Schalter und Taster einlesen mit der MobaLedLib

# Einleitung

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.

In diesem Dokument 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 erkannt. Darum können mit diesem Verfahren auch nur Taster eingelesen werden.

Bei allen 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. Dabei müssen aber unter Umständen die verwendeten Arduino Pins angepasst 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 Prog\_Generator benutzt werden. Dazu wird einfach der Name des Schalters in der Eingangsspalte der Tabelle eingetragen. Ein Schalter kann auch mehrere Aktionen auslösen. Über die „Logic“ Funktion können die Schalter auch mit DCC, Selectrix oder CAN Signalen verknüpft werden so dass eine Aktion entweder vom Taster oder per DCC gestartet werden kann.

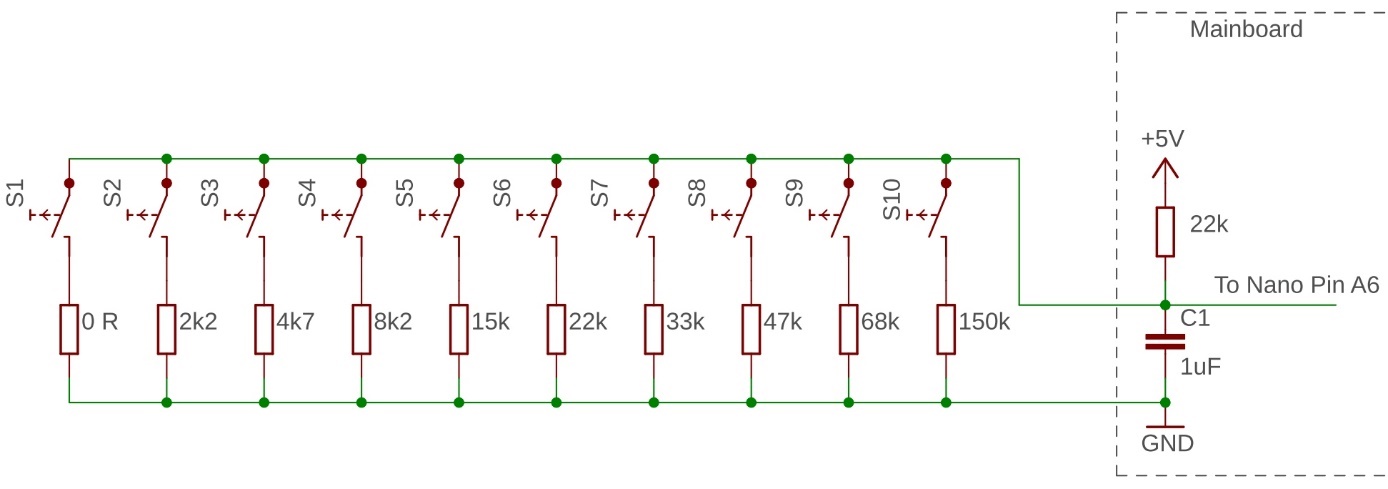
Anstelle von manuell betätigten Schaltern können auf diese Weise auch Ereignisse oder Zustände auf der Moba eingelesen werden. So kann man z.B. über einen Reed Kontakt, oder eine Lichtschranke erkennen, wenn ein Zug an einer bestimmten Position ist. Wichtig ist, dass die Ereignisse einen potentialfreien 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:

Es gibt vier verschieden Verfahren mit denen Schalter eingelesen werden:

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 70 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. Vermutlich können über 100 Taster eingelesen werden.  
   Als Eselsbrücke kann man sich „B“ = Border (Englisch Rand) merken.
3. 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. 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.

# Methode A: Analoge Taster

Bis zu zehn Taster können über eine zwei polige Leitung mit dem Arduino verbunden werden. Die beiden Leitungen werden dabei einfach von Taster zu Taster geführt. An jedem Taster sitzt ein anderer Widerstand über den der Arduino den Taster identifizieren kann. Die Bibliothek misst dazu einfach den entsprechenden Widerstand. Die Widerstände müssen entsprechend dem unten gezeigten Schaltbild gewählt werden. Der Taster 1 hat keinen Widerstand (Im Bild als 0 Ohm Widerstand dargestellt).

Zur Messung wird noch ein 22K Wiederstand auf der Hauptplatine und ein 1uF Kondensator benötigt. Diese Bauteile sind bei der aktuellen Version der Platine noch nicht vorhanden und müssen über das Lochraster Feld und Kabelbrücken nachgerüstet werden:

Bild: Zusätzlicher Widerstand und Kondensator zum einlesen analoger Taster

ToDo

ToDo

Bild: Kabelbrücken auf der Rückseite

## Mögliche Störungen

Die Messung der Widerstandswerte kann insbesondere bei großen Werten durch elektrische Störungen anderer Leitungen beeinflusst werden. Darum sollten die Leitungen nicht zu lang sein und nicht über längere Stecken parallel zu anderen Leitungen geführt werden. Ein verdrillen der Leitungen reduziert die Störeinflüsse ebenfalls. Der 1 uF Kondensator auf der Platine dient als Filter. Er verzögert die Messung aber auch ein wenig so dass die analogen Taster nicht ganz so schnell reagieren wie die im folgenden beschriebenen Schalter. In der Praxis stört das aber nicht.

## Verwendung im Excel Programm

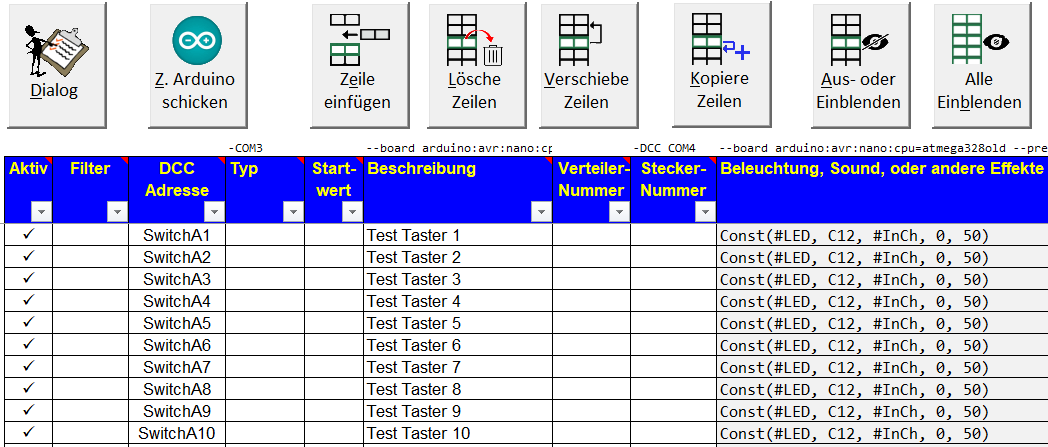
Im Excel Programm werden die analogen Schalter über den Namen „SwitchA“ gefolgt von einer Nummer angesprochen. Der erste Schalter hat dabei die Nummer 1.

Beispiel: SwitchA1

Wenn mehr als 10 Schalter verwendet werden sollen, dann können weitere analoge Eingänge des Arduinos zum einlesen von Tastern genutzt werden. Dazu müssen der Widerstand und Kondensator nachgerüstet werden. Über den Befehl „Set\_SwitchA\_InpLst()“ welcher im Expert Modus des Prog\_Generators verfügbar ist können weitere analoge Eingänge definiert werden.

