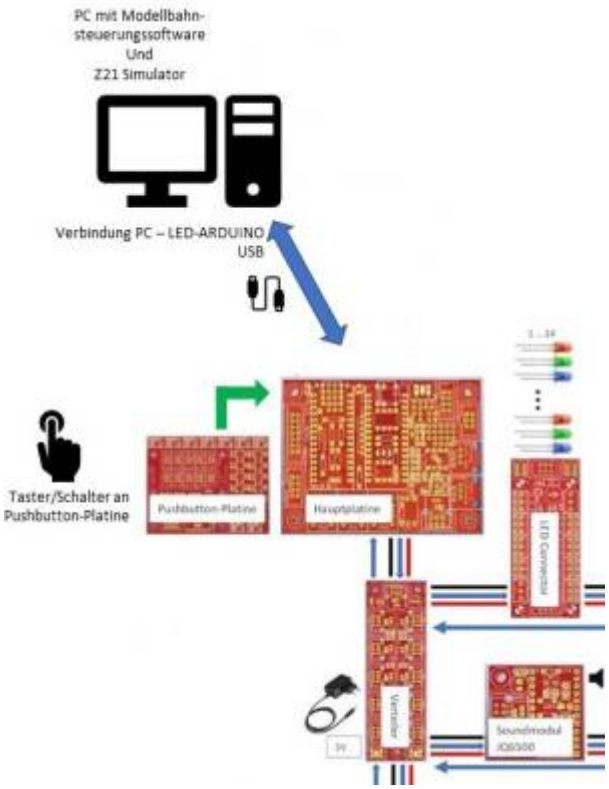
in Arbeit

Direkte Steuerung über den USB-Anschluss

Einleitung

Bei der direkten Steuerung des LED-ARDUINOs ist der ARDUINO immer mit dem USB-Anschluss des Steuer-PCs verbunden. Nicht nur zum Programmieren.

Das folgende Bild zeigt die Grundkonfiguration:



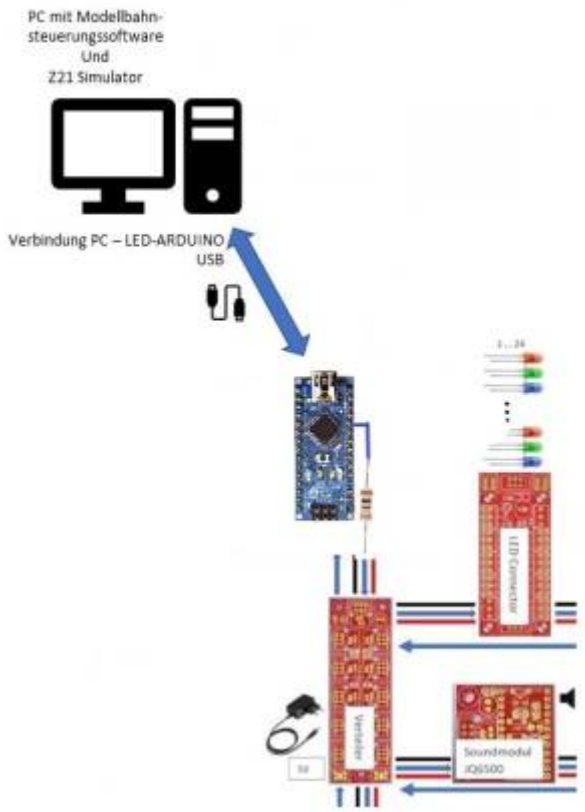
Was sind die Vorteile der direkten Steuerung?

- es reicht die minimal Konfiguration für die MobaLedLib aus - ein ARDUINO, ein 100 Ohm Widerstand und eine 4- oder 6-polige Buchse.
- der Steuerbus der Zentrale und auf dem Gleis wird nicht belastet - gerade bei DCC kann dies ein Problem werden, wenn komplexe Beleuchtungsszenarien geschaltet werden, können Steuerkommandos für Lokomotive verzögert werden. Was beim Bremsen zu ungenauen Haltepunkten führen kann.
- die Informationsübertagung ist etwas schnelle, da die Verzögerung durch die Zentrale und das Protokoll entfallen
- es ist möglich Informationen vom ARDUINO an die Software zurückzumelden, z.B. Tasterdrücke, Schalterstellungen oder Reedkontakte (in Vorbereitung)

Die Lösung hat aber auch Nachteile:

- der ARDUINO muss dauerhaft mit dem USB Ausgang des Steuer-PCs verbunden sein. Das hat folgende Konsequenzen, die man beachten muss:
  - über den ARDUINO sind alle Komponenten der MobaLedLib mit der PC Masse verbunden. Die MobaLedLib darf daher auf keinen Fall irgendeine direkte Verbindung zu den Gleisen haben, z.B. für Besetztmeldungen, diese dürfen nur über Optokoppler laufen.
  - Beim Anschließen von zusätzlichen Netzteilen für die Stromversorgung ist darauf zu achten, dass die +5V Stromversorgung des ARDUINO von der zusätzlichen Stromversorgung auch wirklich getrennt ist. Bei einem Fehler kann der USB Anschluss des PCs zerstört werden.

Das folgende Bild zeigt die minimal Konfiguration:



Wie funktioniert das?

Die Modellbahnsteuerungssoftware muss jetzt irgendwie ihre Kommandos an den ARDUINO über die USB-Schnittstelle schicken. Da keine Modellbahnsteuerungssoftware - zumindest bis jetzt nicht - die MobaLedLib kennt müssen wir das anders lösen.

Und da kommt der Z21-Simulator ins Spiel.

„Also, wat is en Z21 Simulator? Da stelle mehr uns janz dumm. Und da sage mer so: En Z21 Simulator, dat is ene jroße schwarze Box, die hat hinten un vorn e Schnittstell. Dat eine Schnittstell, dat is de Z21 Schnittstell. Und dat andere Schnittstell, dat krieje mer später.“

Die Z21-Schnittstelle

Wenn man eine Modellbahnsteuerungssoftware dazu bringen möchte, mit einer neuen Hardware zusammenzuarbeiten, ist es am einfachsten der Software eine Schnittstelle zu einer Zentrale anzubieten, die sie schon kennt. Hier gibt es mehrere Möglichkeiten. Früher wurde gerne die Intellibox-Schnittstelle dazu verwendet.

Seit ROCO seine Z21 Zentrale auf den Markt gebracht hat, die Beschreibung der Schnittstelle zu Modellbahnsteuerungssoftware veröffentlicht hat und erlaubt diese nachzubilden (was ESU z.B. ausdrücklich für die ECOS-Schnittstelle untersagt) ist die moderne Z21 Schnittstelle das Mittel der Wahl.

Wir müssen also eine Software auf dem PC laufen lassen, die die Z21-Schnittstelle simuliert und der Modellbahnsteuerungssoftware vorgaukelt sie wäre eine Roco Z21. Und diese Software ist der Z21-

Simulator.

Was die Z21 Schnittstelle so interessant macht, ist, dass sie ein LAN Protokoll ist. Das heißt, der Z21-Simulator kann auf demselben PC wie die Modellbahnsteuerungssoftware laufen oder auf irgendeinem anderen Gerät im LAN. Damit ist es auch möglich den Z21-Simulator auf einem Raspberry laufen zu lassen und den ARDUINO an die USB-Schnittstelle des Raspberrys anzuschließen. Damit könnte man die begrenzte Länge der USB-Kabel von max 5m beliebig erweitern. Man könnte an den Raspberry auch mehrere ARDUINOS anschließen, die jeweils für einen Teilbereich der Modellbahn zuständig sind. (Diese Funktion ist in Vorbereitung)

Die ARDUINO-Schnittstelle

Die andere Schnittstelle des Z21-Simulators ist die USB-Schnittstelle. Genauer gesagt wird über die USB-Schnittstelle eine serielle Kommunikation simuliert. Diese serielle Kommunikation entspricht genau der Kommunikation zwischen dem DCC-ARDUINO und dem LED-ARDUINO, wenn eine DCC-Digital Zentrale verwendet wird. Für den LED-ARDUINO ist es deshalb überhaupt kein Unterschied, ob er über eine DCC-Digital-Zentrale und den DCC-ARDUINO die Kommandos erhält oder über den Z21-Simulator und den USB-Anschluss.

Wichtig zu wissen ist, dass der LED-ARDUINO im DCC-Modus programmiert werden muss, damit er mit dem Z21-Simulator zusammenarbeiten kann.

Der Z21-Simulator

Wo finde ich den Z21-Simulator?

Wahrscheinlich haben die meisten MobaLedLib-Anwender der Z21-Simulator schon auf ihrem PC. Der **Z21-Simulator** ist nämlich ein Teil des **Farbtest-Programms**, dass zum Überprüfen und Einstellen der Farben verwendet werden kann. Der Farbtester

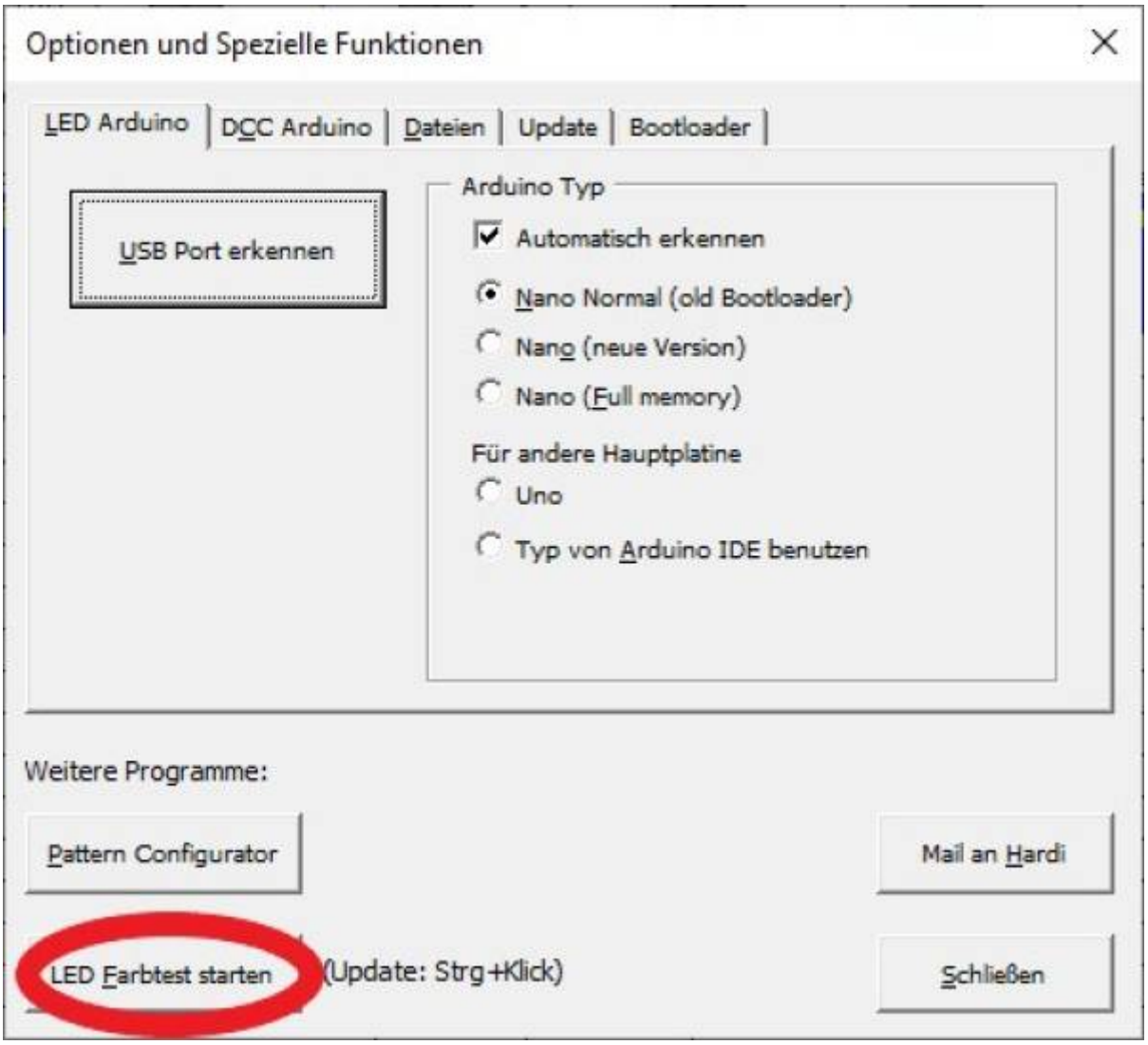
Genauer gesagt, Z21-Simulator und Farbtester sind Teil der **pyProgrammGenerators pyProg_Generator_MobaLedLib.exe**, die auch noch einige weitere Funktionen bietet. pyProgrammGenerator

Wie installiert man den Z21-Simulator?

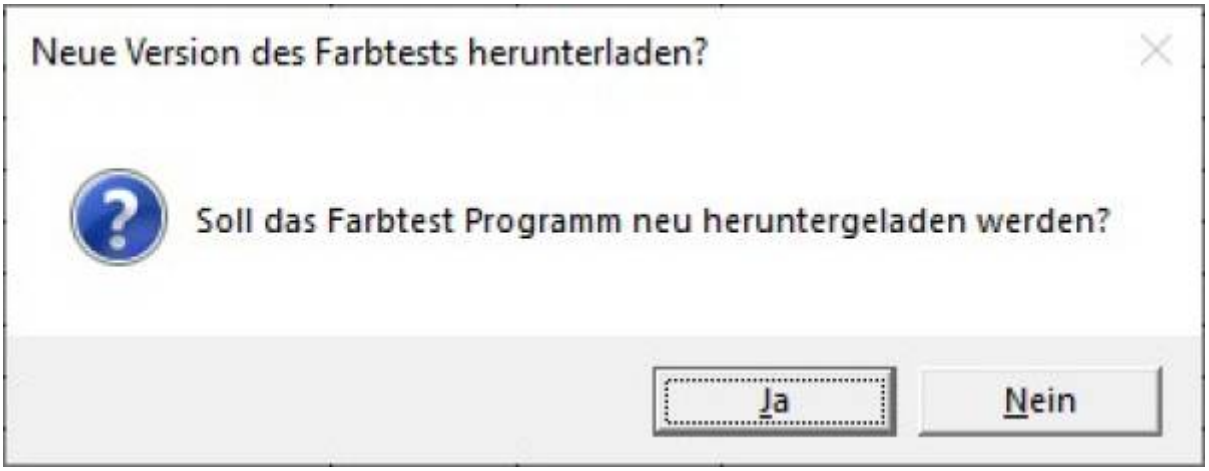
Der Z21-Simulator ist ein Teil des pyProgrammGenerators. Die Installation des pyPrgrammGenerator unter LINUX und Mac ist hier beschrieben: Installation auf einem Raspberry und Installation auf einem Mac

Wenn man den Excel „Programm-Generator“ benutzt ist die Installation viel einfacher:

