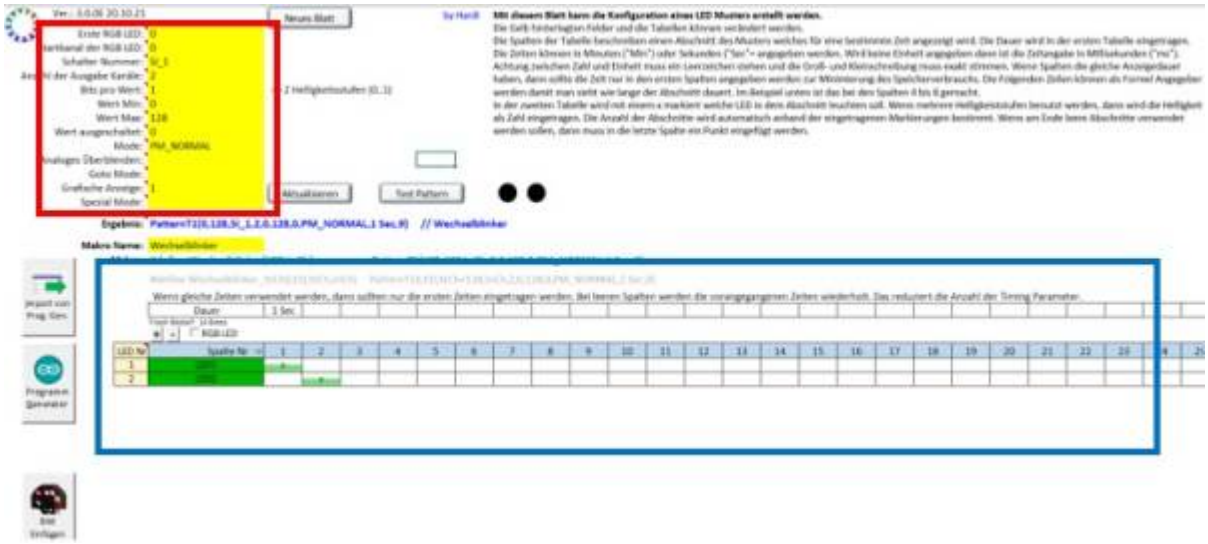
Der Pattern_Configurator

Diese Anleitung befindet sich aktuell noch in Bearbeitung,
bietet aber zum jetzigen Zeitpunkt schon eine große Hilfe.
— *Michael 2022/04/10*

Was ist der Pattern_Configurator?

Der Pattern_Configurator ist neben dem Programm Generator das vielleicht mächtigste Werkzeug der MobaLedLib. Mit dem Pattern_Configurator lassen sich beliebige Beleuchtungsmuster erstellen. Dadurch können neben den vorhandenen Lichteffekten des Programm Generators beliebige neue Effekte erstellt werden. Wenn man z.B. ein spezielles Blaulicht, ein spezielles Lichtsignal, eine bestimmte Abfolge von Soundbefehlen oder auch Bewegungsabläufe von Servos benötigt, kann man dazu den Pattern_Configurator verwenden. Auch die meisten im Programm Generator enthalten Makros wurden ursprünglich mit dem Pattern_Configurator erstellt.

Der Aufbau des Programms



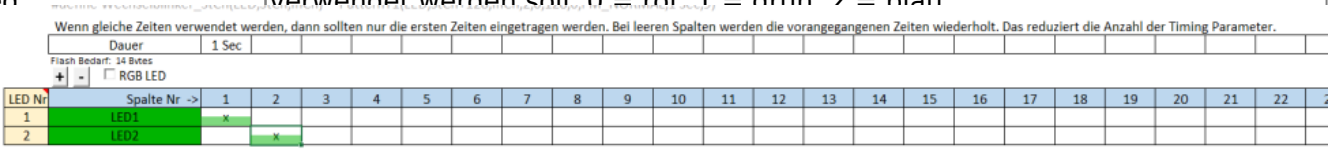
Im oberen Teil des Programms befinden sich die Einstellungen (rot markiert). Hier können verschiedene Parameter eingestellt werden. Diese werden im nächsten Abschnitt genauer erklärt. Im unteren Teil befindet sich eine Tabelle (blau markiert), in der die Beleuchtungseffekte eingetragen werden können.

Die Einstellungen

Im oberen Teil des Pattern_Configurators befinden sich verschiedene Einstellungen. Diese werden im Folgenden genauer beschrieben.