Beispiel: // Set\_SwitchA\_InpLst(A6, A5, A4)

Mit dieser Konfiguration können die Taster getestet werden:



Damit wird jeden Taster eine LED zugewiesen. Die LED leuchtet solange der Taster betätigt ist und verlischt sobald der los gelassen wird.

Anstelle der LED könnte man hier auch einen Sound Befehl verwenden. So könnte man verschiedene Geräusche per Taster abrufen.

Zur Aktivierung von länger andauernden Effekten steuert man mit dem Taster z.B. eine „PushButton“ Funktion. Diese werden in einem Eigenen Kapitel erklärt.

# Methode B: Schalter am Anlagenrand

Im Miniatur Wunderland findet man an vielen Stellen Taster mit denen der Besucher bestimmte Aktionen per Knopfdruck auslösen kann. Das macht allen Besuchern viel Spaß. Dieses Konzept kann mit der MobaLedLib einfach umgesetzt werden. Die Taster können mit LEDs beleuchtet werden welche über Blinksequenz oder Farben den Zustand wiedergeben.

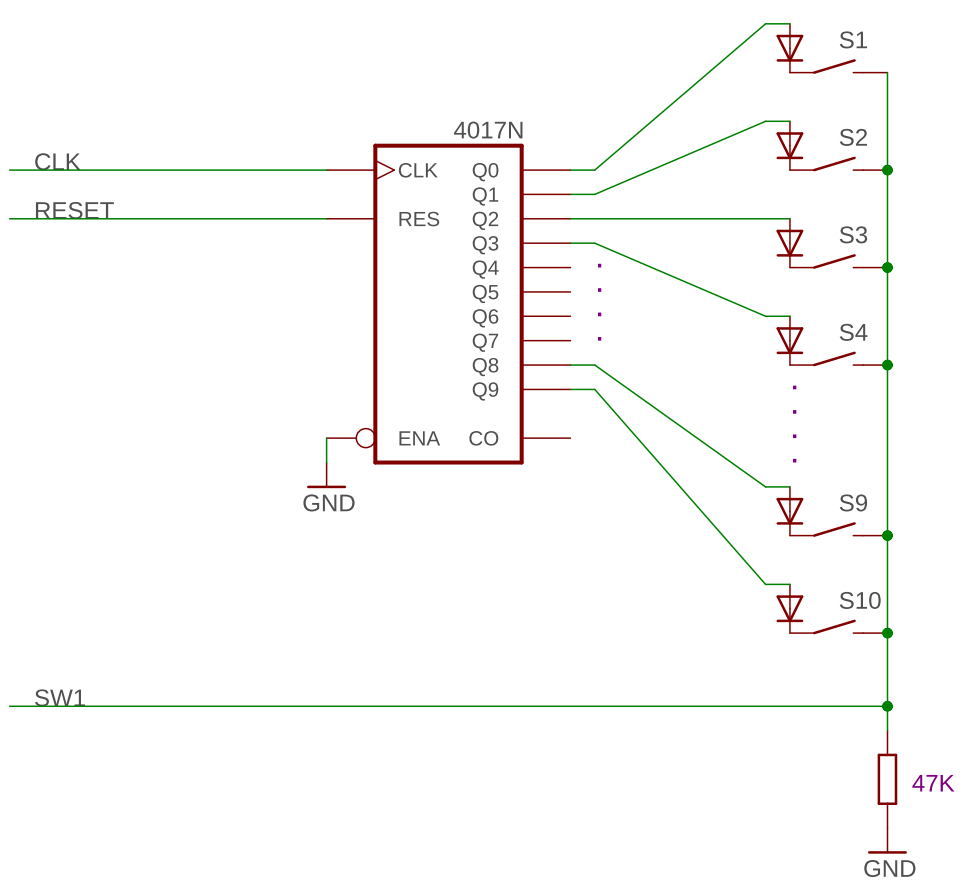
Über nur acht Kabel können fast beliebig viele Taster ausgewertet werden. Über das gleiche Flachkabel werden auch die LEDs der Taster angesteuert. Bis zu 10 Taster werden an eine PushButton\_4017 Platinen angesteckt. Mehrere dieser Platinen können hintereinandergeschaltet werden, wenn mehr Taster benötigt werden.

Wenn die Hauptplatine zentral positioniert ist, können zwei Flachkabel Stränge verwendet werden. Einer deckt die linke Seite der Anlage ab, der zweite die rechte Seite. Dazu wird der erste Strang an den Stecker „KEYBRD“ und der zweite Strang an „KEYBRD1“ angeschlossen.

## Prinzip der Schaltung

Das einlesen vieler Schalter über wenige Leitungen wird dadurch erreicht, dass die Schalter zeitlich nacheinander abgefragt werden. Das geschieht aber so schnell, dass eine Reaktion sofort erfolgt, wenn ein Schalter betätigt wird. Das Programm fragt dazu alle angeschlossenen Kanäle innerhalb von 100 Millisekunden einmal ab. Die Abfragefrequenz wird dabei automatisch an die Anzahl der Taster angepasst.

Zur Auswahl des abzufragenden Schalters wird das IC CD4017 verwendet.



Dieser Baustein ist ein Zähler welcher seine Ausgänge nacheinander aktiviert. Zu Beginn liegt am Ausgang Q0 +5V an. Wenn Schalter S1 aktiv ist, dann fliest über die Diode und den Schalter ein Strom und am Ausgang der Schaltung SW1 liegt ebenfalls 5V an. Das wird von Arduino eingelesen und abgespeichert.

Im nächsten Schritt wird Q1 aktiviert. Q0 ist jetzt wieder angeschaltet. Jetzt kann S2 erfasst werden. Wenn auch dieser Schalter geschlossen ist, dann fliest über seine Diode ein Strom und das kann wiederrum vom Arduino erkannt werden. Die Dioden verhindern dabei, dass das Signal von einem Ausgang rückwärts in einen anderen Ausgang fließen kann. Das würde passieren, wenn zwei Schalter gleichzeitig angeschaltet sind.

Den Befehl zum umschalten schickt der Arduino über die „CLK“ Leitung zum Zähler (4017). Der Reset Anschluss wird dazu verwendet, dass der Zähler wieder bei 0 beginnt. Das Programm aktiviert diesen Ausgang zu Beginn einer Messung.

Erfassung aller Schalter geschieht so schnell, dass der Benutzer keine Verzögerung beim Betätigen eines Schalters bemerkt. Dazu werden alle angeschlossenen Schalter innerhalb von 100 Millisekunden einmal abgefragt.

Mit dieser Schaltung kann man über drei Anschlüsse eines Arduinos zehn Schalter einlesen. Dabei können beliebig viele Schalter gleichzeitig eingeschaltet sein.