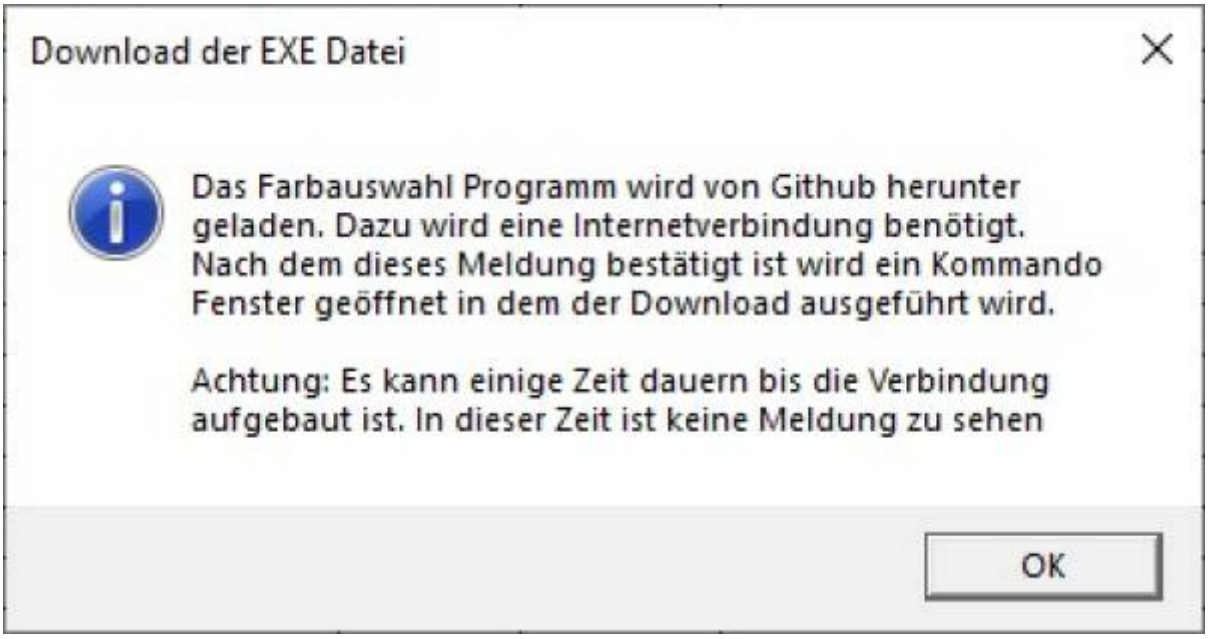
1. Excel „Programm-Generator“ öffnen
2. „Optionen“ anklicken - das folgende Fenster öffnet sich:



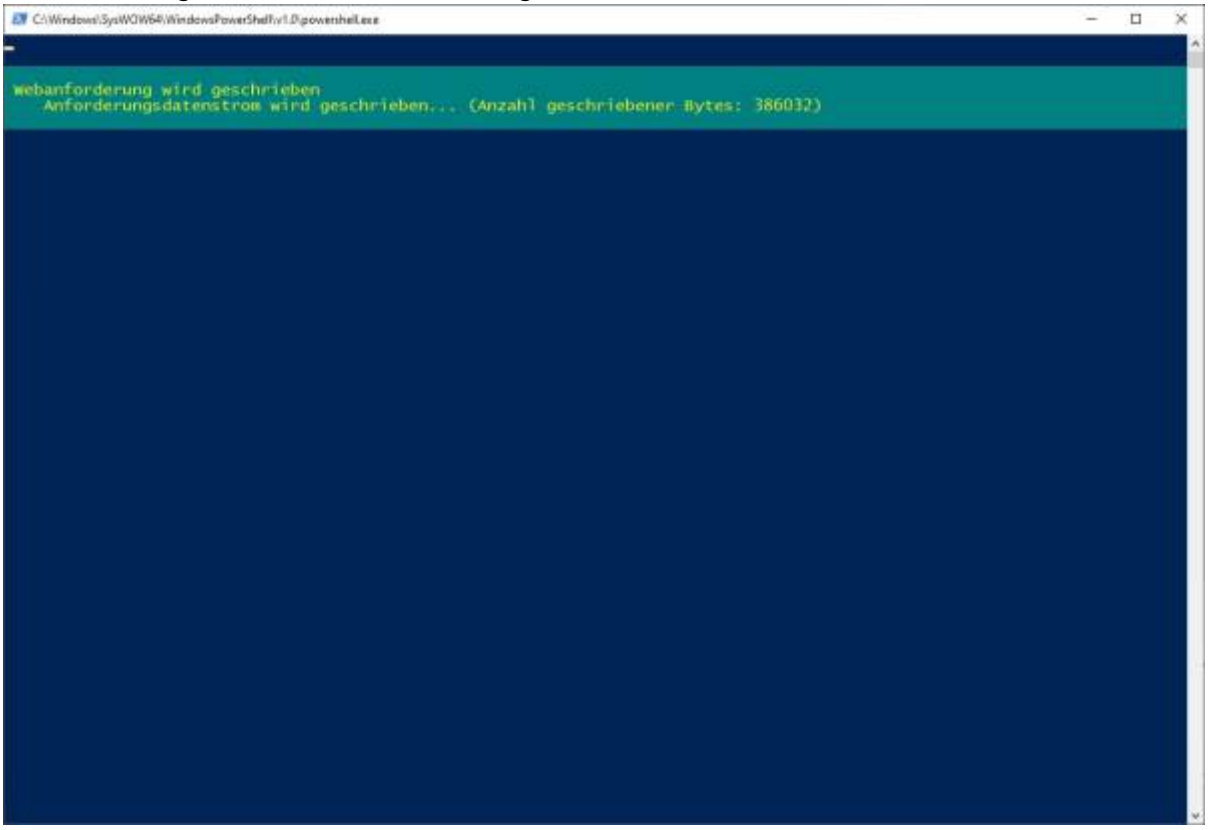
3. Button „LED Farbtest starten“ anklicken



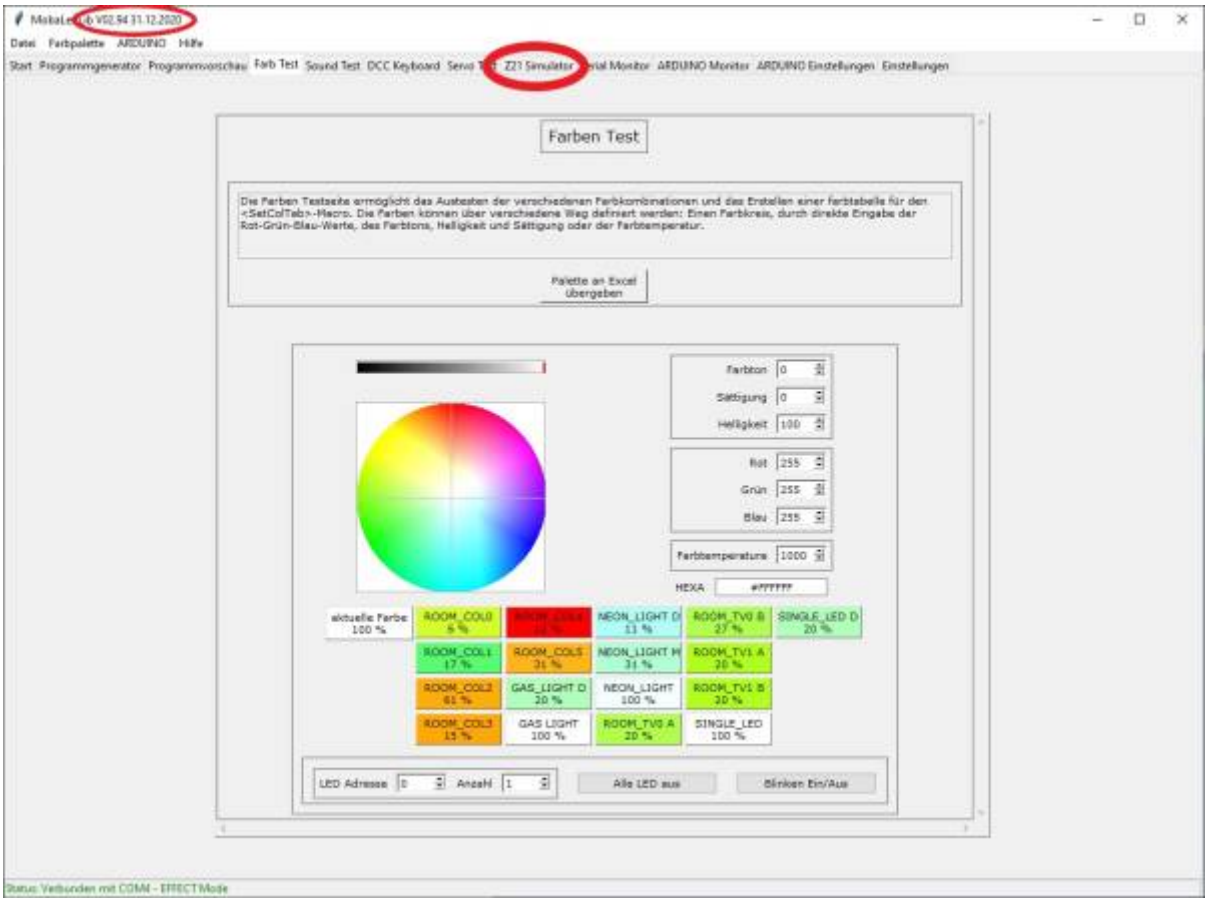
4. Wenn noch das Programm noch nicht heruntergeladen wurde öffnet sich der folgende Dialog:



5. herunterladen mit „ja“ bestätigen
6. nochmal bestätigen und der Download beginnt:



7. danach öffnet sich die Farbtest-Seite des pyprogramGenerators:



8. Bitte die Version oben Links am Fensterrand überprüfen. Es sollte mindestens Version V2.94 sein. Wenn nicht, dann das Programm schließen und den Button „LED Farbtest starten“ zusammen mit der STRG-Taste anklicken. Es wird dann die neueste Version heruntergeladen.
9. Den Reiter „Z21-Simulator“ anklicken
10. Die Z21-Simulator-Seite öffnet sich.

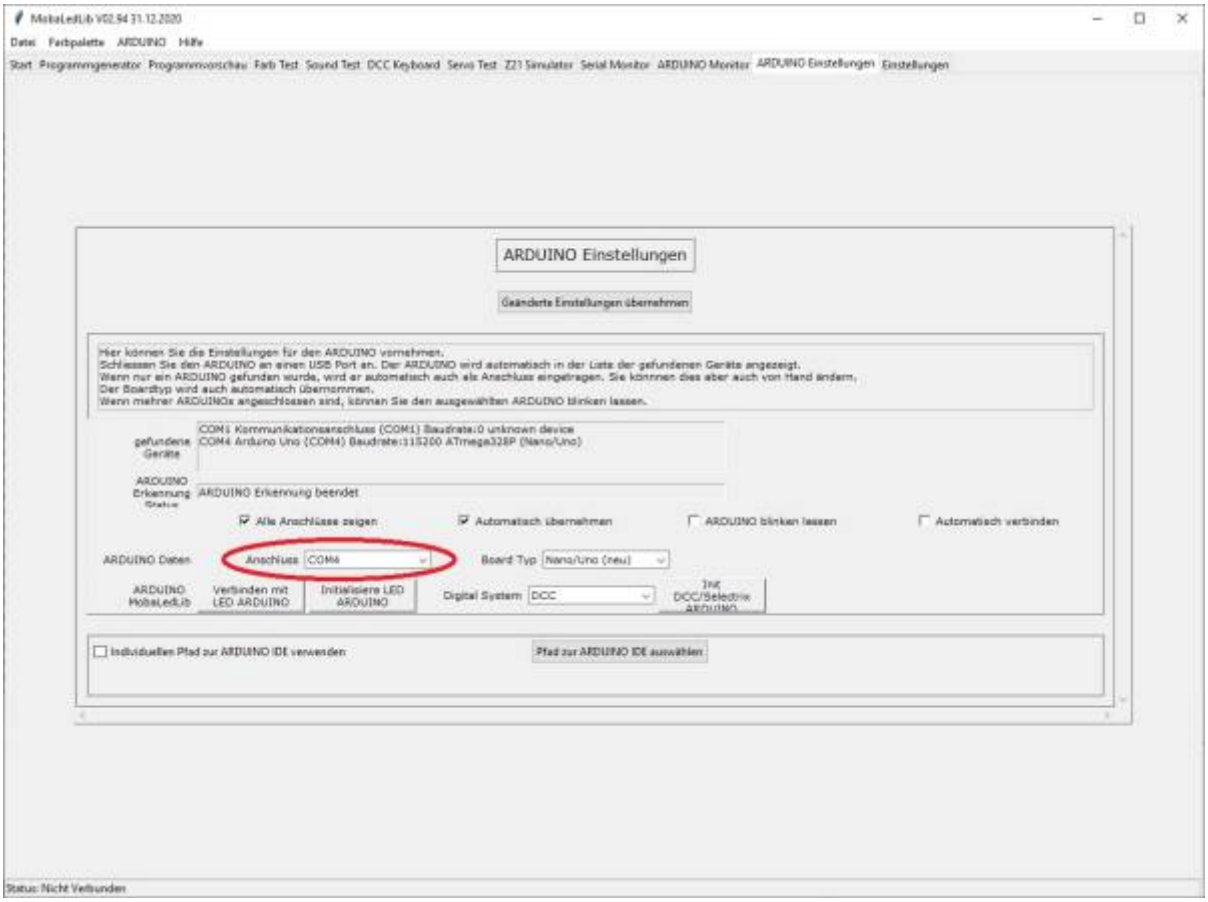
Wie startet man den Z21-Simulator?

Der Z21 Simulator kann folgendermaßen gestartet werden:

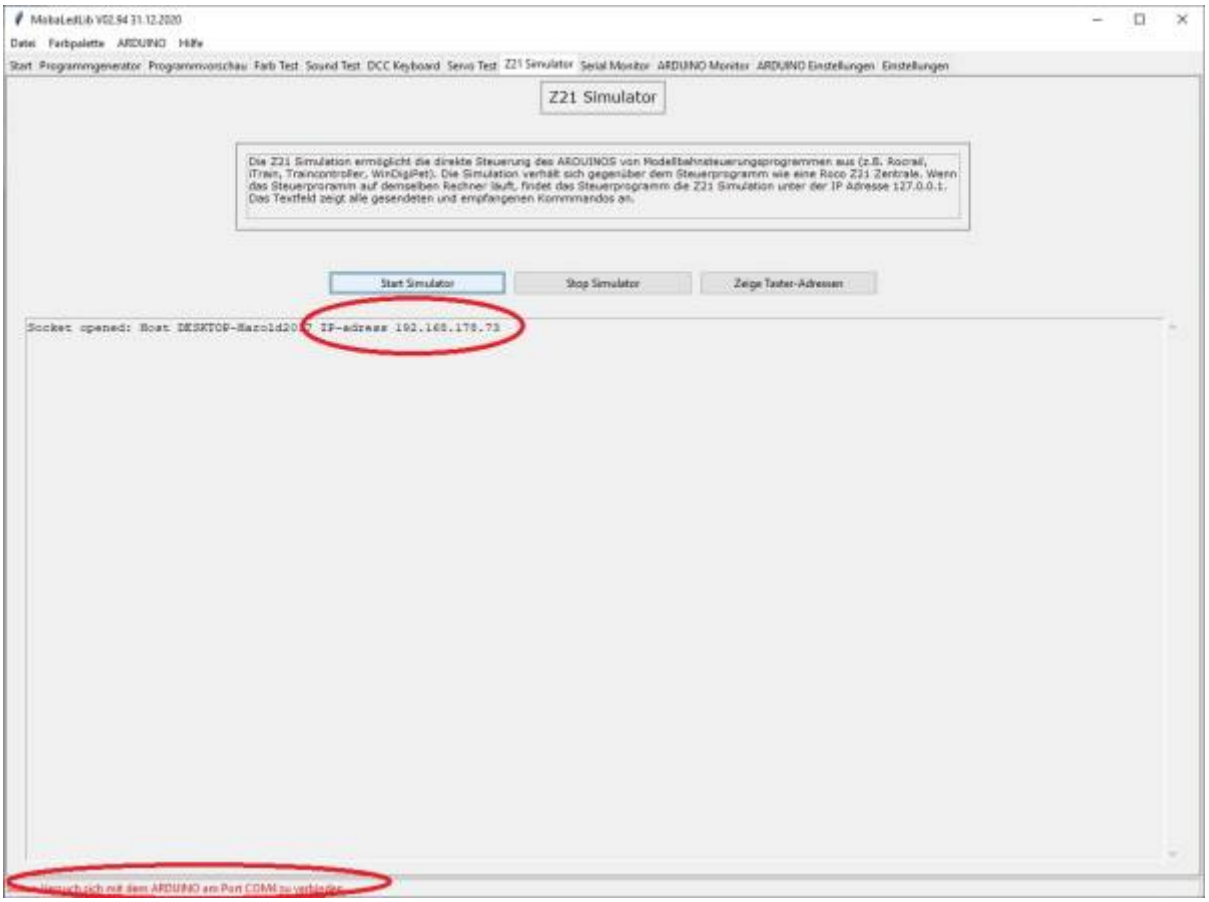
1. Über den Startbutton auf der Z21-Simulator-Seite
2. Automatisch beim Start des pyProgrammGenerators durch eine Kommandozeilenoption

Start des Z21-Simulators mit Startbutton

1. ARDUINO an den USB-Anschluss des PC anschließen
2. pyProgrammGenerator starten
3. auf der ARDUINO-Einstellungen-Seite überprüfen, ob der ARDUINO richtig erkannt wurde



4. die geänderten Einstellungen speichern
5. den Reiter Z21-Simulator auswählen
6. der Z21-Simulator wird gestartet. Im Meldungsfenster wird angezeigt, dass der Port für den Z21-Simulaor geöffnet wurde. Außerdem wird die IP-Adresse des PCs angezeigt. Diese IP-Adresse wird nachher beim Einrichten der Modellbahnsteuerungssoftware oder der Smartphone-App benötigt.In der Status-Zeile ganz unten wird die Meldung angezeigt, dass eine Verbindung zum ARDUINO aufgebaut wird.