Konfiguration von Beleuchtungseffekten

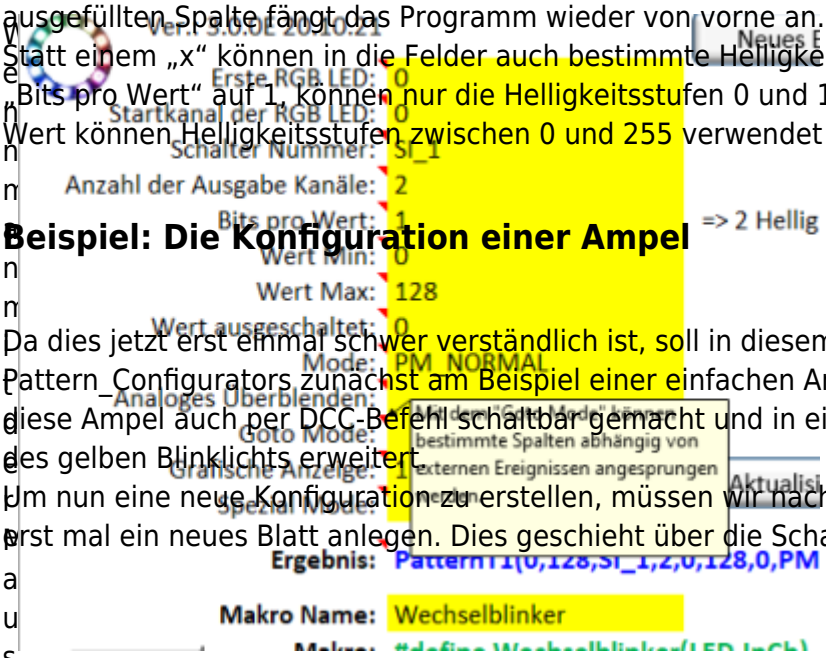
In der unteren Tabelle des Programms können die Beleuchtungseffekte eingetragen werden. Dafür werden nebeneinander die einzelnen Schritte eingetragen. Man muss also seinen Beleuchtungseffekt in einzelne Schritte unterteilen. Diese Schritte werden in einzelne Spalten nebeneinander in die Tabelle eingetragen und laufen dann nacheinander ab. Über der Tabelle kann in der Zeile „Dauer“ eine Dauer für jeden Schritt eingetragen werden. Für einen Wechselblinker sieht das so aus:



Zunächst soll für eine Sekunde Led 1 leuchten, Led 2 ist ausgeschaltet. Dann soll für eine Sekunde Led 2 leuchten, während Led 1 ausgeschaltet ist.
Dieses Wechselblinker Beispiel ist auch direkt beim Öffnen des Pattern_Configurators als Beispiel enthalten.

Oben ist in der Zeile Dauer in der ersten Spalte „1 Sec“, also eine Sekunde eingetragen. Darunter steht in der Zeile „LED1“ ein „x“. Ein „x“ bedeutet, dass die Led mit der oben angegebenen, maximalen Helligkeit eingeschaltet werden soll. Wird die Zelle leer gelassen, eine „0“ oder ein „.“ eingetragen, wie bei „LED2“, ist die Led ausgeschaltet.
In der zweiten Spalte wurde das Feld für die Dauer leer gelassen, dadurch wird die vorangegangene Zeit wiederholt. Das Feld für „LED1“ ist leer, die Led ist also ausgeschaltet, im Feld für „LED2“ ist ein „x“ eingetragen, die Led leuchtet also.
Die Dauer wird dabei normalerweise in Millisekunden angegeben, für Sekunden oder Minuten können „Sec“, „Sek“, „sec“, „sek“ oder „Min“ eingetragen werden.
Die einzelnen Spalten laufen nacheinander ab. Das Programm beginnt also ganz links und springt nach einer Sekunde dann in die nächste Spalte und aktiviert die Leds passend zur eingetragenen Konfiguration. Danach springt das Programm in die dritte Spalte und so weiter. Nach der letzten ausgefüllten Spalte fängt das Programm wieder von vorne an.
Statt einem „x“ können in die Felder auch bestimmte Helligkeitswerte eingetragen werden. Steht „Bits pro Wert“ auf 1, können nur die Helligkeitsstufen 0 und 1 eingetragen werden. Bei 8 Bits pro Wert können Helligkeitsstufen zwischen 0 und 255 verwendet werden.