## LEDs in den Tastern

Im Miniatur Wunderland findet man viele Taster an Anlagenrand mit denen der Besucher bestimmte Aktionen auslösen kann. Bei jedem Taster befinden sich zwei LEDs über die angezeigt wird ob die Aktion bereit ist oder gerade läuft. Das ist gerade bei länger andauernden Aktionen nützlich, weil der Besucher daran erkennen kann ob er den Taster drücken kann oder ob er noch warten muss bis die Attraktion wieder bereit ist. Die Anzeige ist als Feedback wichtig.   
Bei der MobaLedLib können die LEDs noch mehr. Sie können so konfiguriert werden, dass sie beliebige Farben, Helligkeiten und Blinkmuster widergeben. Das kann Beispielsweise so genutzt werden:

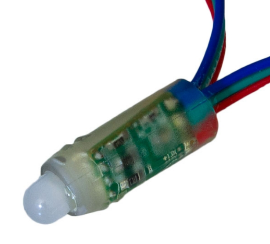
* Im Ausgeschalteten Zustand Leuchtet die LED schwach damit man den Schalter im abgedunkelten Raum findet
* Wenn die Aktion aktiv ist dann Blinkt die LED
* Bei Aktionen mit mehreren Zuständen blinkt die LED unterschiedlich oft. Wenn die Taste einmal gedrückt wird leuchtet die LED einmal kurz gefolgt von einer längeren Pause. Mit dem zweiten Druck auf den Taster Blinkt sie zweimal gefolgt von einer Pause. Auf diese Weise können bis zu 5 Zustände visualisiert werden.

Dieses Verhalten wird in der Konfiguration über den Befehl „PushButton\_w\_LED\_BL\_0\_x“ generiert. „x“ steht dabei für eine Zahl zwischen 1 und 5. Die Zahl entspricht der Anzahl der Zustände.

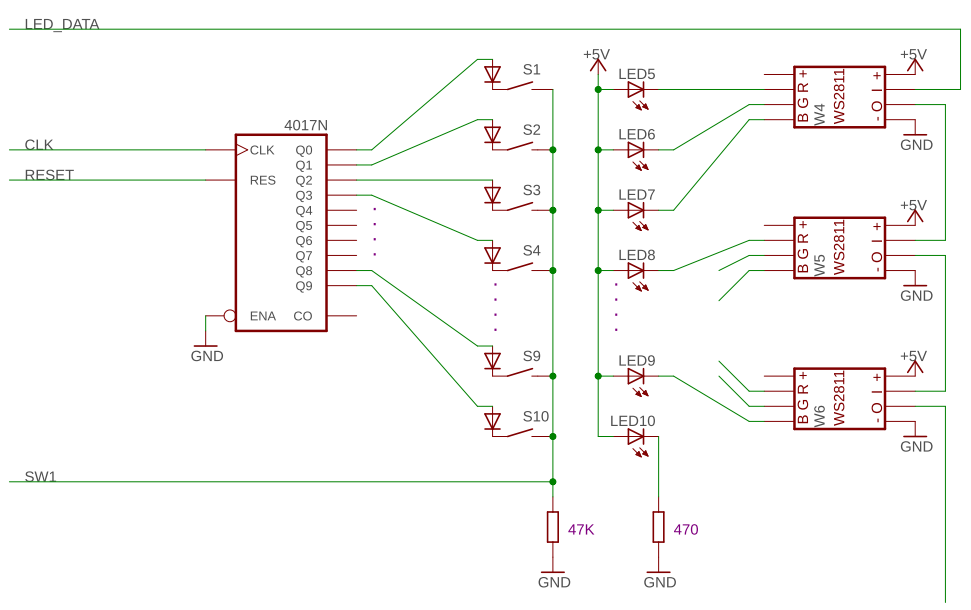
Zur Anzeige können entweder Taster mit eingebauter LED verwendet werden:



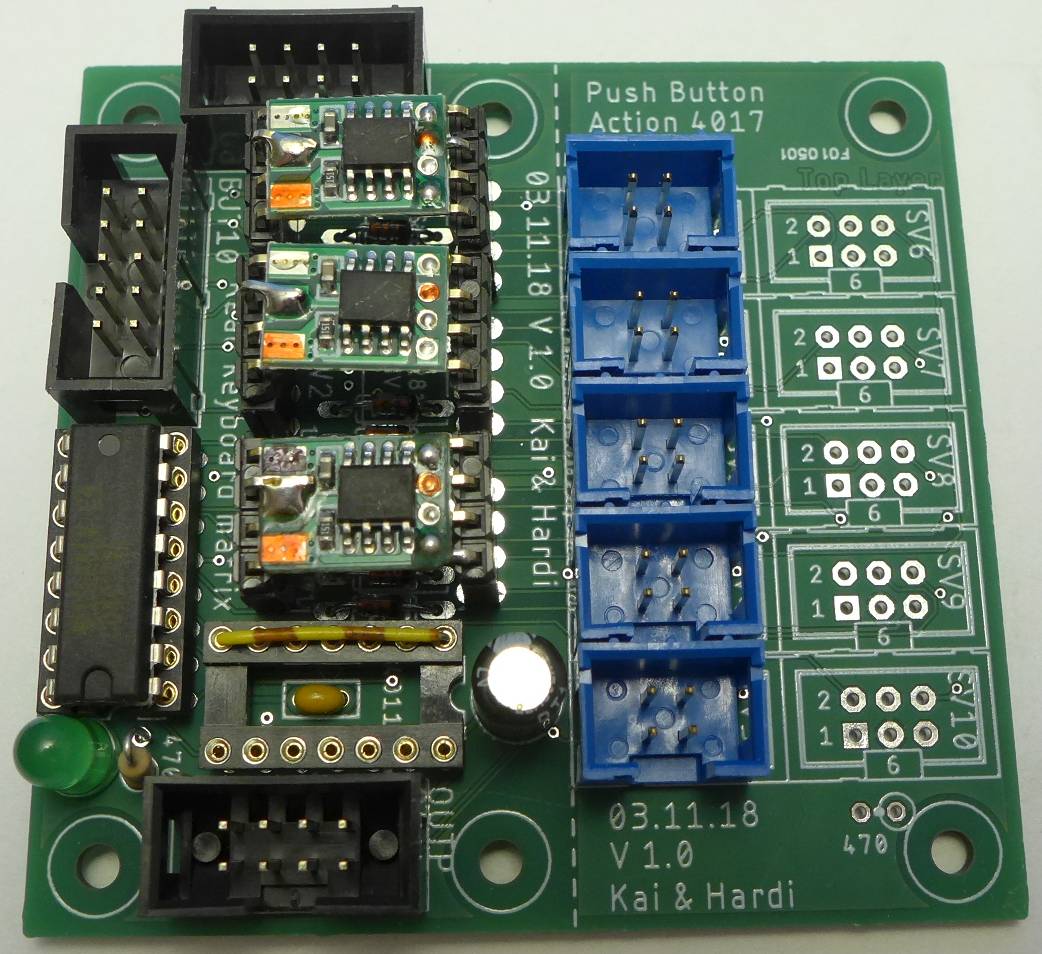
Oder separate (RGB) LEDs verwendet werden:



Auf der PushButton\_4017 Platine sind bereits Steckplätze für 3 WS2811 Module vorgesehen. Im Schaltbild sieht das so aus:



Damit werden jetzt 4 Signal Leitungen zum Arduino benötigt. Dazu kommen noch zwei Leitungen für die Versorgungsspannung. Auf der Platine werden 8-polige Stecker verwendet welche oben und unten auf der folgenden Platine zu sehen sind:



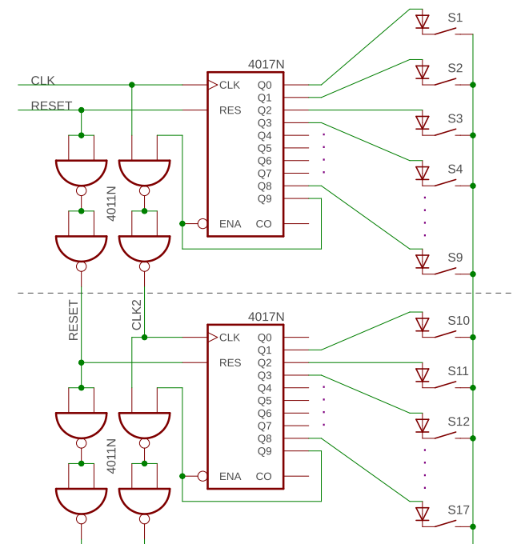
Der Stecker auf der oben ist der Eingang. Er wird über ein Flachkabel mit der Hauptplatine verbunden. An den Stecker an unteren Rand (OUTP) kann eine weitere PushButton\_4017 Platine angeschlossen werden. Damit hat kann man die Anzahl der Taster (fast) beliebig erweitern. Das ist ein ganz besonderes Feature der Schaltung.