7. Wenn alles funktioniert hat, und der Z21-Simulator auch die Verbindung zum ARDUINO herstellen konnte, wird die Meldung „Z21 Simulator started“ angezeigt.
8. Jetzt kann die Modellbahnsteuerungssoftware oder die Smartphone App die MobaLedLib steuern. Wie das genau geht, wird weiter unten beschrieben. -

Start des Z21-Simulators über Kommandozeilenoption

Da es nicht sehr komfortabel ist, jedes mal, wenn man die MobaLedLib mit der Modellbahnsteuerungssoftware steuern möchte, den Z21-Simulator von Hand zu starten, gibt es eine Kommandozeilenoption, mit der man das Programm anweisen kann, den Z21 Simulator automatisch zu starten.

Die Option zum einschalten des Z21-Simulators lautet: –z21simulator True

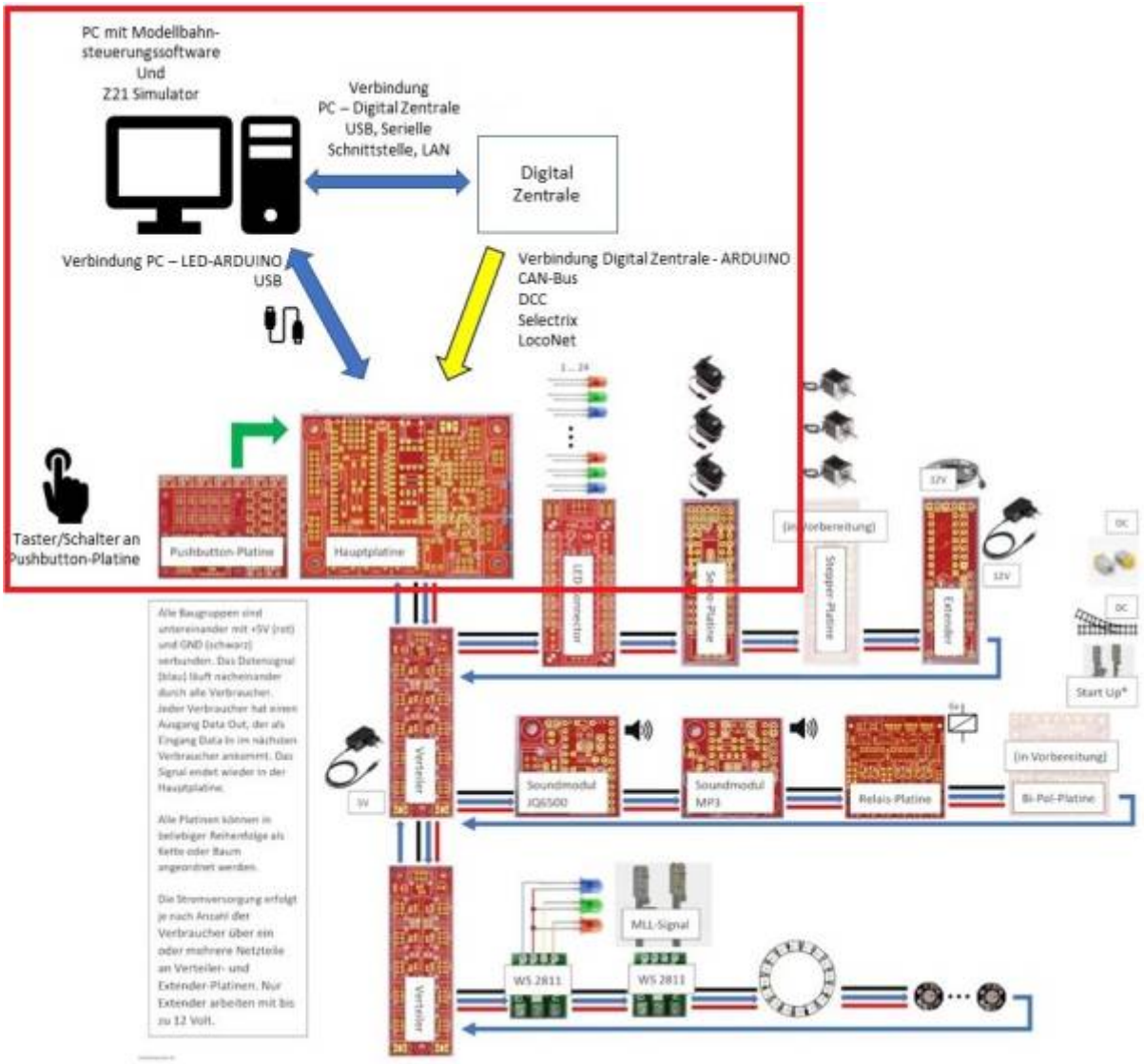
Die komplette Kommandozeile würde also so aussehen: pyProg_Generator_MobaLedLib –z21simulator True

ACHTUNG: Diese Funktion ist in Vorbereitung und noch nicht implementiert!

Steuerung über eine Modellbahnsteuerungssoftware

Wie oben beschrieben, kann die MobaLedLib über zwei Wege von einer Modellbahnsteuerungssoftware kontrolliert werden.

1. über eine Digitalzentrale wie z.B. CS2/CS3, ECOS, Z21, Lenz, TAMS etc.
2. direkt über den USB-Anschluss und den Z21-Simulator



Steuerung über Digitalzentrale

Die Verbindung zwischen der Modellbahnsteuerungssoftware und der Digitalzentrale wird entsprechend der Dokumentation der Software hergestellt. Beispiele für Die Einrichtung von verschiedenen Digitalzentralen zur Steuerung der MLL findet Ihr weiter unten.

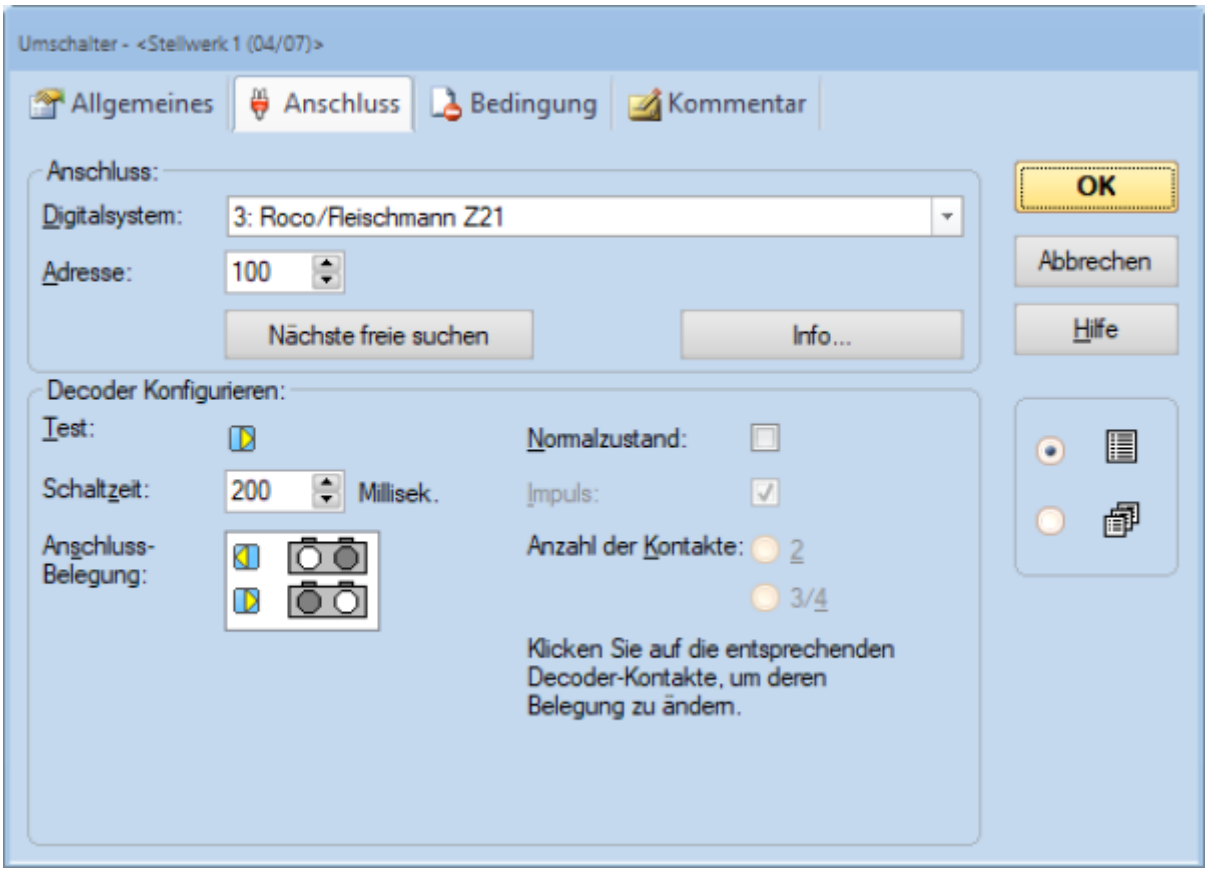
Will man einzelne MobaLedLib-Macros durch Befehle von der Modellbahnsteuerungssoftware ein- bzw. ausschalten, dann muss man diese als Schalter, Signal oder Weiche in der Modellbahnsteuerungssoftware einrichten. Der Schalter bekommt eine Adresse. Wie die Adresse aussieht ist abhängig von dem Steuerungsprotokoll CAN, DCC oder Selectrix.

Als Beispiel schauen wir uns hier DCC an:

Bei DCC ist diese Adresse eine Zahl zwischen 1 und 2028 z.B. 100.

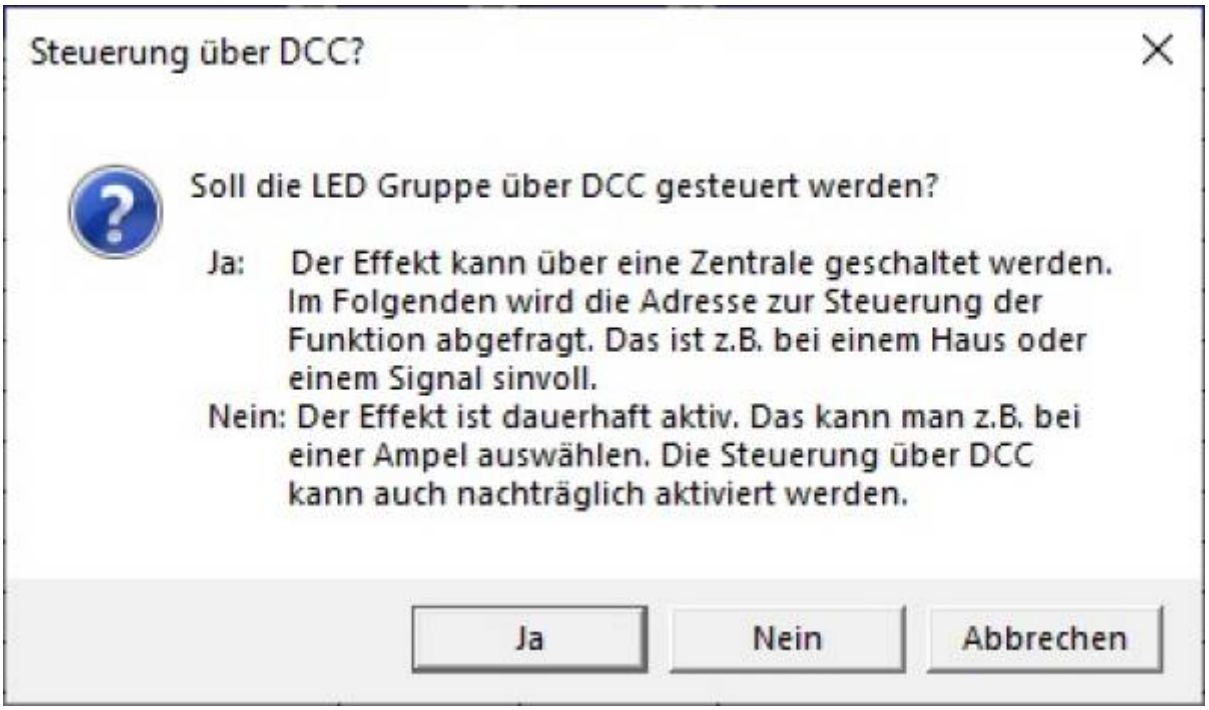
Diese Adresse wird bei dem Schalter in der Modellbahnsteuerungssoftware eingetragen.