Bezeichnung	Erklärung
Erste RGB Led	Dieser Wert gibt die Nummer der verwendeten Led in der Kette an. Bei der Verwendung des Programm Generators wird dieser Wert allerdings nicht verwendet und kann unbeachtet bleiben.
Startkanal der RGB Led	Gibt an, welcher Kanal eines WS2811 Moduls für die erste WS2811 Led verwendet werden soll. 0 = rot, 1 = grün, 2 = blau
Kanäle	betrachtet.
Bits pro Wert	Dieser Wert gibt die Anzahl der verfügbaren Helligkeitsstufen für die Leds an. (1 Bit = 2 Helligkeitsstufen, 8 Bits = 256 Helligkeitsstufen, beliebige Zwischenwerte sind ebenfalls möglich)
Wert Min	Dieser Wert gibt die minimale Helligkeit der Leds an.
Wert Max	Dieser Wert gibt die maximale Helligkeit der Leds an.
Wert ausgeschaltet	Dieser Wert gibt die Helligkeit der Leds an, wenn das Pattern_Configurator Makro z.B. mittels einer DCC Adresse, abgeschaltet wurde.
Mode	Mit dieser Einstellung können verschiedene Modi aktiviert werden, welche das Makro beeinflussen. Für den Einstieg sind diese aber nicht relevant und der Mode sollte auf PM_Normal stehen. Eine genauere Beschreibung dazu ist im Kapitel „Mode“ zu finden.
Analoges Überblenden	Wird hier eine 1 eingetragen, wird das analoge Überblenden aktiviert. Dabei verändert sich die Helligkeit einer Led langsam bis zum vorgegebenen Wert.
Goto Mode	Wird der Goto Mode aktiviert, können einzelne Spalten durch externe Ereignisse wie einen Tastendruck direkt angesprungen werden.
Grafische Anzeige	Aktiviert die grafische Anzeige der Lichteffekte in der Tabelle.
Spezial Mode	Wird hier ein „C“ eingetragen, wird der Charlieplexing Modus für die Verwendung von Charlieplexing aktiviert.



Beispiel: Die Konfiguration einer Ampel

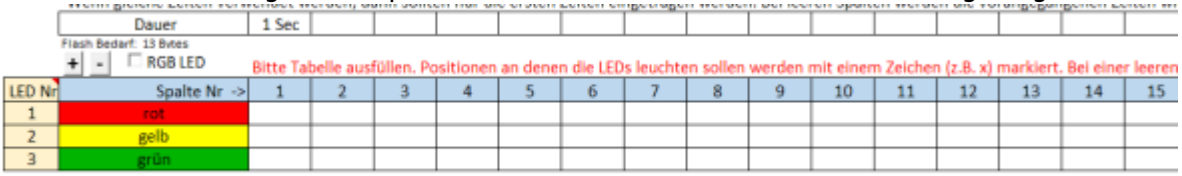
Da dies jetzt erst einmal schwer verständlich ist, soll in diesem Abschnitt die Verwendung des Pattern_Configurators zunächst am Beispiel einer einfachen Ampel erklärt werden. Weiter unten wird diese Ampel auch per DCC-Befehl schaltbar gemacht und in einem weiteren Schritt um die Funktion des gelben Blinklichts erweitert.
Um nun eine neue Konfiguration zu erstellen, müssen wir nach dem Starten des Pattern_Configurators erst mal ein neues Blatt anlegen. Dies geschieht über die Schaltfläche „Neues Blatt“.

über die Einstellung im Pattern_Configurator fährt, wird eine Erklärung angezeigt (Mauszeiger auf rotes Dreieck bewegen).

Nach dem Auswählen der Schaltfläche erscheint die Frage, ob wir die Einstellungen des aktuellen Blattes übernehmen möchten. Diese Frage beantworten wir mit „Nein“. Daraufhin kann man dem Blatt noch einen Namen geben, z.B. „Ampel“. Anschließend erhält man ein leeres Blatt des Pattern_Configurators, in dem wir unsere Ampel konfigurieren können. Zunächst einmal müssen wir die Einstellungen im oberen Teil anpassen. Die meisten Werte können wir bei der Standardeinstellung belassen, nur die Anzahl der Ausgabe Kanäle müssen wir anpassen. In diesem Fall soll eine Ampel mit drei Leds konfiguriert werden, also muss in das Feld eine 3 eingetragen werden. Außerdem kann das Feld „Wert Max“ angepasst werden, um die Helligkeit der Leds zu ändern. Alle anderen Einstellungen sind erst mal nicht relevant. Als nächstes sollte man sich den genauen Ablauf überlegen. Die Zeiten können dabei natürlich beliebig verändert werden. Bei einer Ampel sieht das so aus:

- Zunächst soll die rote Led für 5 Sekunden leuchten. Die anderen Leds sind dabei ausgeschaltet.
- Danach sollen die gelbe und die rote Led für 1 Sekunde leuchten.
- Danach soll die grüne Led für 5 Sekunden leuchten.
- Anschließend soll die gelbe Led für 1 Sekunde leuchten.
- Danach beginnt der Ablauf wieder von vorne, also mit der roten Led.

Das muss nun in den Pattern_Configurator übertragen werden. Zur besseren Übersichtlichkeit kann man an den Anfang der Zeilen für die Leds auch noch „rot“, „gelb“ und „grün“ schreiben und die Zellen wie von Excel gewohnt einfärben. Die dort gewählte Farbe wird mit verringerter Deckkraft automatisch in die folgenden Spalten übernommen, sobald dort Werte eingetragen werden.



Nun wird der Ablauf in die Tabelle eingetragen. Für die erste Dauer wird dafür wie vorher festgelegt 5 Sekunden eingetragen. Darunter wird bei der roten Led ein „x“ eingetragen, da diese Led leuchten soll. Die Felder für gelb und grün werden freigelassen, da die Leds nicht leuchten sollen.

Wenn gleiche Zeiten verwendet werden, dann sollten nur die ersten Zeiten eingetragen werden. Bei leeren Spalten werd

Dauer	5 Sec										

Flash Bedarf: 14 Bytes

+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4	5	6	7	8	9	10	11
1	rot	x										
2	gelb											
3	grün											

Anschließend sollen die rote und die gelbe Led für eine Sekunde leuchten, man trägt also bei der Dauer „1 Sec“ und bei der roten und gelben Led je ein „x“ ein.

Wenn gleiche Zeiten verwendet werden, dann sollten nur die ersten Zeiten eingetragen werden. Bei leeren Spalten werden die vor

Dauer	5 Sec	1 Sec										

Flash Bedarf: 16 Bytes

+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4	5	6	7	8	9	10	11	12
1	rot	x	x										
2	gelb		x										
3	grün												

Das Gleiche macht man nun noch für die nächsten Schritte:

Wenn gleiche Zeiten verwendet werden, dann sollten nur die ersten Zeiten eingetragen werden. Bei leeren Spalten werden

Dauer	5 Sec	1 Sec										

Flash Bedarf: 17 Bytes

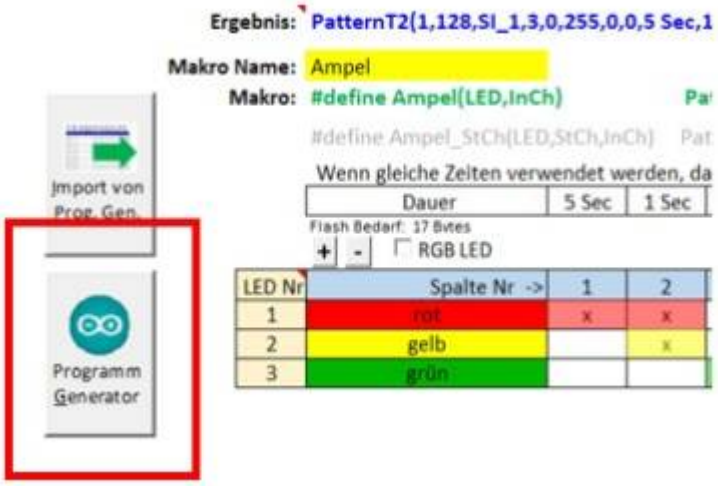
+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4	5	6	7	8	9	10	11
1	rot	x	x									
2	gelb		x		x							
3	grün			x								