In dem Beispiel sind 5 der insgesamt 10 möglichen Stecker zum Anschluss der Taster Bestückt. Die Idee dahinter ist, dass man diese Platinen an verschiedenen Stellen am Rand der Modellbahn platziert und über ein 8-poliges Kabel untereinander verbindet. An jede Platine werden dann mit kurzen Kabeln die Taster angeschlossen. Die Taster können über 4-polige oder 6-polige Buchsen angesteckt werden. Die 4-Polige Variante kann verwendet werden, wenn man nur eine (im Schalter eingebaute) LED verwendet. Sollen RGB LEDs verwendet werden, dann muss ein 6-poliger Wannenstecker benutzt werden.

## Kaskadierung der Taster Platinen

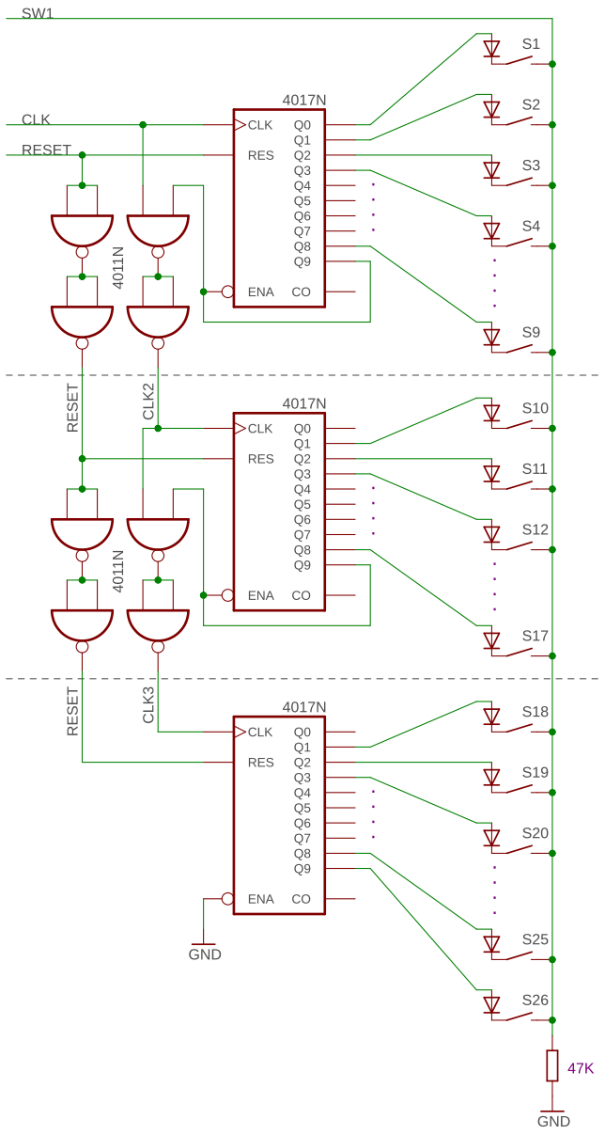
Zur Verwendung mehrere PushButton\_4017 Platinen wird ein weiteres IC (CD4011) benötigt. Dieser Baustein enthält vier NAND-Gatter. Das sind Komponenten mit denen zwei Eingängen UND Verknüpft werden und das Ergebnis anschließend Invertiert wird. Damit wird das Takt Signal vom Arduino erst dann weitergegeben wie Schalter auf der ersten Platine eingelesen sind.

Im Schaltbild sieht das so aus:



### Funktionsweise:

Das erste NAND-Gatter verknüpft das Taktsignal mit dem Ausgang Q9. Ein UND Gatter liefert erst dann eine 1 an seinem Ausgang, wenn beide Eingänge 1 sind. Damit wird das Taktsignal erst dann weitergeleitet, wenn Q9 1 ist. Das ist nach 9 Impulsen am Eingang des Zählers der Fall. Die Rückführung des Signals Q9 auf den Enable (ENA) Eingang des Zählers sorgt dafür, dass der Zähler gestoppt wird, wenn Q9 aktiv wird. Das Taktsignal wird über ein zweites Gatter des Bausteins invertiert und dann an den nächsten Zähler weitergegeben.

Die ersten 9 Kanäle werden damit vom ersten 4017 abgedeckt. Danach übernimmt der zweite 4017. Der Ausgang Q9 des ersten Bausteins bleibt so lange High bis die Zähler über den Reset Eingang zurückgesetzt werden. Das bedeutet aber auch, das mit der ersten Platine nur 9 Schalter eingelesen werden können, wenn ein zweites Modul folgt. Darum ist es auch nicht weiter schlimm, dass die PushButton\_4017 Platine nur 9 steuerbare LED-Ausgänge hat (Aufmerksamen Lesern ist vielleicht aufgefallen, dass die LED10 direkt über einen 470 Ohm Widerstand mit Masse verbunden ist und nicht über einen WS2811 gesteuert werden kann.).

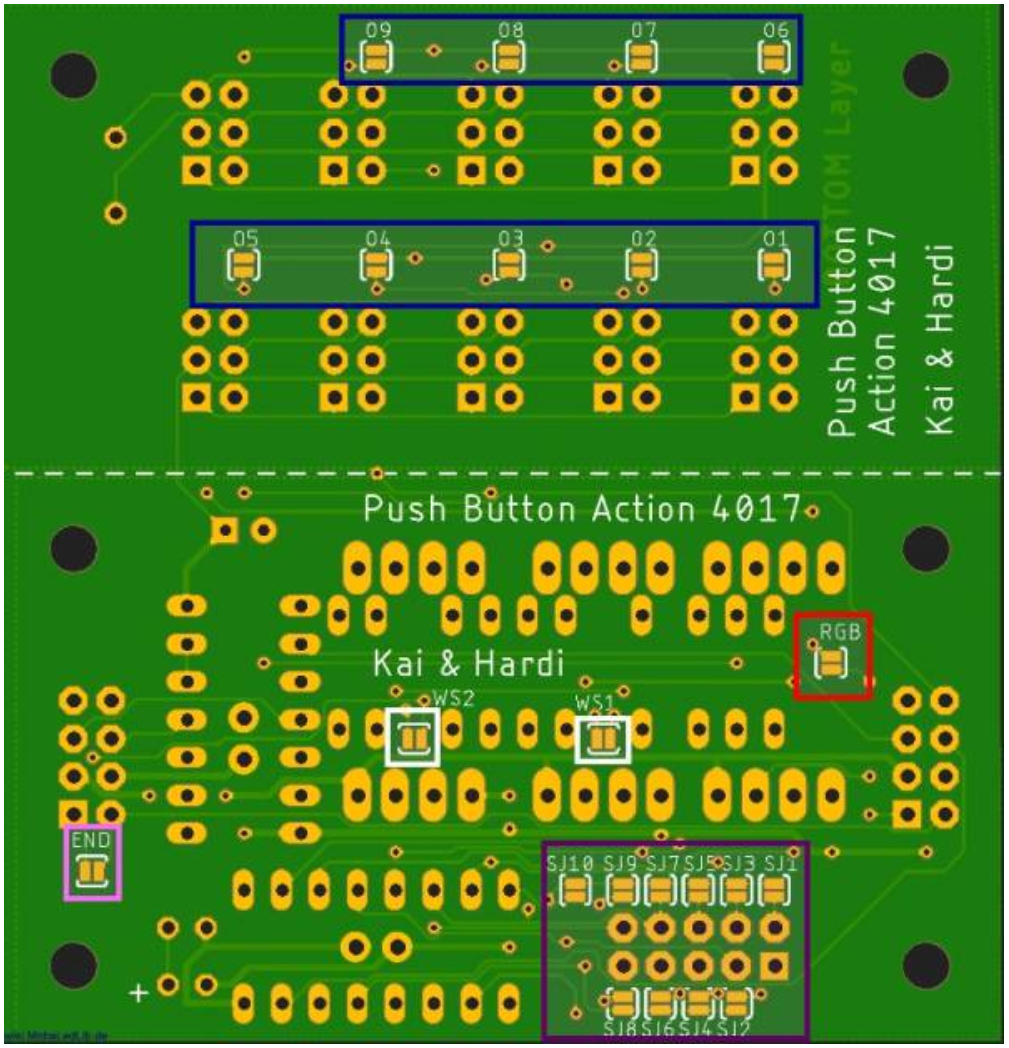
Die beiden linken NAND-Gatter invertieren das Reset Signal zweimal und verstärken das Signal dadurch lediglich.