Beispiel für TrainController:

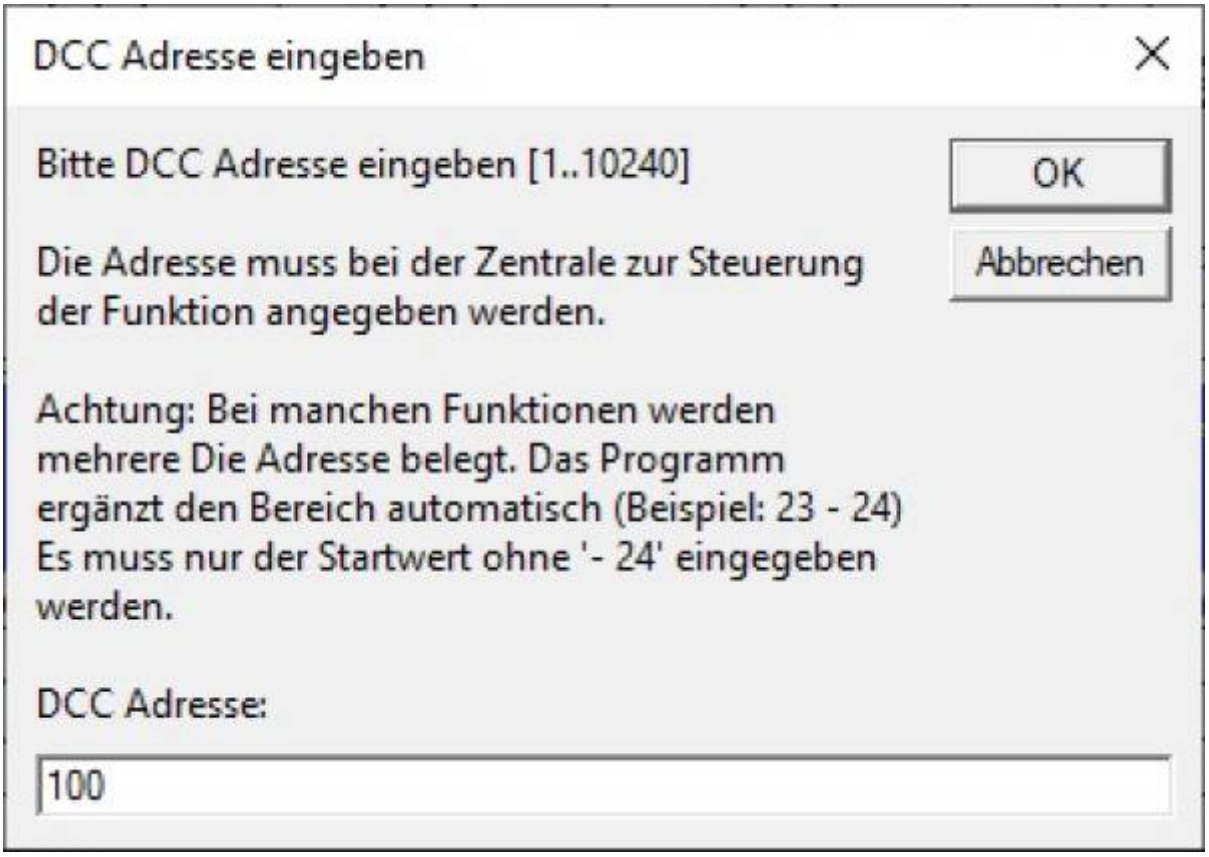


Bei der Programmierung der MobaLedLib-Macros kann man jetzt diese Adresse mit angegeben, entweder als Antwort auf die Frage nach einer Adresse oder direkt in der Excel-Tabelle

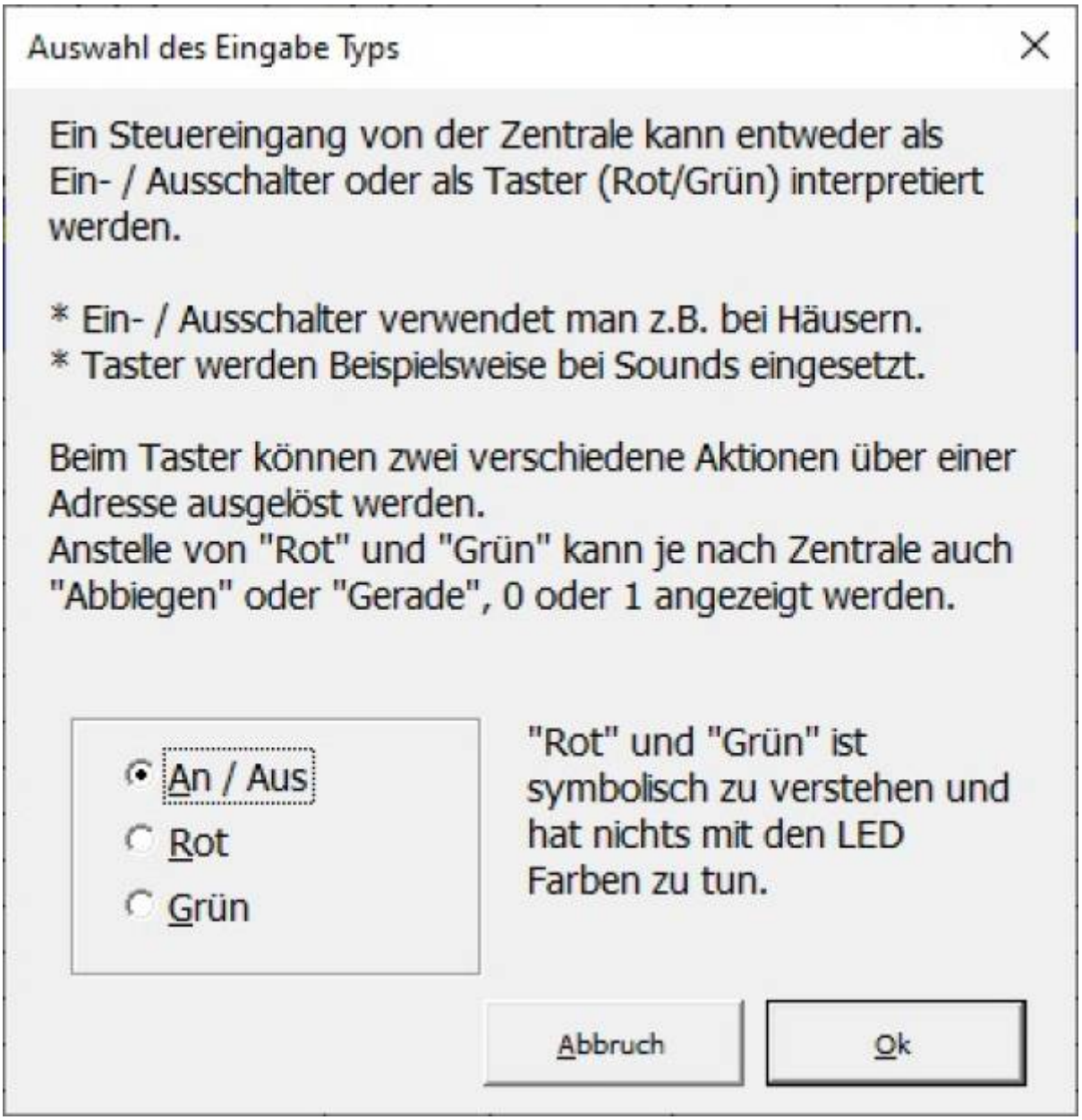
Beispiel für die Abfrage:



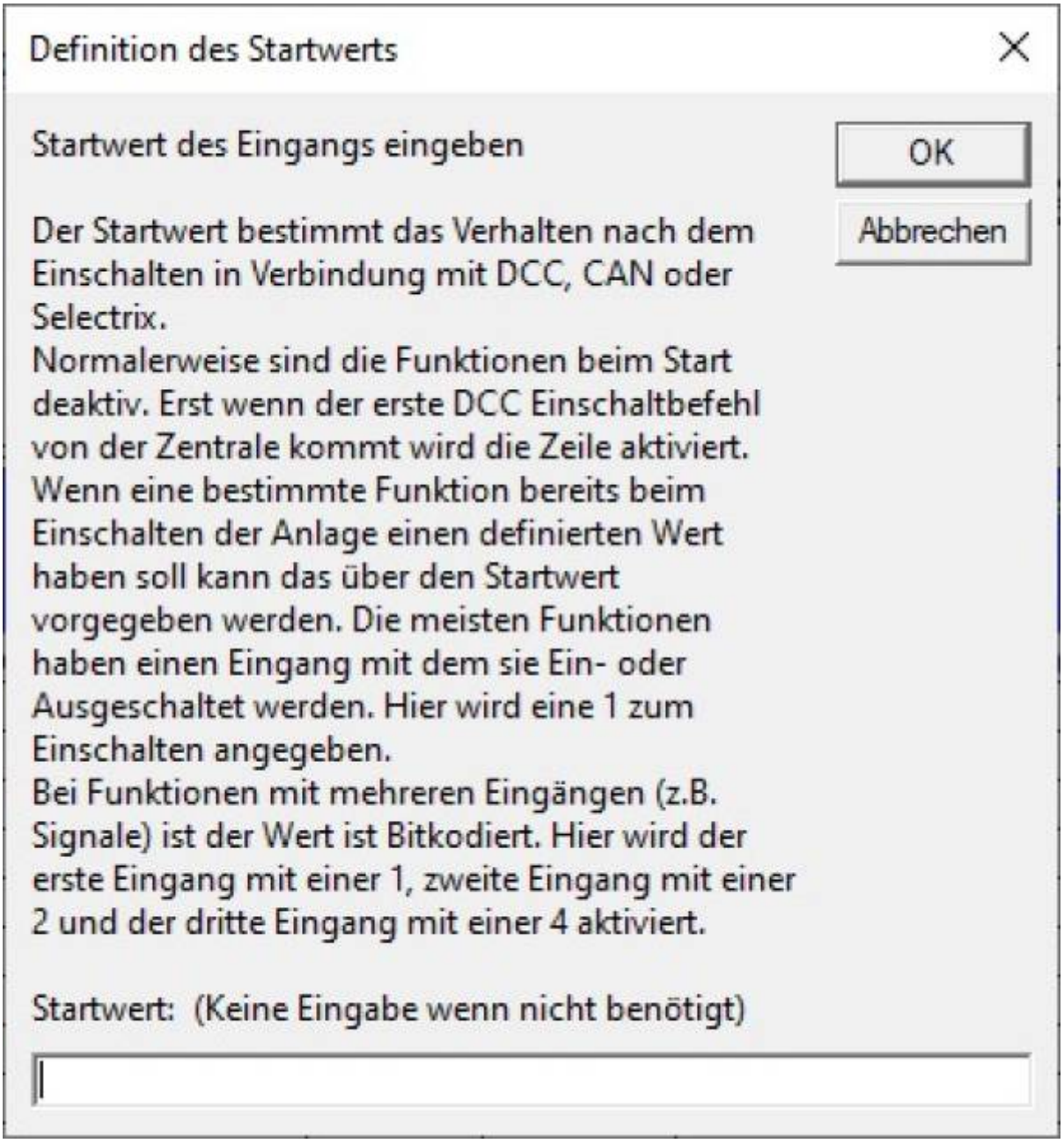
- Mit „Ja“ antworten

- 

DCC Adresse eingeben (hier 100)



- Eingabetyp Ein/Aus ist für Schalter und Effekte, bei Signalen kann es auch „Rot“ oder „Grün“ sein



- Der Startwert sollte leer gelassen werden. Die Modellbahnsteuerungssoftware kümmert sich um die richtige Einstellung

Oder direkt in der Excel Tabelle:



Die Adresse wird in die Spalte „Adresse oder Name“ eingetragen. Der Typ in die Spalte Typ.

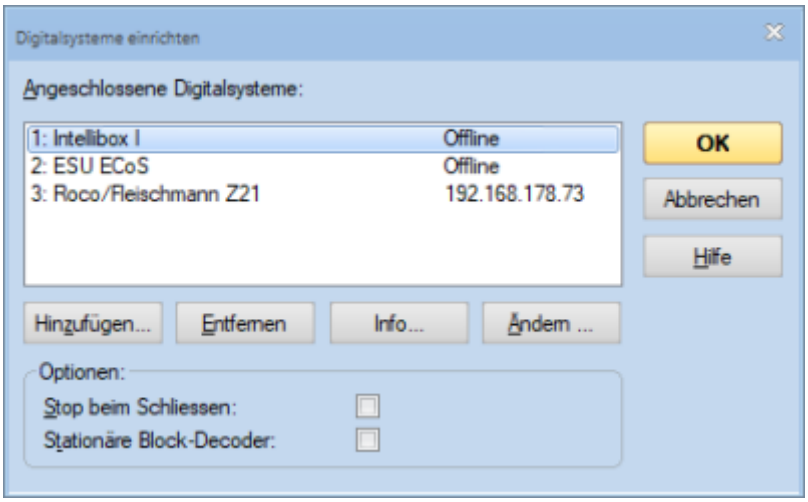
ACHTUNG: **Probleme mit DCC Adressen by verschiedenen DCC Zentralen** Einige Hersteller und Programmierer von DCC-Zentralen und deren Software nummerieren die Weichen ab Modul 0 (mit jeweils 4 Weichen), andere DCC-Zentralenhersteller erst ab Modul 1. Diese unterschiedliche Zählweise ist historisch aus einer Schwäche der Spezifikation NMRA S-9.2.1 gewachsen, wobei keine der beiden Zählweisen grundsätzlich als „falsch“ bezeichnet werden konnte.

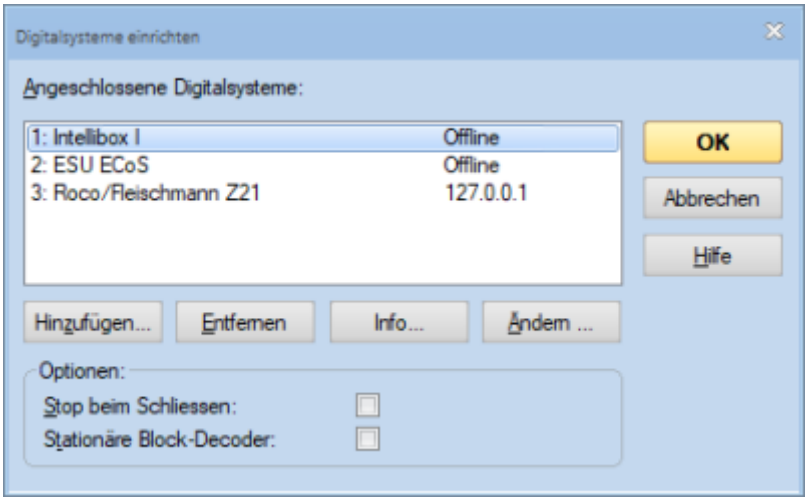
Details zu diesem Problem und der Behebung findet Ihr hier: Probleme bei DCC

Steuerung direkt über USB und Z21 Simulator

Im Prinzip funktioniert die direkte Steuerung über USB für die Modellbahnsteuerung genauso wie die Steuerung über eine Roco Z21 Digitalzentrale mit DCC. Der einzige Unterschied ist, dass als IP-Adresse der Z21 Zentrale die IP-Adresse eingegeben werden muss, die der Z21 Simulator nach dem Start anzeigt.

Laufen der Z21-Simulator und die Modellbahnsteuerung auf dem selben PC kann stattdessen auch die IP-Adresse 127.0.0.1 angegeben werden. Diese Adresse bedeutet, dass sich das Ziel auf dem selbem Gerät befindet.





Für den ARDUINO müssen die Adressen, wie bei einer Digitalzentrale mit DCC angegeben werden. Siehe Beispiel in dem vorangegangenen Kapitel.

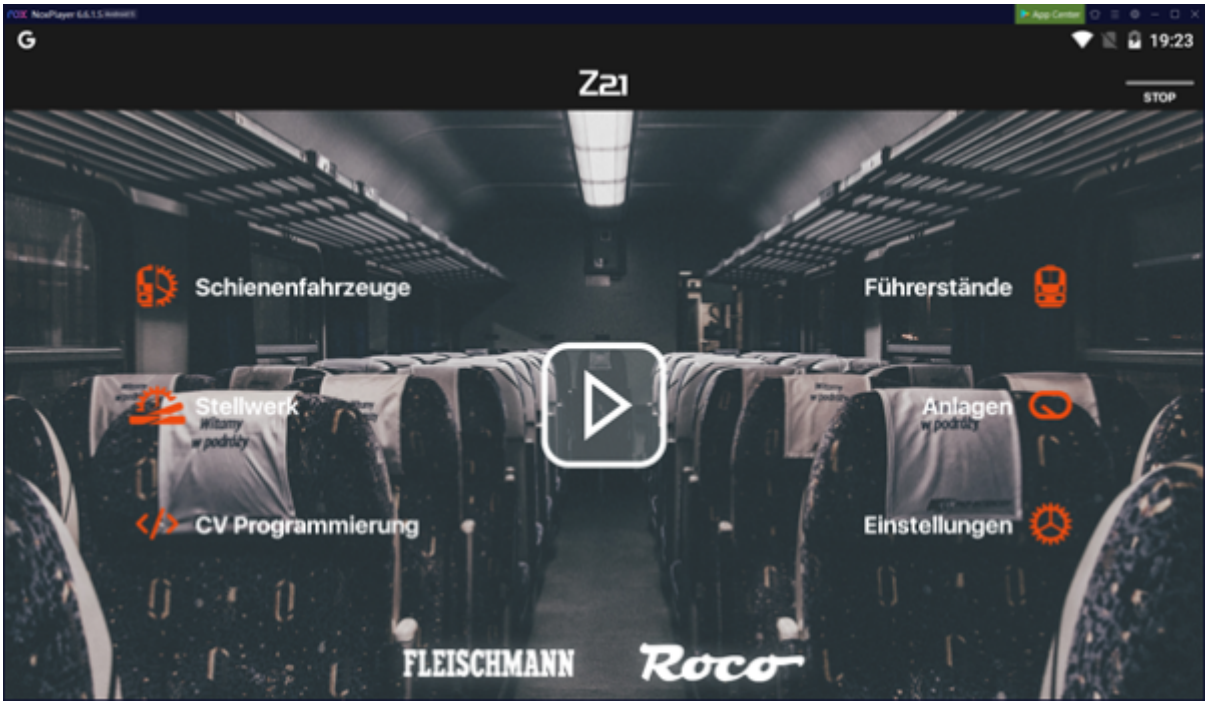
Steuerung über Smartphone App

Die Steuerung der MobaledLib über eine Smartphone App ist eine Interessante Möglichkeit. Wenn die Digitaltentrale eine Smartphone App unterstützt kann diese App natürlich für die Steuerung der MobaLedLib genutzt werden.