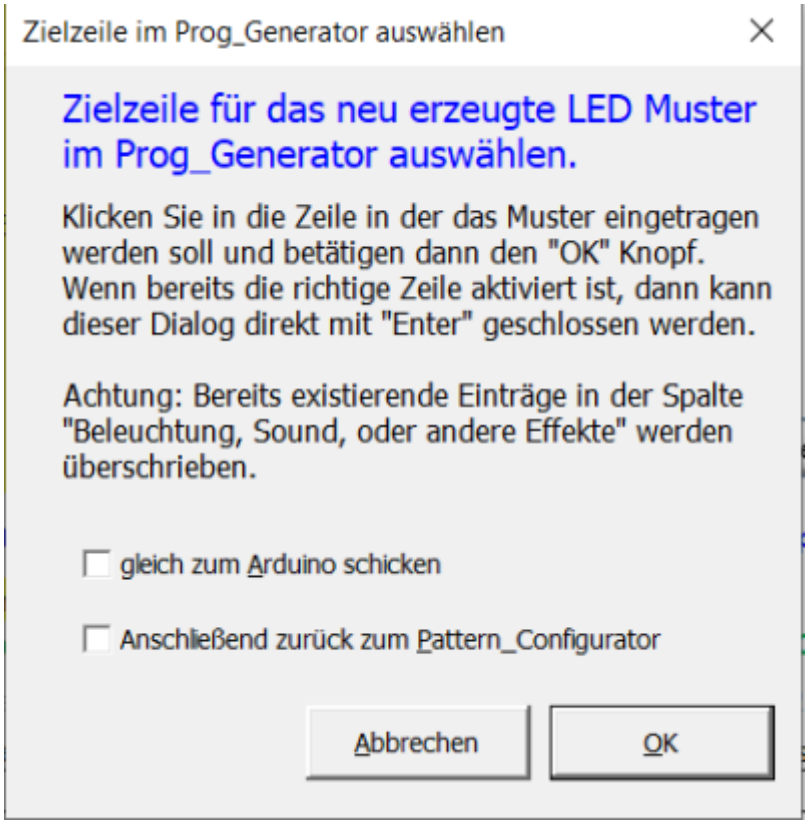
Die Zeiten müssen dabei nicht wieder eingetragen werden, da diese gleich wie bei den ersten Schritten sind und dadurch automatisch wiederholt werden. Nun ist die erste Konfiguration mit dem Pattern_Configurator fertig und kann in den Programm Generator übertragen werden.

Konfiguration zum Programm Generator übertragen

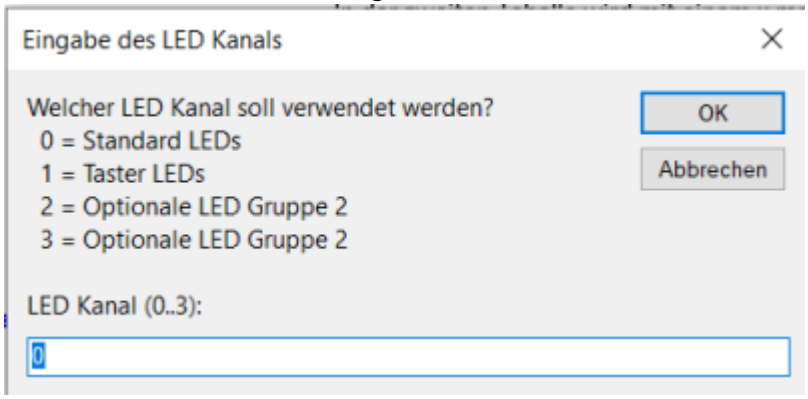
Damit die erstellte Konfiguration jetzt auf den Arduino geladen werden kann, müssen wir diese zunächst in den Programm Generator übertragen. Dies geschieht über die Schaltfläche „Programm Generator“.



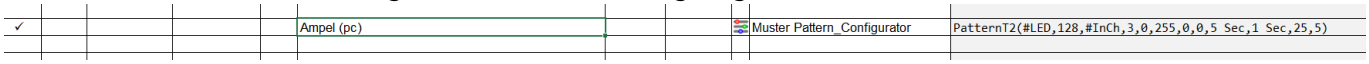
Anschließend öffnet sich der Programm Generator und es erscheint dieses Fenster:



Danach muss im Programm Generator die Zeile ausgewählt werden, in die das Pattern_Configurator Makro kopiert werden soll. Danach erscheint das Fenster „Eingabe des LED Kanals, in dem der verwendete Led – Kanal eingestellt werden kann.