Beim folgenden Zähler kann der erste Kanal ebenfalls nicht genutzt werden, weil dieser so lange aktiv ist bis die Schalter des vorangegangenen Moduls eingelesen sind. Erst mit dem 10. Taktimpuls wird der Ausgang Q1 des folgenden 4071 aktiviert. Im Schaltplan erkennt man, dass der Taster S10 am Pin Q1 des Zählers angeschlossen wird.

Wenn wie rechts gezeigt eine weitere Platine folgt dann entfällt auch der 9. Ausgang. Eine Mittlere Platine kann darum nur 8 Schalter auswerten. Am besten man bestückt die erste und letzte Taster Buchse nicht damit Fehlbedienungen ausgeschlossen sind. Wenn diese Kanäle versehentlich benutzt werden dann werden fälschlicherweise mehrere Taster als gedrückt gemeldet (Bei Q9 werden alle folgenden Taster gemeldet, Bei Q0 alle Vorangegangenen.)

## Taster Anzahl reduzieren

Wenn man nicht alle verfügbaren Kanäle einer Platine benutzen will, weil keine weiteren „Knopf Druck Aktionen“ in der Nähe vorgesehen sind kann man die Anzahl der abgefragten Schalter über Lötbrücken auf der Unterseite der Platine begrenzen. Das hat den Vorteil, dass Rechenzeit des Prozessors und Eingangskanäle gespart werden. Dazu setzt man einen der Lötjumper SJ1 – SJ10:

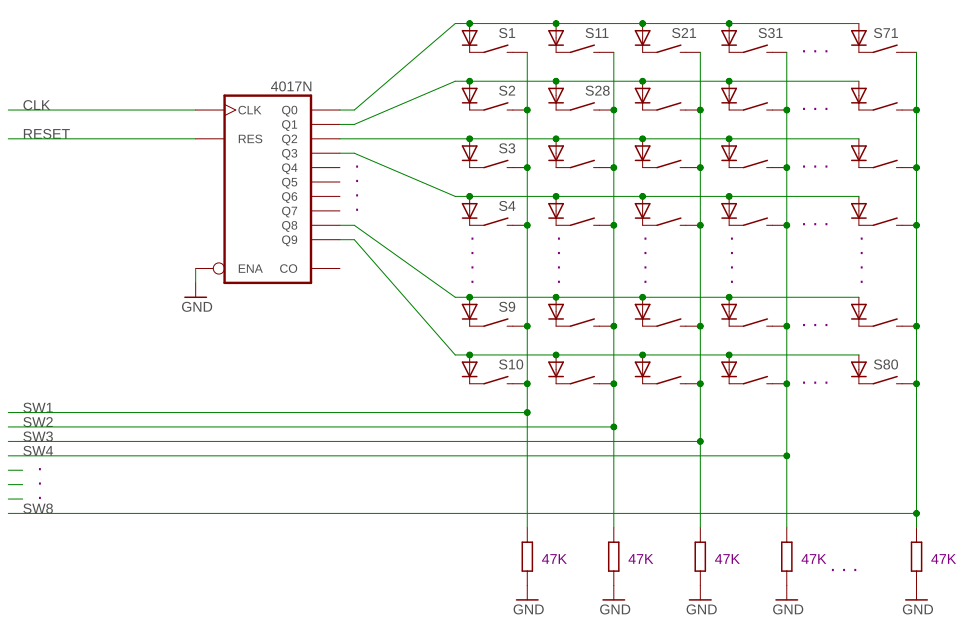


Achtung: Es muss immer mindestens ein Jumper gesetzt werden. Auch wenn nur eine Platine verwendet wird. Bei der ersten Version der Platine (V1.0) wurde der Jumper SJ10 vergessen. Dieser ist in dem ersten Bild der Platine den gestreiften Draht ersetzt. Er ist nur in dieser Version nötig.

Im Wiki findet man eine Ausführliche Anleitung dazu: https://wiki.mobaledlib.de/anleitungen/bauanleitungen/push\_button\_action\_300de#beispiele\_fuer\_die\_loetjumper\_sj1\_-\_sj10

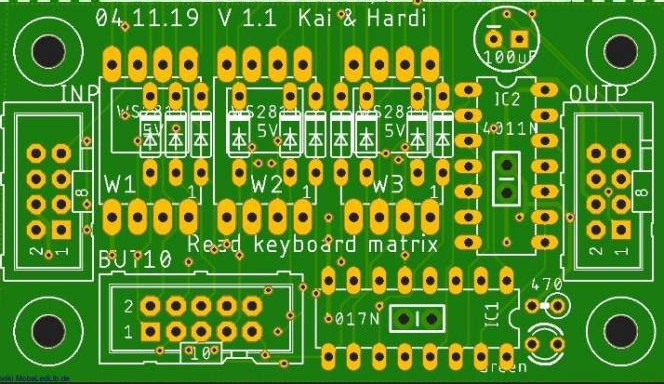
# Methode C: Viele Schalter für ein Weichenstellpult