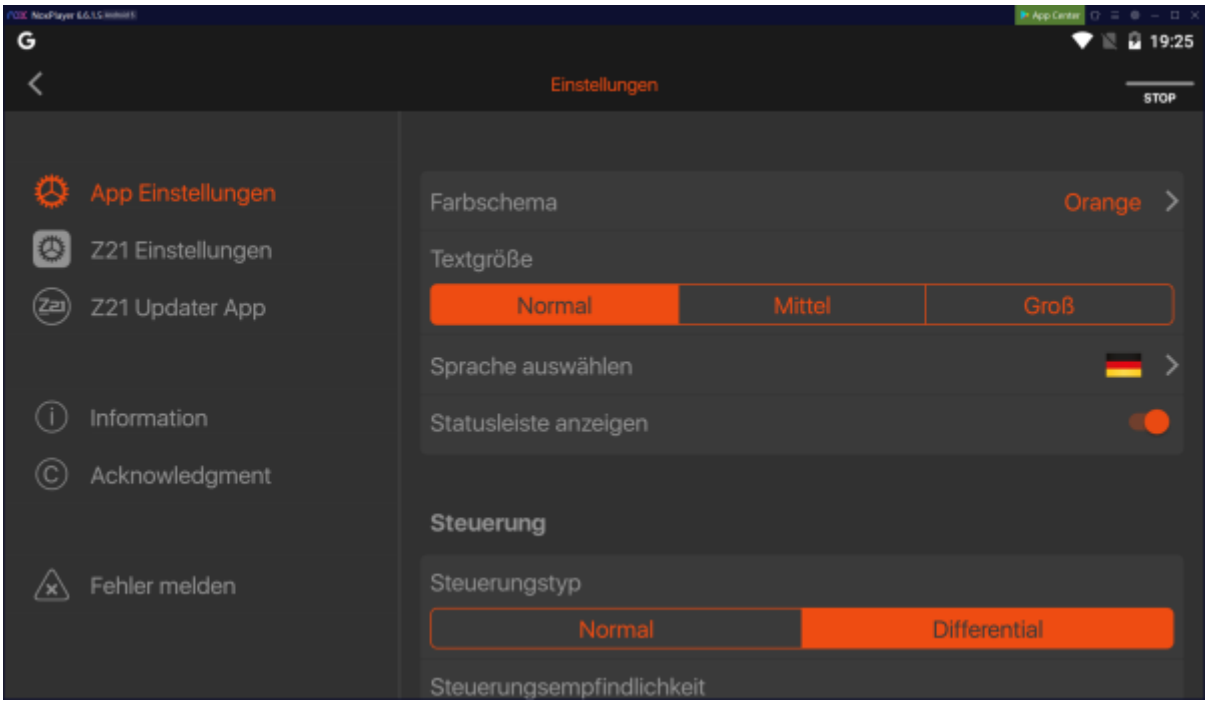
Der Z21-Simulator eröffnet hier aber eine neue Möglichkeit.

Die ROCO Z21 App ist frei verfügbar. Auf der verlinkten Seite ist beschrieben, wie man die ROCO Z21 App for Android oder IOS herunterlädt und installiert.

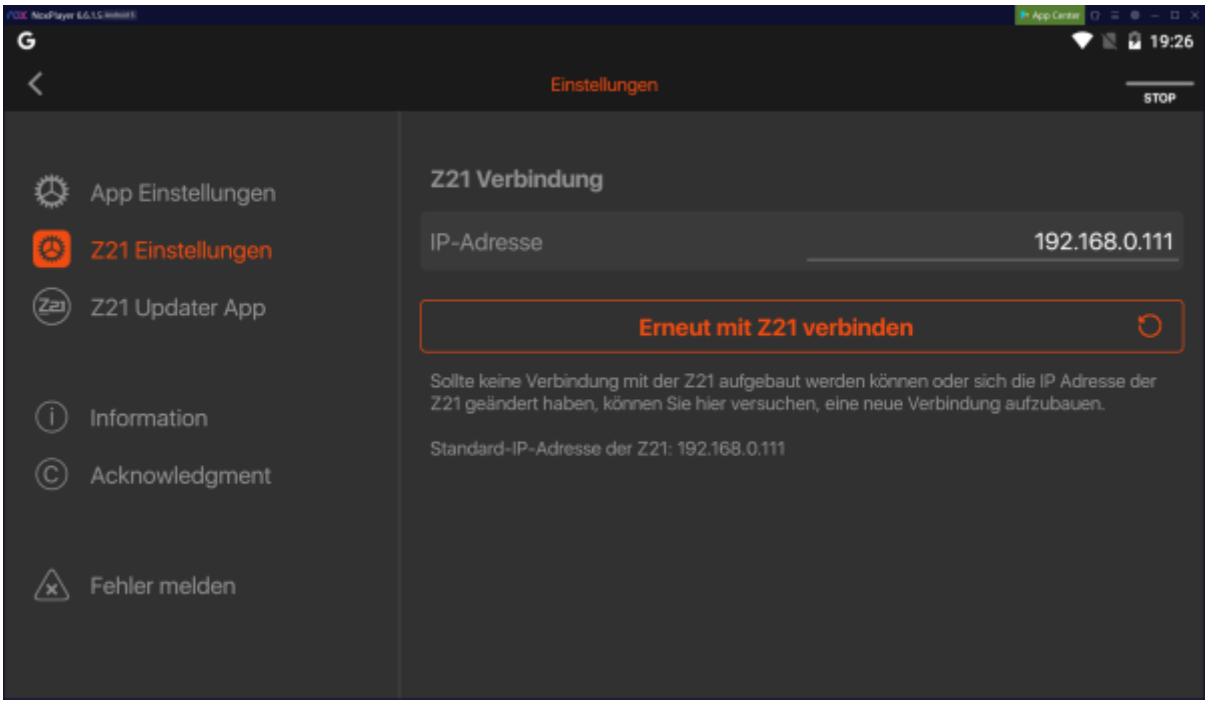
Nach dem Start der Z21 App wird folgender Bildschirm angezeigt.



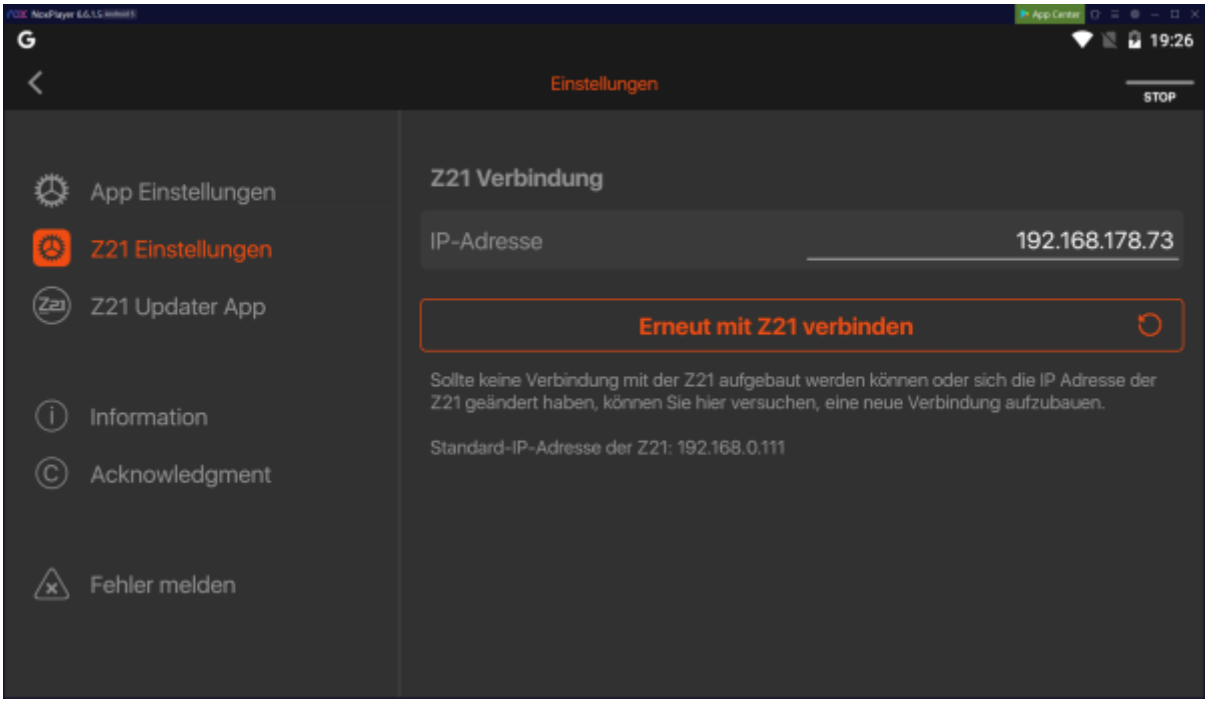
Auf „Einstellungen“ tippen. Dann erscheint die „Einstellungen“ Seite



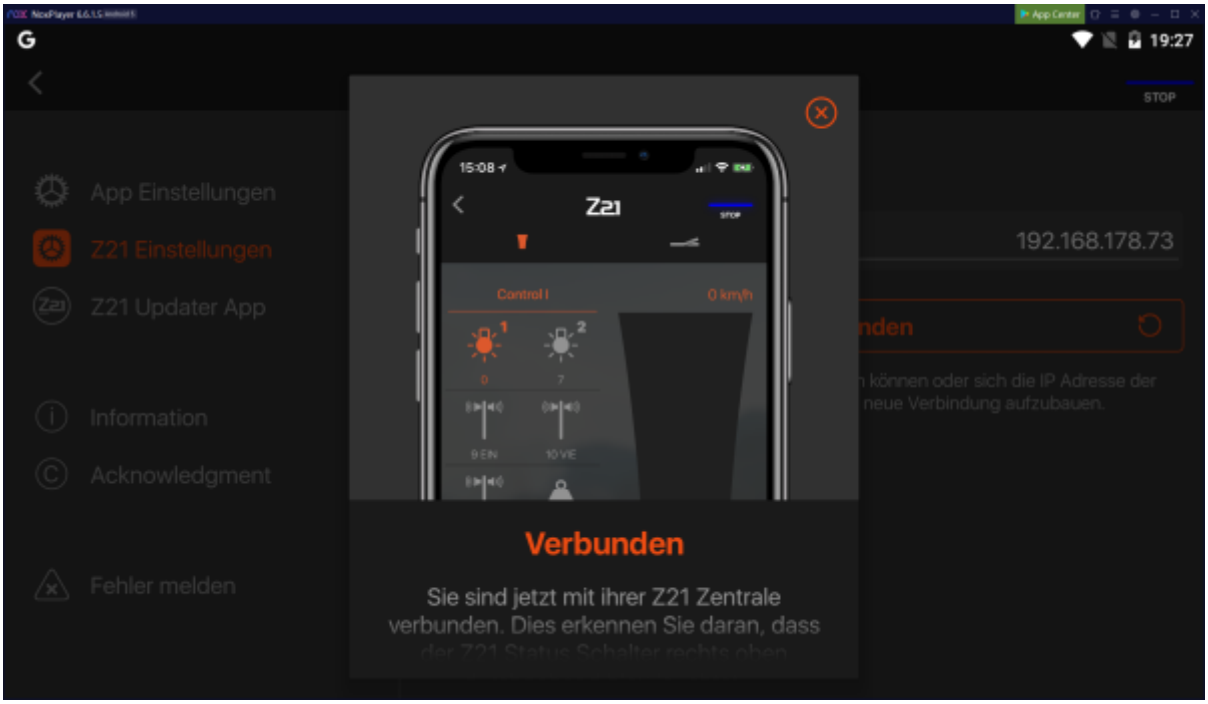
Auf „Z21 Einstellngen“ tippen. Die Z21-Einstellungen Seite erscheint.



Unter IP-Adresse ist die Standard Z21 IP-Adresse eingetragen 192.168.0.111. Diese Adresse muss ersetzt werden durch die IP-Adresse, die beim Start des Z21 Simulators angezeigt wird. In diesem Fall 192.168.178.73



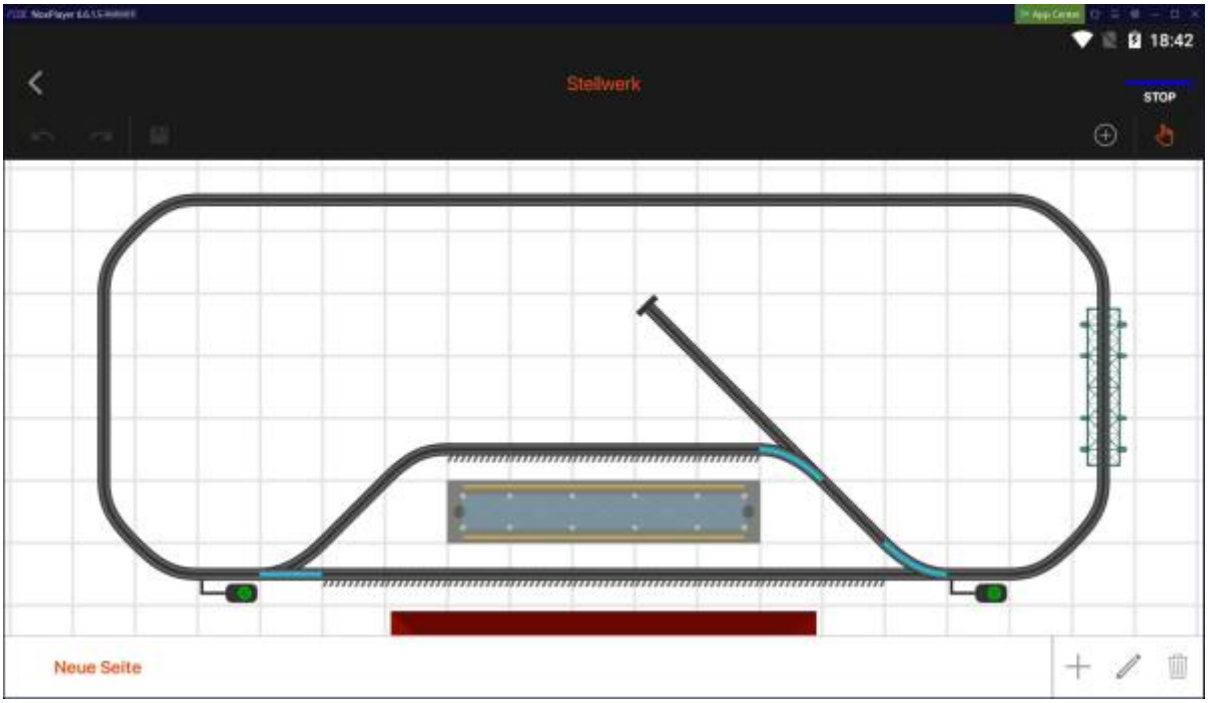
Auf den Button „Erneut mit Z21 verbinden“ tippen. Die App meldet, dass sie mit der Z21-Simulation verbunden ist.



Schalter in der Z21 App einrichten

Im folgenden werden wir in der Z21 App einen Schalter einrichten, der eine Lampe mit der Adresse 100 ein- bzw ausschaltet.