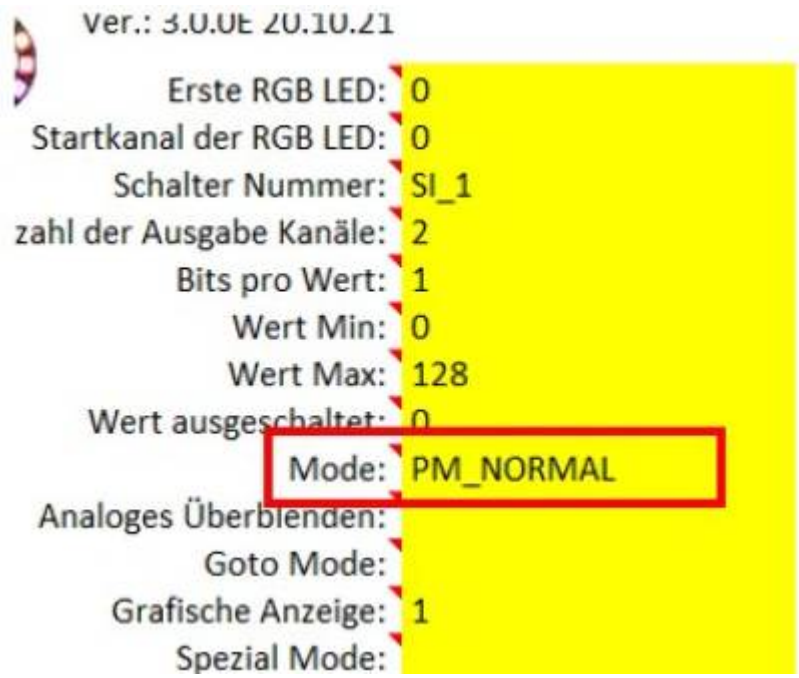
Nun wird die Zeile in den Programm Generator eingefügt und sollte etwa so aussehen:



Anschließend kann die Konfiguration wie gewohnt zum Arduino geschickt werden.

Mode

Mit dem Mode kann der Ablauf des Programms beeinflusst werden. Die Mode Einstellung befindet sich im oberen Bereich des Pattern_Configurators.



Die verschiedenen Modi werden in der folgenden Tabelle beschrieben.

Mode	Beschreibung
PM_SEQUENZ_W_RESTART	Die Pattern Sequenz läuft nur einmal aktiviert durch ein externes Ereignis wie einen Taster durch. Bei einer erneuten Aktivierung während der Laufzeit startet die Sequenz von vorne.
PM_SEQUENZ_W_ABORT	Die Pattern Sequenz läuft nur einmal aktiviert durch ein externes Ereignis wie einen Taster durch. Bei einer erneuten Aktivierung während der Laufzeit wird die Sequenz abgebrochen.
PM_SEQUENZ_NO_RESTART	Die Pattern Sequenz läuft nur einmal aktiviert durch ein externes Ereignis wie einen Taster durch. Bei einer erneuten Aktivierung während der Laufzeit läuft das Programm normal weiter ohne neu zu starten.
PM_PINGPONG	Wenn die Pattern Sequenz durchgelaufen ist, startet sie nicht wieder von vorne, sondern wechselt die Richtung und läuft in umgekehrter Reihenfolge wieder zurück.

Zusätzlich können auch sogenannte Flags eingetragen, die mit den Modes über ein „+“ kombiniert werden können.

Flag	Beschreibung
PF_SLOW	Die Pattern Sequenz läuft 4-mal langsamer als gewöhnlich durch.
PF_INVERT_INP	Invertiert das Eingangssignal
_PF_XFADE	Spezieller Fade-Modus, der vom aktuellen Helligkeitswert statt vom Wert des vorherigen Zustands ausgeht
PF_NO_SWITCH_OFF	Schalten die LEDs nicht aus, wenn der Eingang ausgeschaltet ist. Nützlich, wenn mehrere Effekte die gleichen LEDs verwenden, die durch den Eingangsschalter abwechselnd verwendet werden.
PF_EASEINOUT	Invertieren den Eingangsschalter ⇒ Effekt ist aktiv, wenn der Eingang aus ist