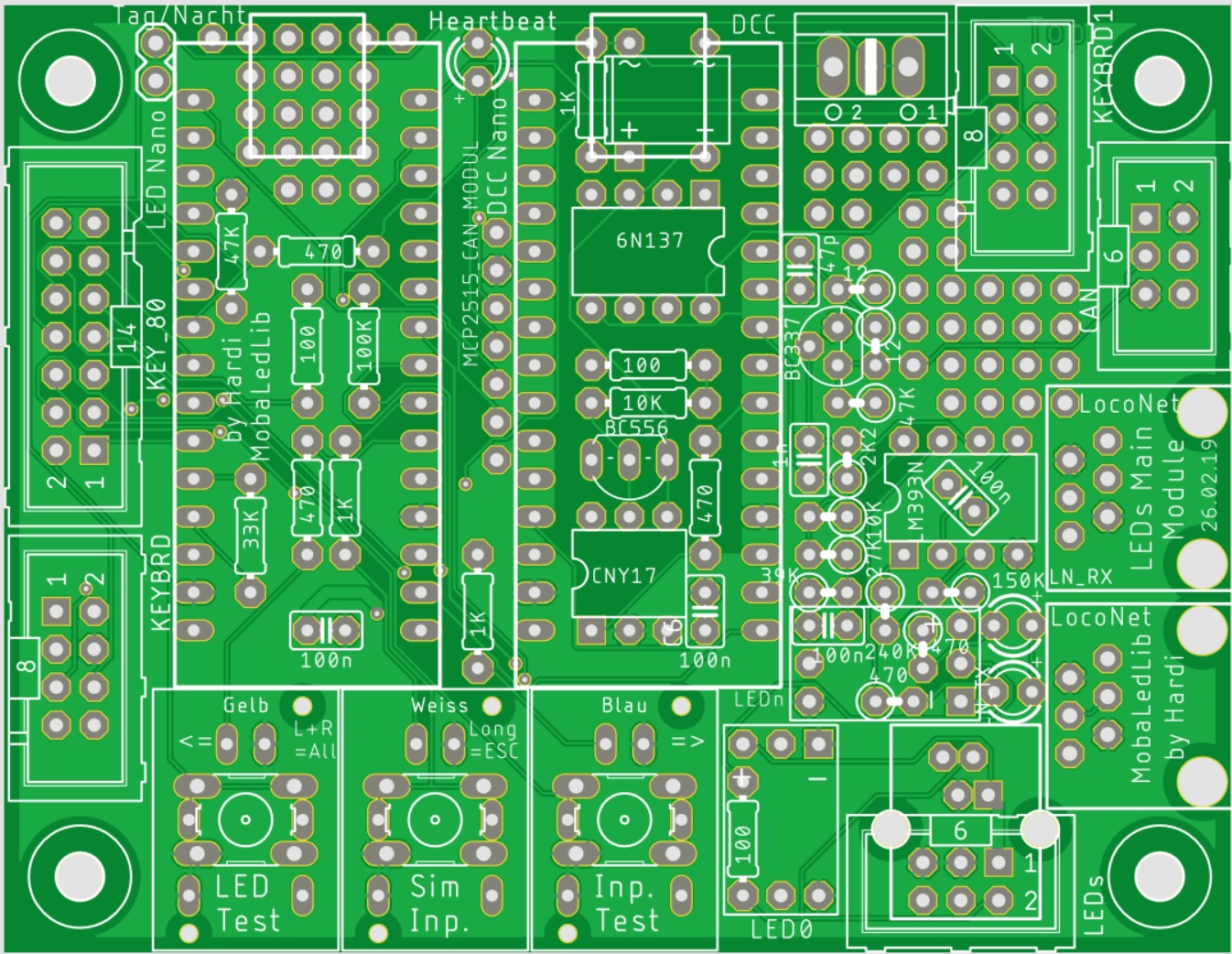
Das Verfahren zum einlesen von Tastern kann man noch erweitern. Dazu wird nicht nur eine Leitung zum einlesen der aktiven Schalter, sondern mehre Leitungen verwendet. Wenn man acht Eingänge am Arduino benutzt (WS1 – SW8) können 80 Schalter eingelesen werden.



Diese Variante ist zum einlesen von Schaltern und Tastern in einem Weichenstellpult gedacht.

Dazu wird die PushButton\_4017 Platine an der dafür vorgesehenen Stelle getrennt:

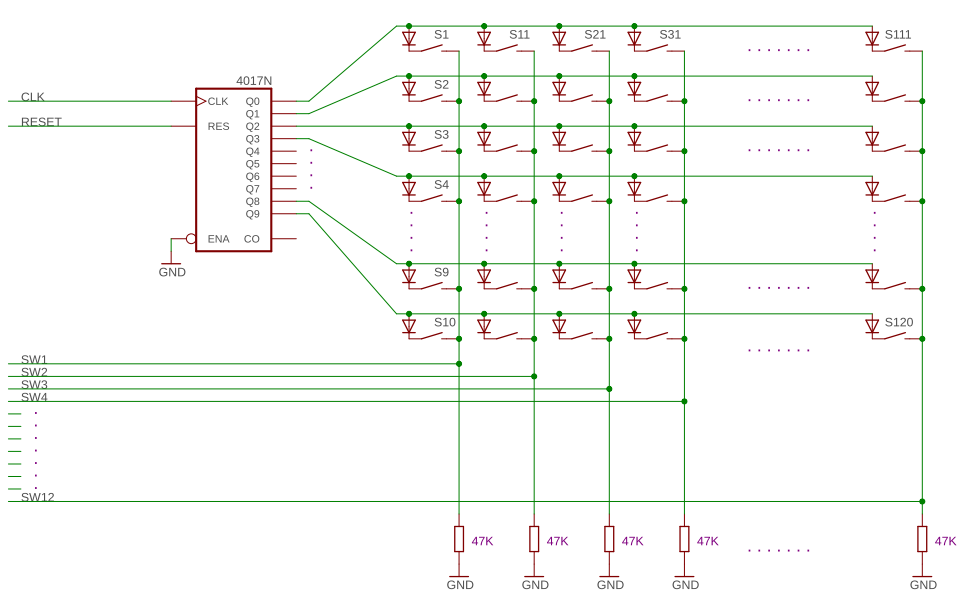


Die Schalter werden jetzt über den 10-poligen Stecker „BUT10“ auf der PushButton\_4017 Platine (Bild oben) und den Stecker „KEY\_80“ auf der Hauptplatine (Bild unten) angeschlossen. 

Über den 14-poligen Stecker auf der Hauptplatine können bis zu 12 Eingänge des Arduinos zum einlesen von Schaltern verwendet werden. Damit können 120 Schalter ausgewertet werden.