In der App Hauptseite „Stellwerk“ antippen. Es öffnet sich die Seite zum Bearbeiten eines Stellwerks.

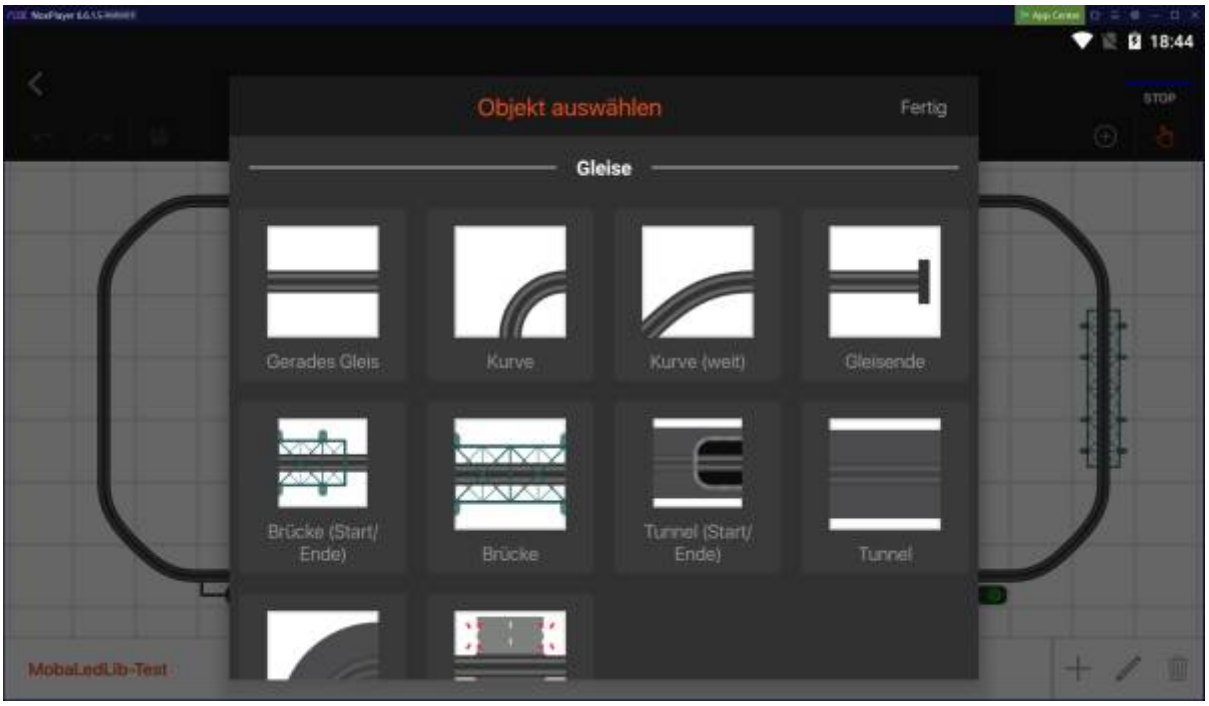


„Neue Seite“ antippen, um den Name des Stellwerks zu ändern.

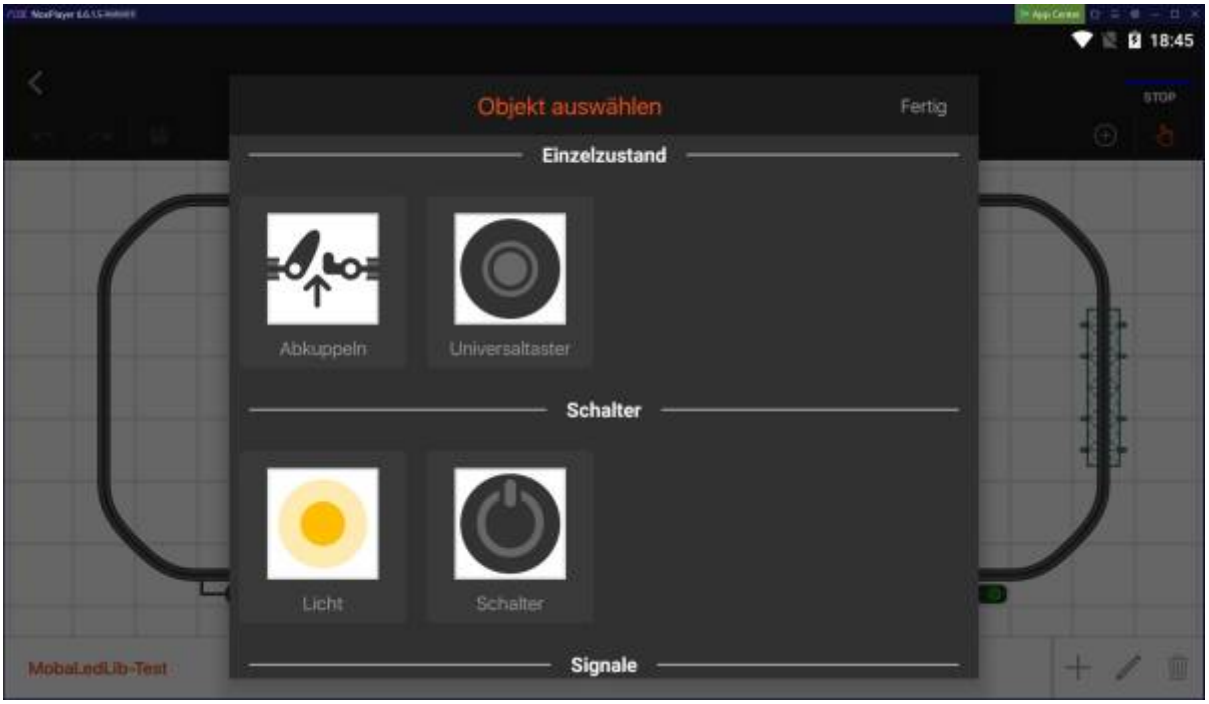


Das kleine „+“ antippen, um einen Schalter in das Stzellwerk einfügen zu können.

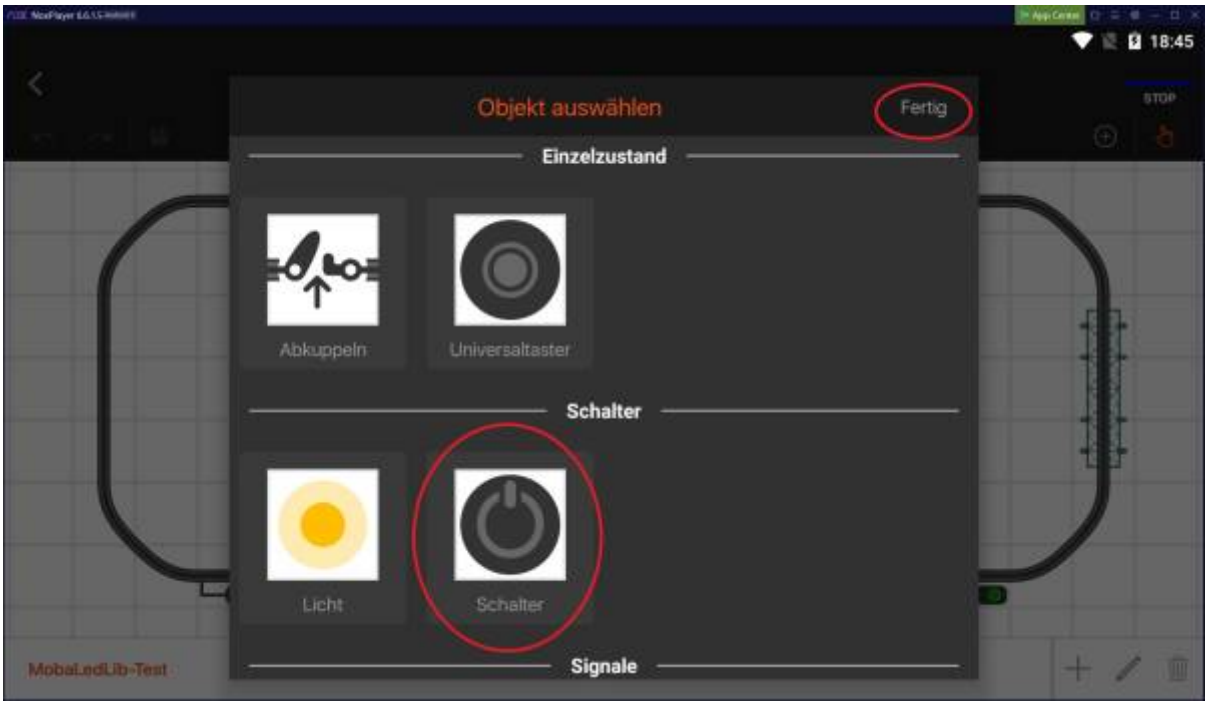
Die „Object Auswahl“-Seite wird angezeigt.



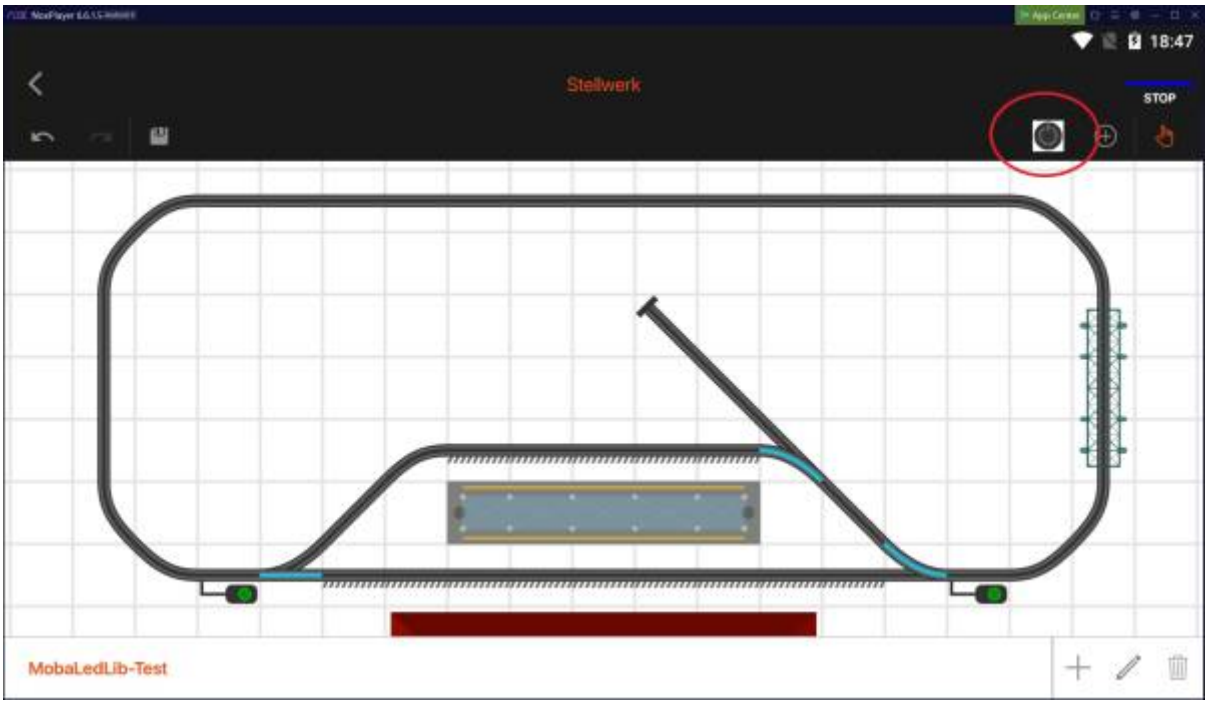
Nach unten scrollen bis zum Schalter



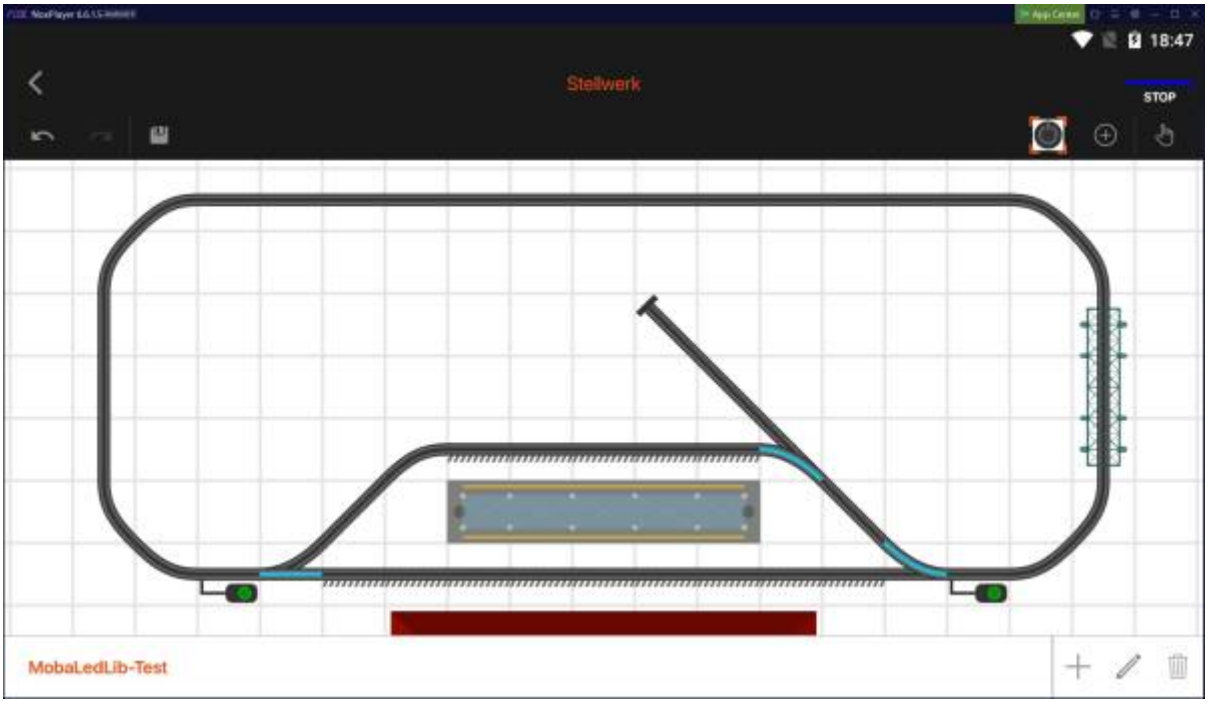
„Schalter“ auswählen



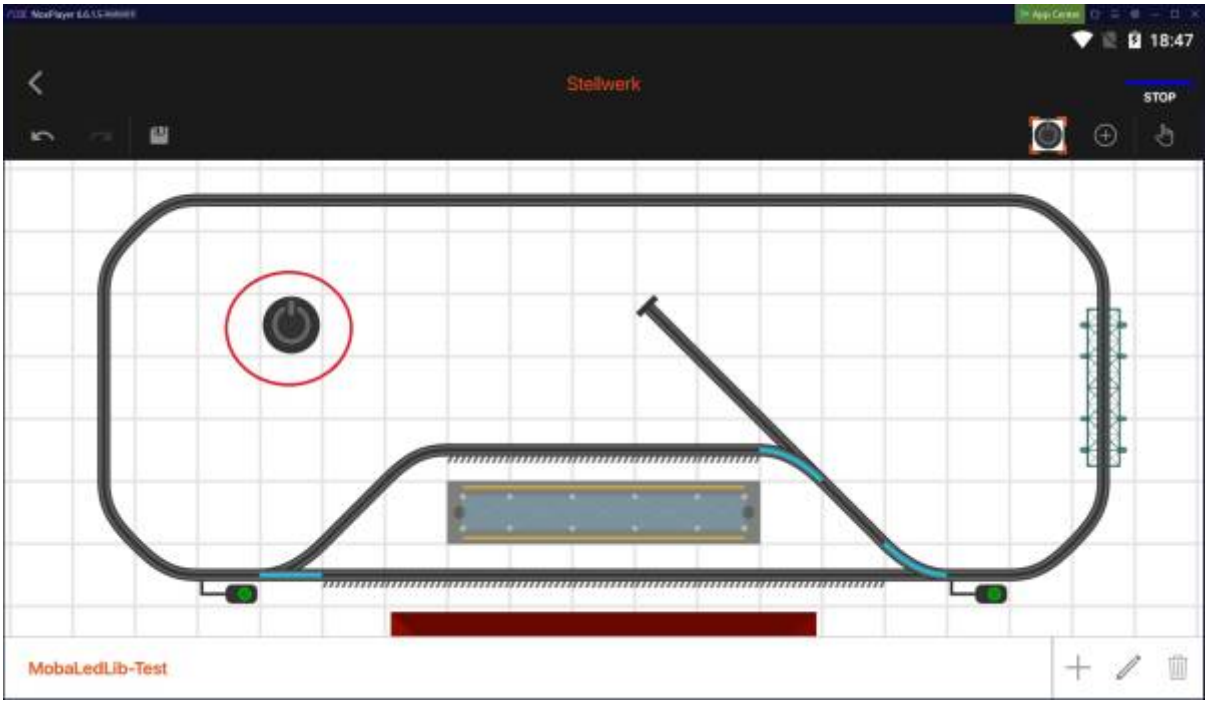
Das Schalter-Symbol ist jetzt in der Auswahlleiste verfügbar



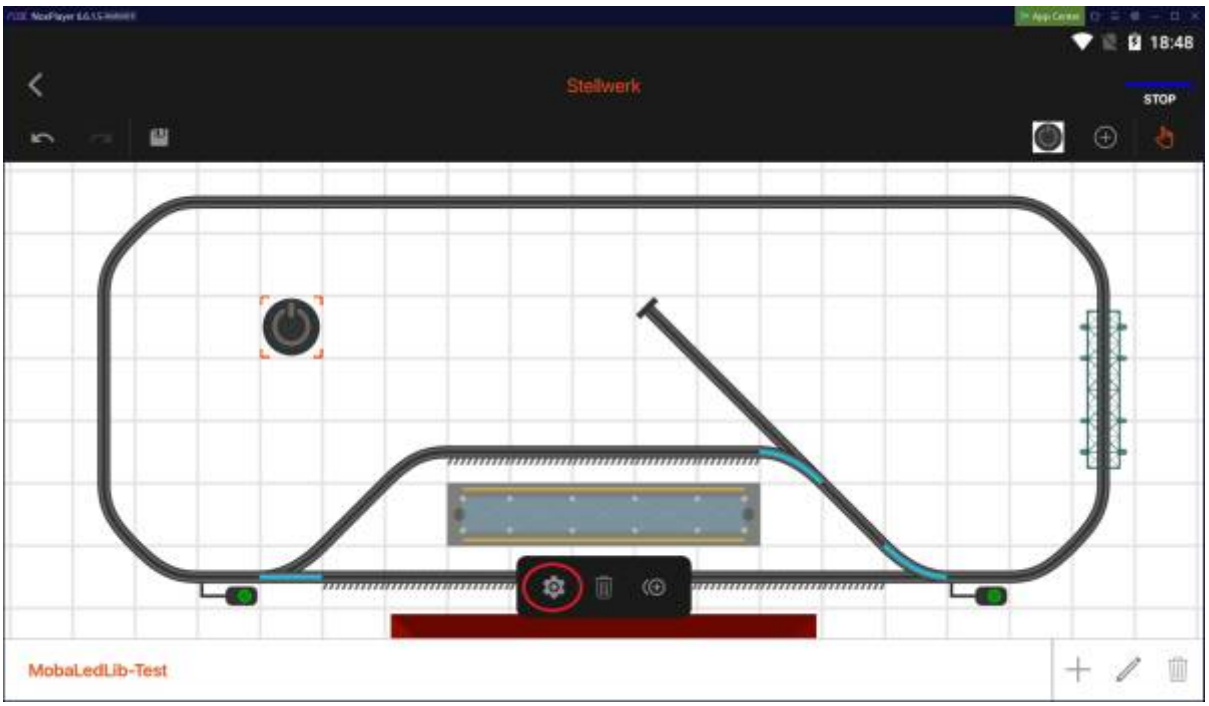
Das kleine Schaltersymbol antippen so dass es „aktiv“ ist.



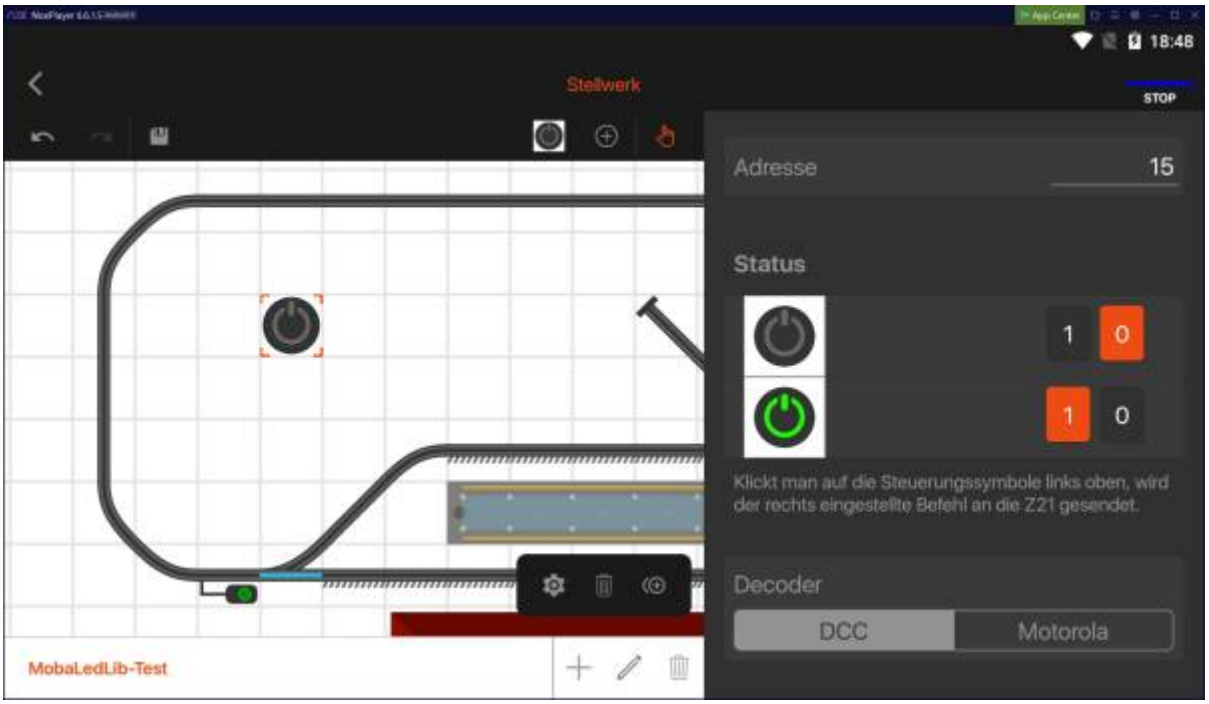
Und irgendwo im ein Stellwerksfeld antippen, in dem der Schalter eingefügt werden soll. Der Schalter erscheint in dem Feld.



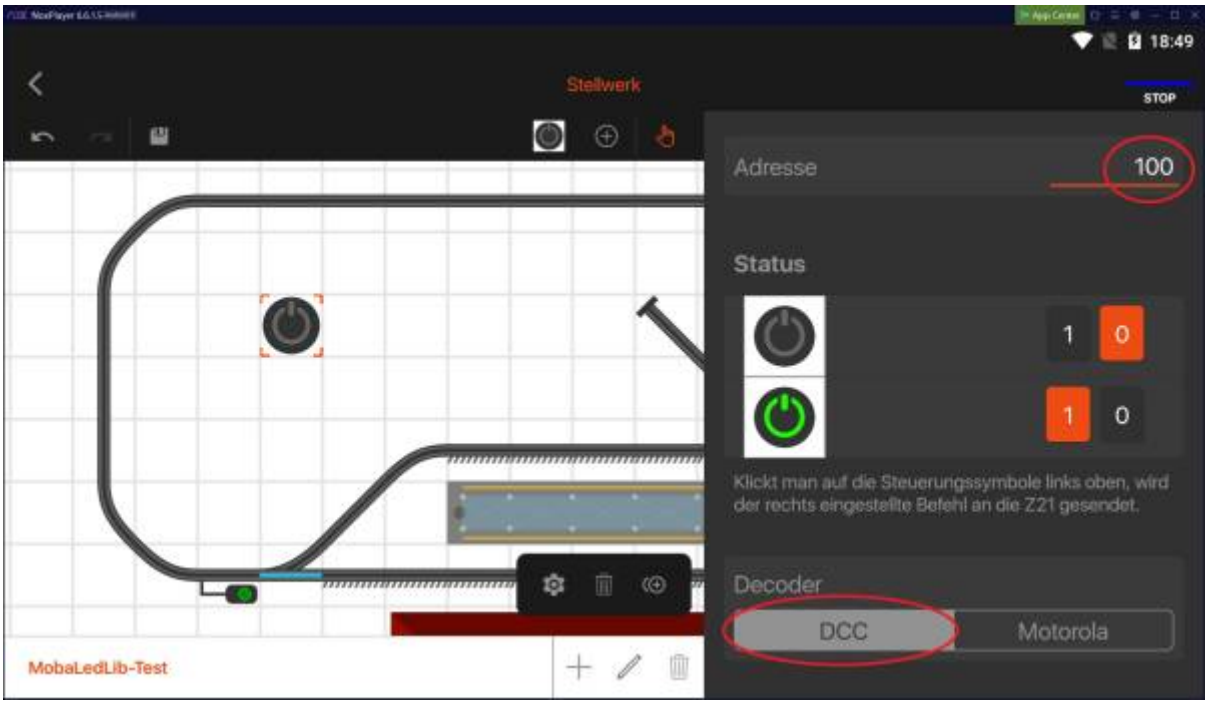
Den Schalter antippen und es erscheint ein Auswahlmenue. Das Einstellungen-Symbol antippen, um die Einstellungen des Schalters zu ändern.



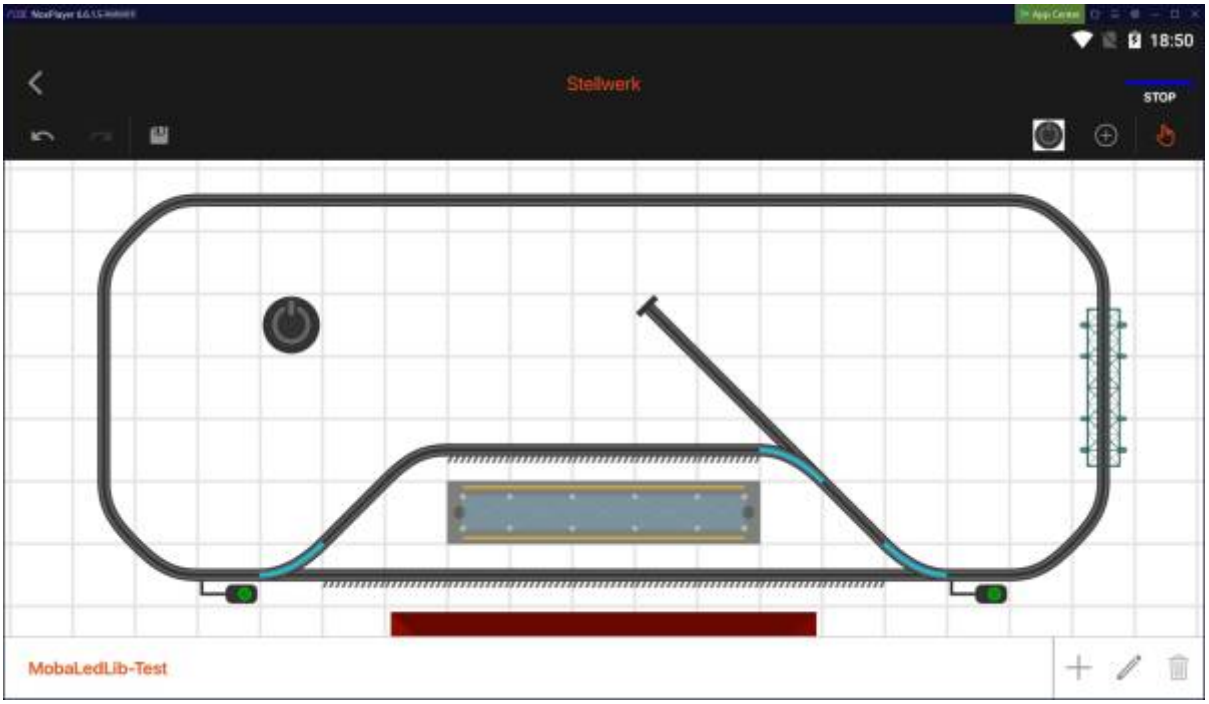
Es erscheint die Einstellen-Seite.



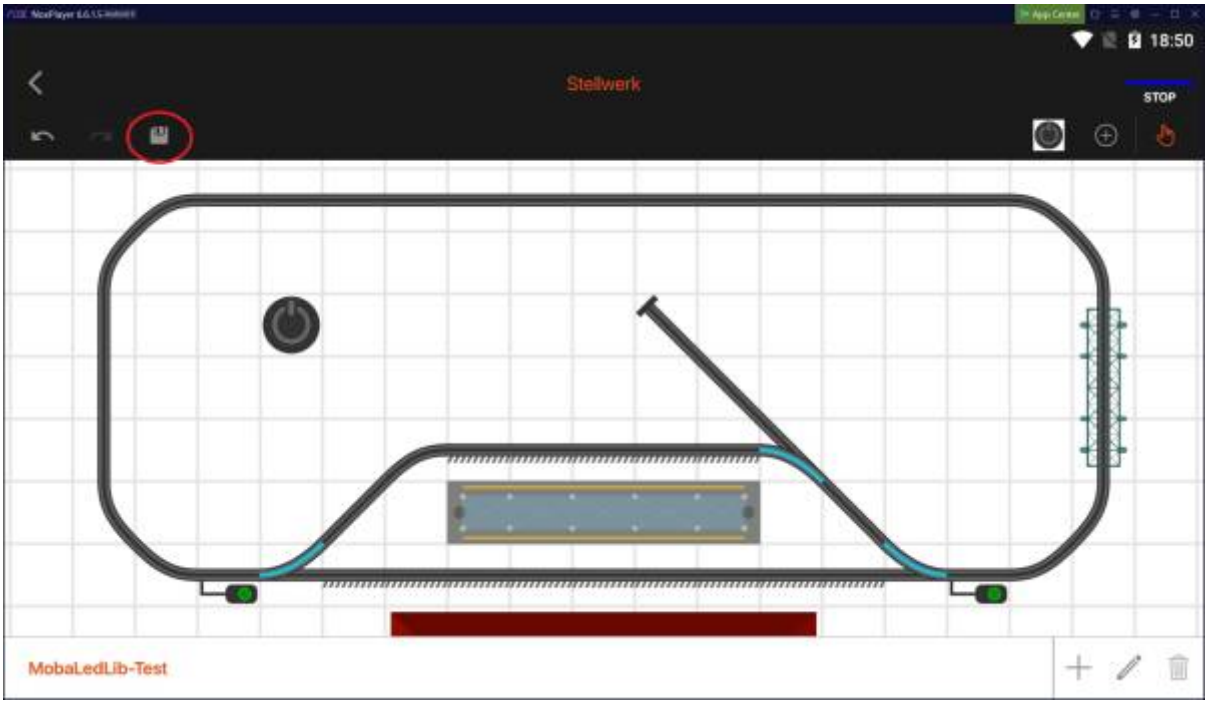
Die Adresse auf 100 ändern. Darauf achten, dass der Decoder auf DCC steht. Durch antippen der Schalter-symbole kann man den Schalter testen.