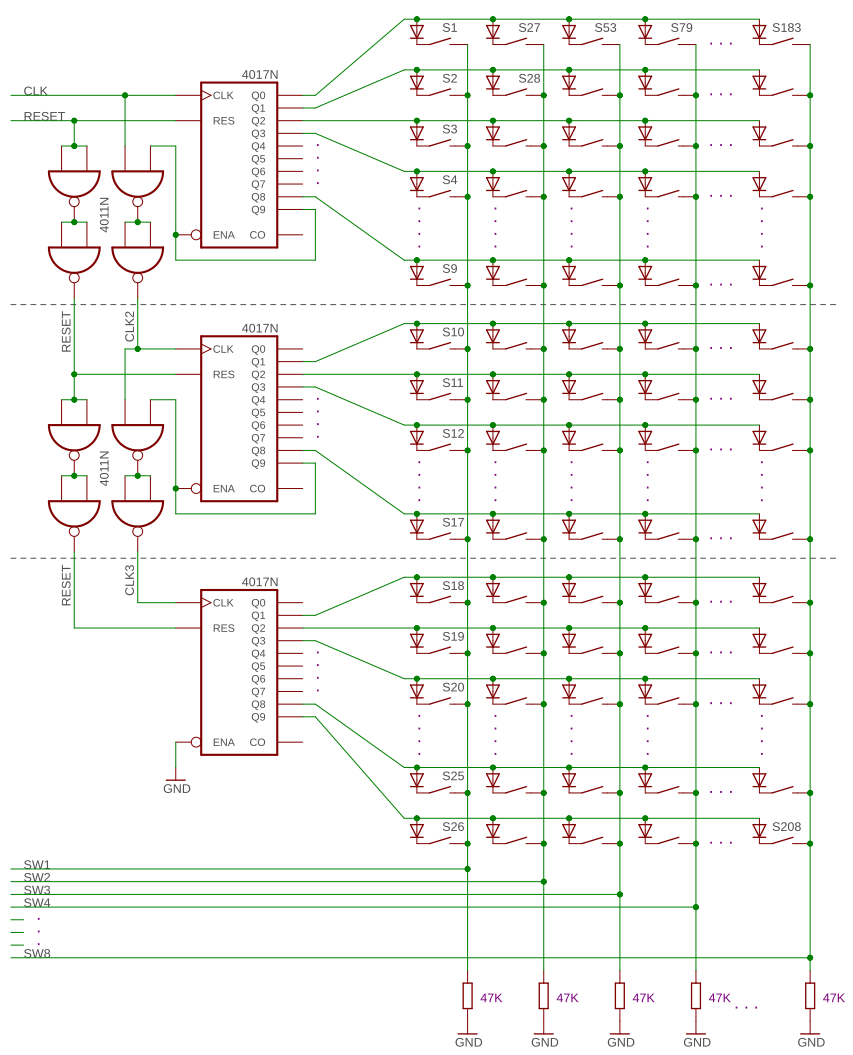
Wichtig ist, dass jede „Schalter“ Leitung zum Arduino mit einen 47K Widerstand auf Masse gezogen wird, wenn kein Taster betätigt wird. Diese Widerstände sind bis auf einen nicht auf der Hauptplatine vorhanden.

Da der Arduino relativ wenige Anschlüsse hat werden mache Pins mehrfach benutzt. Man muss sich entscheiden welche Pins für das Einlesen der Schalter verwenden will. Damit wird bestimmt welche Methoden zum einlesen von Schaltern verfügbar sind und wie viele Schalter eingelesen werden können. Im Folgenden wird darauf noch mal eingegangen.



Bei dieser Variante ist es genau so wie bei der Methode B möglich mehrere Platinen hintereinander zu schalten. Damit kann man entweder noch mehr Schalter einlesen oder man benötigt weniger Eingangskanäle am Arduino und hat dadurch Pins für andere Aufgaben zur Verfügung. Auch hier verliert man wieder durch das zusammenschalten der Zähler einige Kanäle. Trotzdem ist es wie das Folgende Bild zeigt mit 8 Eingängen am Arduino möglich über 200 Schalter auszuwerten.

Theoretisch könnte man noch mehr Platinen verwenden aber dann stößt man an die Grenzen das im Arduino verfügbaren Speichers. Über die MobaLedLib können maximal 250 Schalter eingelesen werden. Allerdings hat man dann keine Eingangsvariablen mehr für andere Aufgaben zur Verfügung.

Generell ist das einlesen von so vielen Schaltern möglich, aber dann wird es besser sein, wenn man die Daten über zusätzliche C++ Funktionen auswertet. Hier kann man die Schalter dann über geeignete Schleifen im Programm verarbeiten und muss nicht für jeden Schalter eine eigene Zeile im Prog\_Generator anlegen. Das Einbinden eigener C++ Funktionen in Kombination mit der automatisch per Excel generierten Konfiguration ist jeder Zeit möglich. 

# Methode D: Direkt auf der Hauptplatine vorhandene Taster

Die letzte und einfachste Methode zum einlesen von Tastern mit der MobaLedLib benötigt zunächst keine zusätzliche Hardware. Die drei auf der Hauptplatine vorhandenen Taster können über die Namen „SwitchD1“ bis „SwitchD3“ direkt im Prog\_Generator verwendet werden.

Auch hier kann die Anzahl der Eingänge erweitert werden. Dazu müssen der Bibliothek nur die benutzten Pins mitgeteilt werden. Die Anschlüsse welche für die drei Taster verwendet werden sind zusätzlich am Stecker „Key\_80“ verfügbar. Damit können anstelle der Taster auf der Hauptplatine auch externe Schalter angeschlossen werden. Diese können auch parallel benutzt werden. Auf diese Weise könnten bis zu 12 externe Schalter an die Platine angeschlossen werden. Aber das ist eine Extrem „teure“ Variante zum Lesen von Schaltern, da jeder Schalter einen eigenen der raren Pins belegt. Damit können dann nur noch 10 analoge Schalter (Methode A) eingelesen werden.   
Beim direkten Einlesen der Schalter können aber auch wie bei „B“ und „C“ alle Schalter gleichzeitig betätigt werden.

# Verwendung der Schalter im Prog\_Generator

Die Schalter werden über den Namen „Switch“ (Englisch Schalter) gefolgt von einem Buchstaben welcher den Typ beschreibt (A – D) und einer Nummer angesprochen.

Beispiel: „SwitchB7“

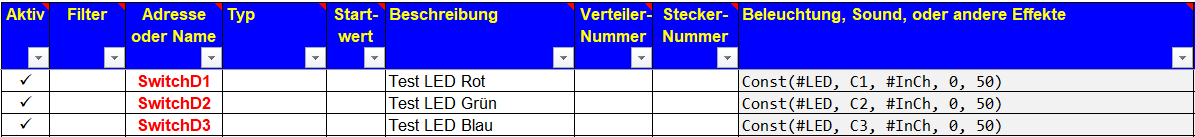
Es gibt folgende Schalter Typen:

* „SwitchA“ = Analoge Schalter
* „SwitchB“ = Schalter am Anlagenrand (B = Border)
* „SwitchC“ = Schalter im Weichenstellpult (C = Console)
* „SwitchD“ = Direkt auf der Hauptplatine vorhandene Schalter

Auf den Namen folgt eine Nummer zwischen 1 und N. Die maximale Anzahl der möglichen Schalter einer Gruppe hängt von deren Konfiguration ab. Das wird im Abschnitt “Konfiguration der benutzen Eingänge“ beschrieben.

Diese Namen können ganz einfach im Prog\_Generator verwendet werden. Momentan trägt man die gewünschte Schalter Bezeichnung einfach „von Hand“ in die Tabelle ein. Vielleicht wird das in einer Zukünftigen Version auch über einen Dialog möglich sein.

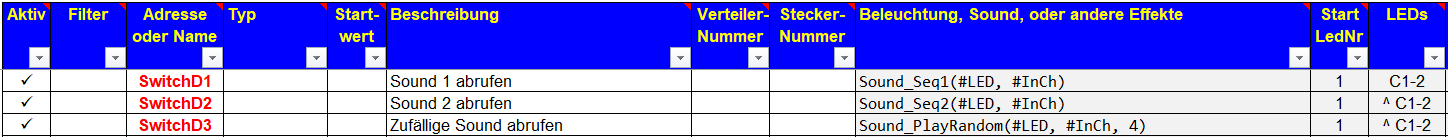
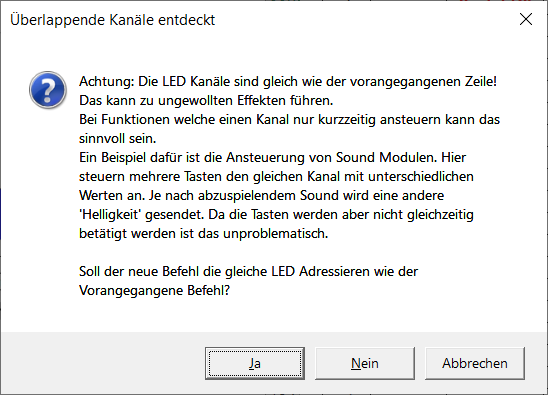
Die Schalter können an zwei verschiedenen Stellen verwendet werden.