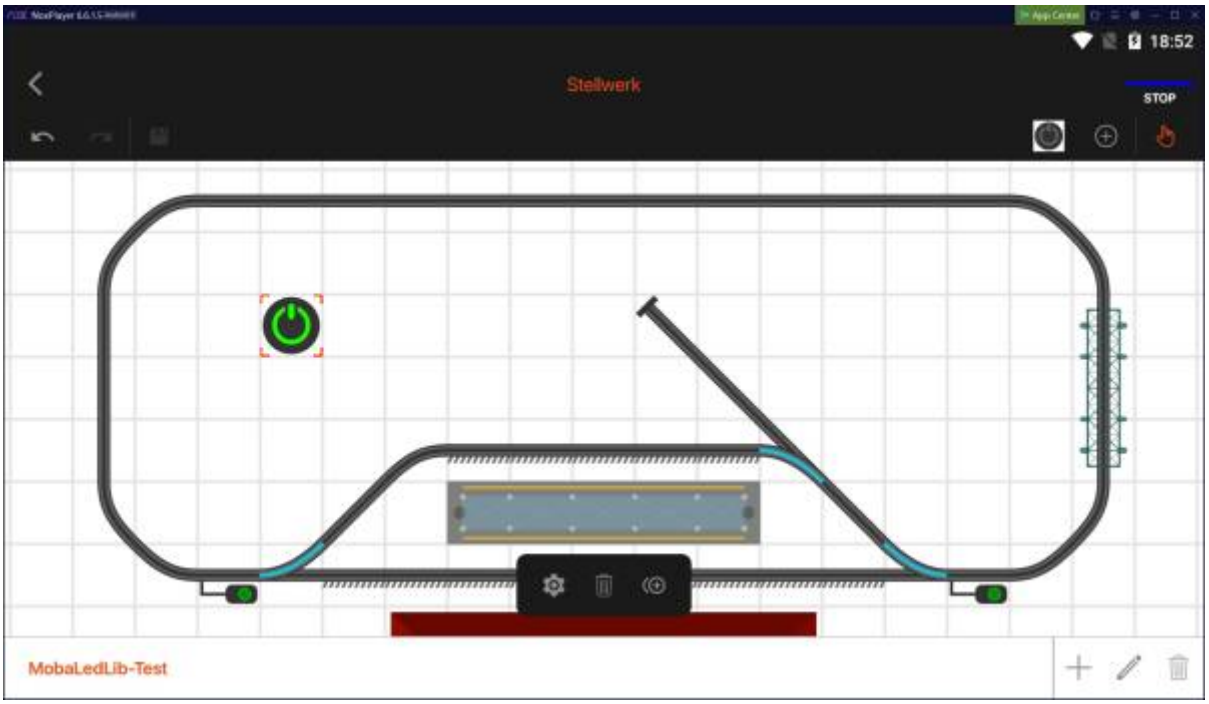
Irgendwo ins Stellwerk tippen. Das Menu verschwindet und die Einstellungen sind gespeichert.



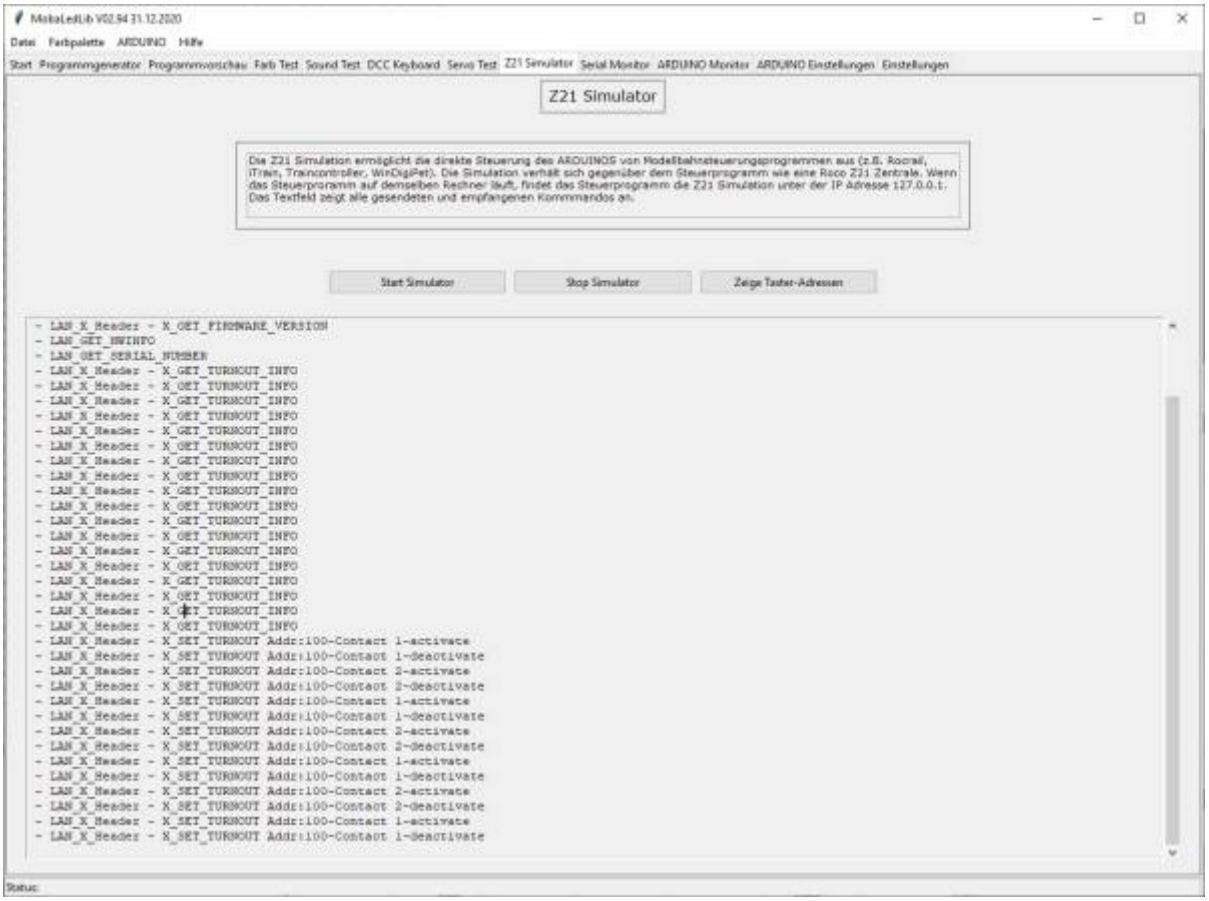
Mit dem kleine Disketten Symbol das geänderte Stellwerk speichern



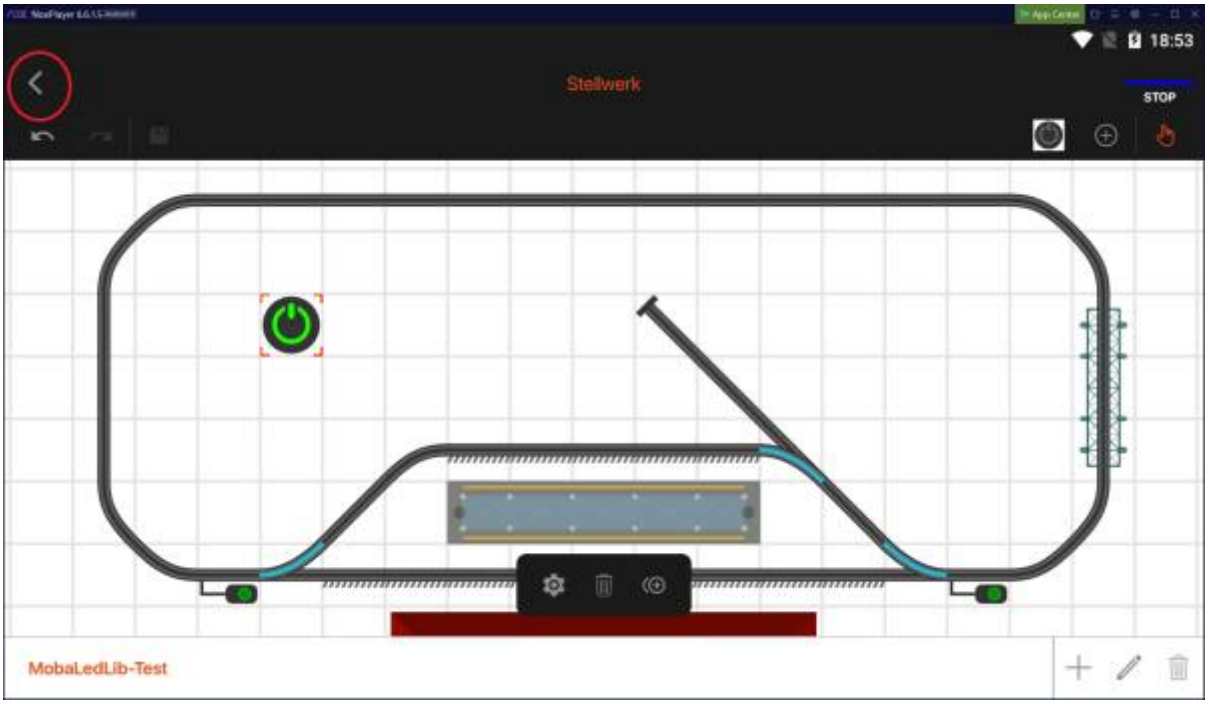
Durch antippen des Schalters kann man ihn ein- bzw ausschalten. Das Licht am ARDUINO sollte jetzt auch schalten.



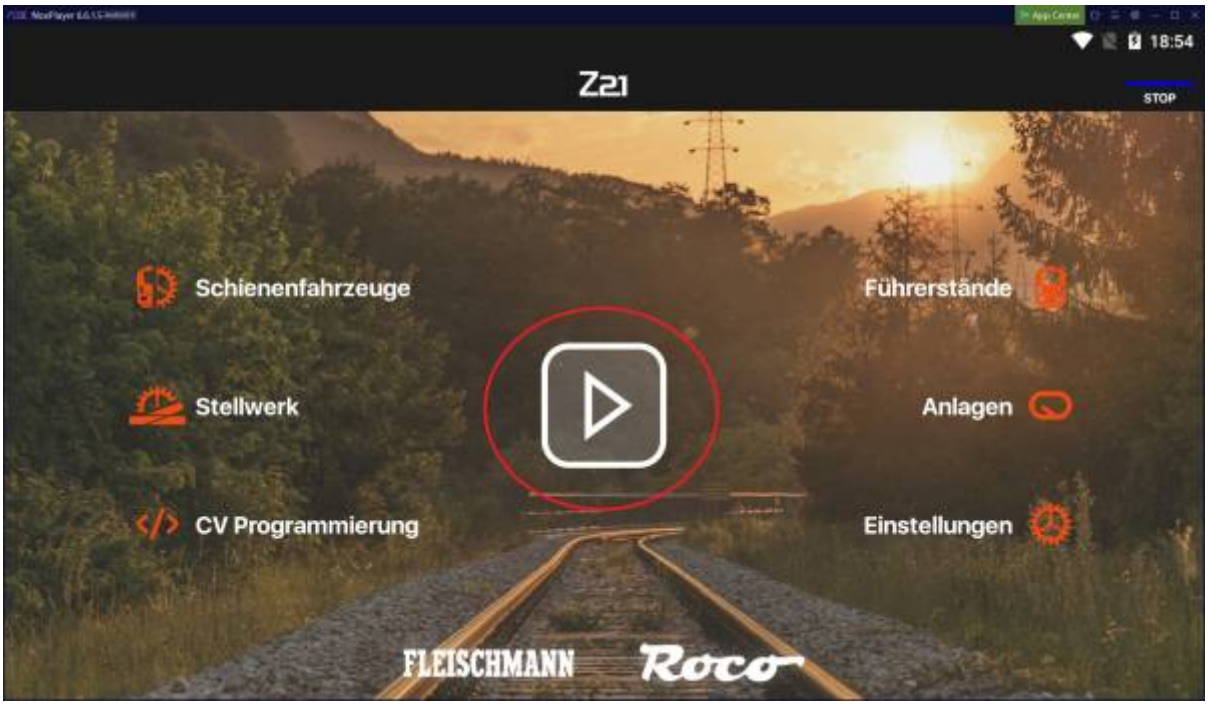
Im Z21 Monitor kann man die Schaltungen mit verfolgen.



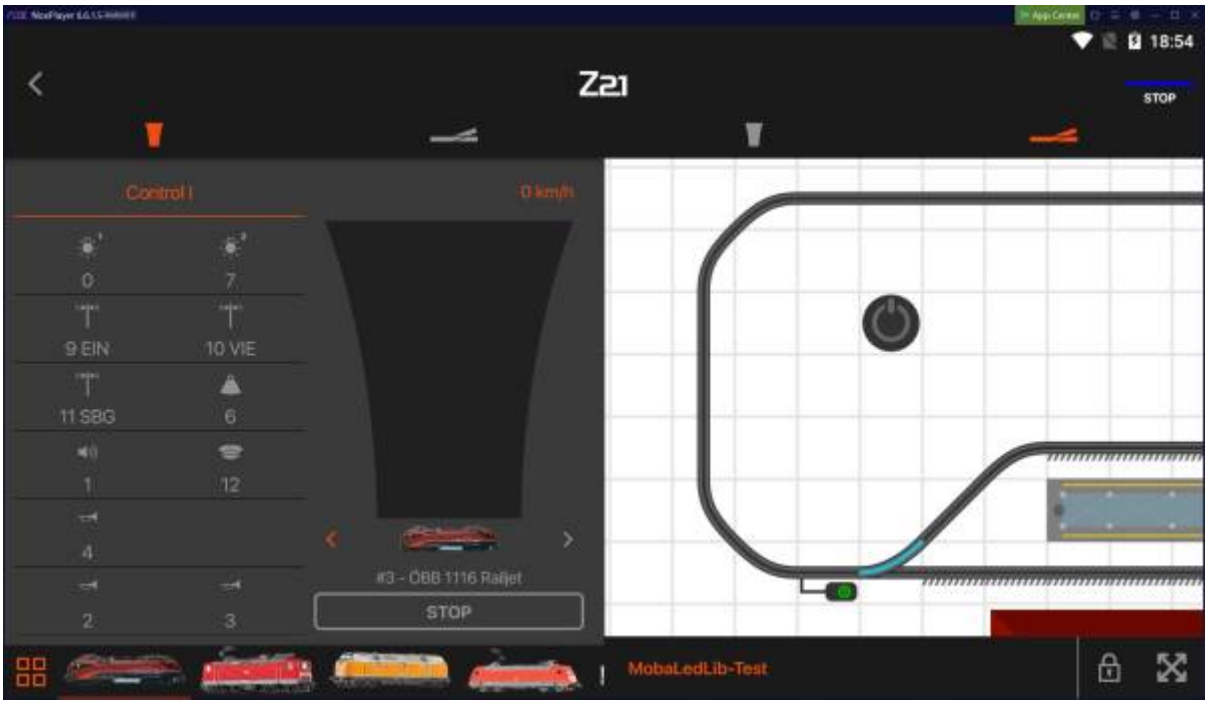
Die Stellwerksbearbeitung beenden durch antippen des „Pfeils nach links“



Das Stellwerk starten



Es erscheint das Stellwerk mit dem neuen Taster.



Genauso wie der hier als Beispiel eingerichtete Schalter kann natürlich auch jedes andere Symbol, wie z:B. Licht, Weiche, Signal, verwendet werden. Für die weiteren Schritte bitte der Z21-App Anleitung folgen.

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

Permanent link:
https://wiki.mobaledlib.de/anleitungen/allgemein/ansteuerung_mll_ueber_software?rev=1610567435

Last update: **2021/01/13 20:50**