1. In der Spalte „Adresse oder Name“  
   Anstelle einer (DCC) Adresse kann in der Spalte auch der Name eines Schalters angegeben werden. Hier werden die drei Taster auf der Hauptplatine zum Test mit jeweils einer LED verknüpft:  
     
   Achtung dabei handelt es sich nicht um die LEDs bei den Tastern, sondern um die drei Farben einer WS2812 RGB LED. Damit kann die Funktionsweise der Schalter schnell überprüft werden.
2. Die Schalter können aber auch in Spalte „Beleuchtung, Sound, oder andere Effekte“ benutzt werden. Hier gibt es einige Funktionen welche mehrere Eingänge haben. Ein Beispiel dafür ist die „Logic“ Funktion. Sie ist ein mächtiges Werkzeug mit dem man beliebige logische Verknüpfungen implementieren kann (Sie erscheint im Auswahl Dialog wen der Experten Mode aktiviert ist). Das kann man dazu nutzen eine Funktion per DCC ***oder*** per Taster zu steuern:  
     
   Hier werden gleich drei neue Features der MobaLedLib gezeigt.   
   Zunächst einmal die Verwendung des Schalters „SwitchD1“ in der Effekt Spalte. Dieser Schalter wird „Oder“ verknüpft mit dem Eingang der Funktion. Er wird über den speziellen Namen „#InCh“ abgerufen. Im Arduino wird er durch den DCC Kanal 1 / Rot ersetzt.   
   Die dritte Neuerung ist die Verwendung von eigenen Variablen. Das Ergebnis der „Logic“ Funktion wird in die Benutzer definierte Variable „LogicRes“ geschrieben. Der Name ist frei wählbar, muss aber den C++ Konventionen entsprechen und darf noch nicht im Programm vorhanden sein. Wenn man Beispielsweise als Ergebnis Variable den Namen „Logic“ verwenden würde dann erscheint diese Fehlermeldung:   
   

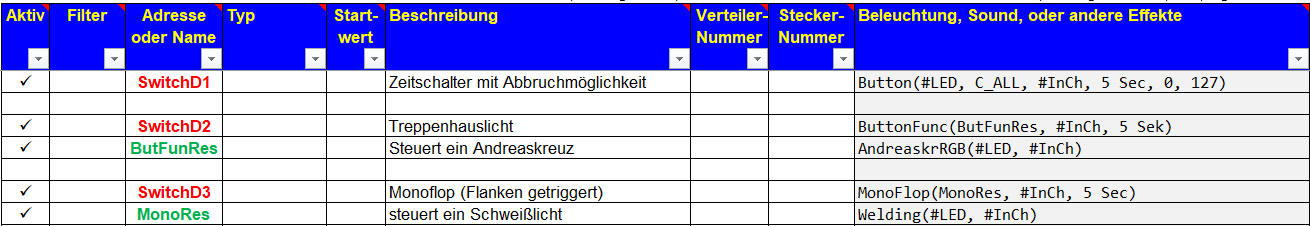
## Beispiele mit Schaltern

In diesen Abschnitt werden einige kleine Beispiele aus der Praxis vorgestellt. Die Beispiele verwenden die Tasten auf der Hauptplatine „SwitchD1“ bis „SwitchD3“, weil diese Tasten immer verfügbar sind. Wenn man über die entsprechenden Zusatztasten und Platinen verfügt, dann kann man genauso jeden anderen Tastentyp verwenden.

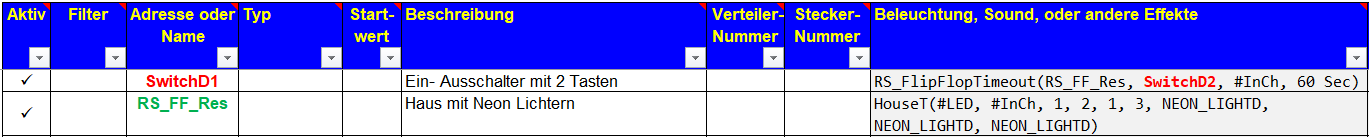
### Abrufen eines Sounds per Taster

Anstelle der in den vorangegangenen Beispielen verwendeten LED kann man den Sound Befehl verwenden. So kann der Benutzer per Knopfdruck einen Sound abrufen. Wenn man so wie in dem Beispiel mehrere Sound Befehle nacheinander eingibt, dann kommt die folgende Abfrage:  
Welche man mit „Ja“ beantwortet. Das sorgt dafür, dass sich alle drei Zeilen auf den gleichen LED-Kanal beziehen. Damit ist „Start LedNr“ bei allen drei Zeilen gleich und in Spalte „LEDs“ wird ein „^“ eingetragen was bedeutet, dass die gleiche LED-Nummer wie die letzte Zeile benutzt wird.

### Zeitschalter



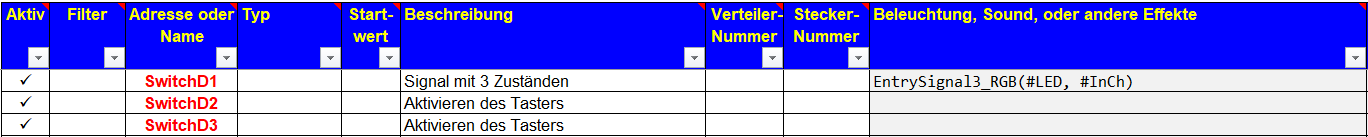
### Taster als Ein- und Ausschalter



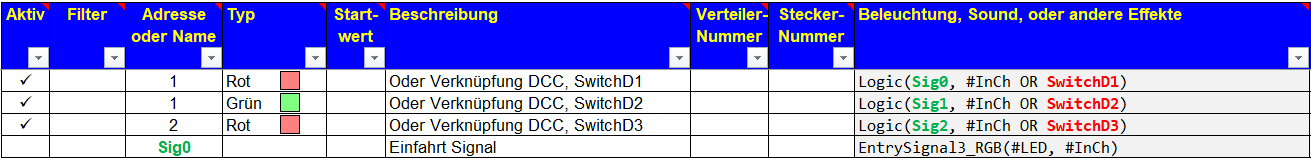
### Knopf Druck Aktion

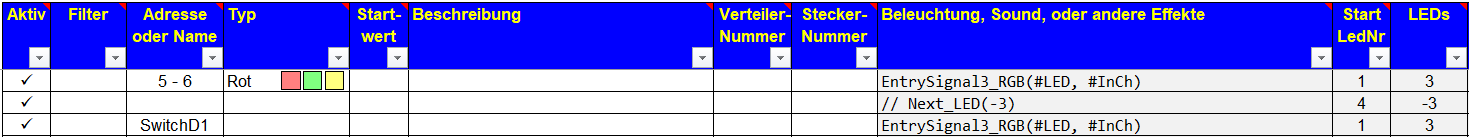


### Signal Ansteuerung



### Signal Ansteuerung mit DCC oder Tastern





### Burg Illumination

## Konfiguration der benutzen Eingänge

# Eigene Variablen