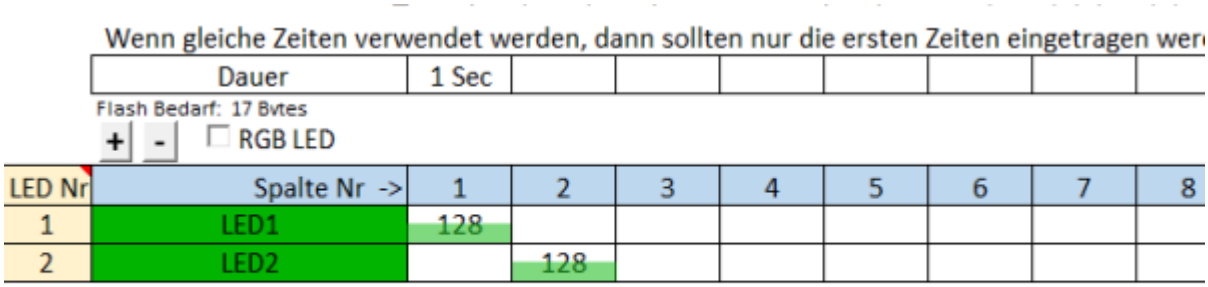
Verschiedene Helligkeitsstufen

Mit dem Pattern_Configurator können Leds nicht nur eingeschaltet und ausgeschaltet, sondern

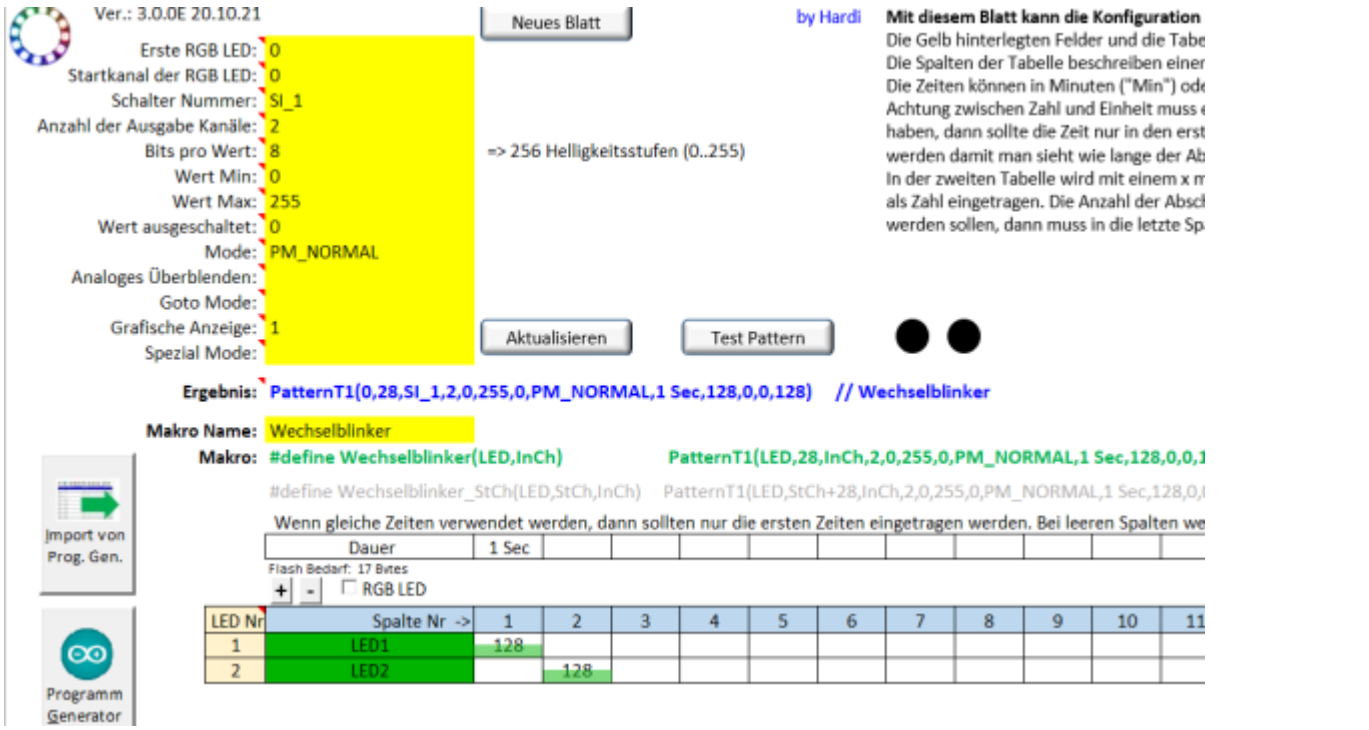
natürlich auch gedimmt werden. Um mehrere Helligkeitsstufen verwenden zu können, muss zunächst der Wert „Bits pro Wert“ verändert werden. Mit diesem kann festgelegt werden, wie viele Helligkeitsstufen verfügbar sind. Dieser Wert sollte immer so gering wie möglich sein, da dadurch sehr viel Speicherplatz gespart werden kann. 1 Bit pro Wert entspricht dabei 2 Helligkeitsstufen (0 und 1). 8 Bits pro Wert entsprechen 256 Helligkeitsstufen (0 bis 255). Die anderen Werte können aus der folgenden Tabelle entnommen werden:

Bits pro Wert	**Anzahl der Helligkeitsstufen**	**Werte in Prozent (%)**
1	2 (0 und 1)	0, 100
2	4 (0 bis 3)	0, 33, 67, 100
3	8 (0 bis 7)	0, 14, 29, 43, 57, 71, 86 100
4	16 (0 bis 15)	0, 7, 13, 20, 27, 33, 40, 47, 53, 60, 67, 73, 80, 87, 93, 100
5	32 (0 bis 31)	in 3,2%-Schritten steigend
6	64 (0 bis 63)	in 1,6%-Schritten steigend
7	128 (0 bis 127)	in 0,8%-Schritten steigend
8	256 (0 bis 255)	in 0,4%-Schritten steigend

In die Tabelle, in der die Konfiguration eingetragen wird, trägt man dann kein „x“ mehr ein, sondern eine Helligkeitsstufe. Wird ein „x“ eingetragen, entspricht das der vollen Helligkeit. In der Tabelle eingetragen sieht das dann so aus:



Wenn die Funktion „grafische Anzeige“ aktiviert ist, kann man die Helligkeit der Leds auch in der Tabelle sehen. Man sieht, dass die Felder für die Leds zur Hälfte grün hinterlegt sind. Das bedeutet, dass die Leds mit halber Helligkeit leuchten. Trägt man zum Beispiel einen Helligkeitswert von 64 ein, wird nur noch das untere Viertel der Felder grün hinterlegt. Dadurch kann man die Helligkeit der Leds sehr gut erkennen. Verwendet man aber zum Beispiel nur 4 Bits pro Wert, stehen 16 Helligkeitsstufen zur Verfügung. In den meisten Fällen sollte das ausreichen. Dann entspricht die volle Helligkeit einem Wert von 15. Dementsprechend können dann auch nur Werte von 0 bis 15 in die Tabelle eingetragen werden. Umso weniger Bits pro Wert verwendet werden, umso geringer wird der Speicherbedarf der Konfiguration. Der aktuelle Speicherbedarf wird im Pattern_Configurator angezeigt.



Optimierung des Speicherbedarfs durch weniger Helligkeitswerte

Wie weit man mit den Helligkeitswerten runter gehen kann um somit Speicherplatz auf dem Arduino zu sparen, zeigt folgendes Beispiel:

Eine RGB-LED soll im deaktivierten Zustand weiß und im aktivierten Zustand rot sein. Für den roten Zustand würde ein „x“ und somit 1 Bit pro Wert reichen (zwei Helligkeitswerte).
Nicht so für die weiße LED. Für diese wurde im Farbtestprogramm beispielsweise ein Farbwert von Rot 255, Grün 192 und Blau 96 ermittelt.

Der größte gemeinsame Teiler aller drei Werte ist 32. Teilt man die 256 Helligkeitswerte durch 32, so kommt man auf 8 Helligkeitswerte. Diese erreicht man bereits bei 3 Bits pro Wert statt bei 8 Bits. In die Tabelle trägt man bei 3 Bits pro Wert daher folgende Werte ein: Rot = X, Grün = 6 (192/32), Blau = 3 (96/32)

Werden im Farbtestprogramm Werte ermittelt, bei denen sich kein entsprechender gemeinsamer Teiler bilden lässt, sollten man prüfen, ob eine minimale Anpassung ein noch akzeptables Ergebnis erzeugt. Wurden beispielsweise die Werte Rot 255, Grün 204 und Blau 92 ermittelt, sollte der oben genannte Farbwert in Betracht gezogen werden. Der Farbunterschied rechtfertigt in den seltensten Fällen 8 Bits pro Wert.

Rundumlicht als Beispiel

Wie stark sich das Reduzieren der „Bits pro Wert“ auf den Speicherbedarf auswirkt, zeigt folgendes Beispiel. Das Muster zeigt zwei Rundumlichter mit jeweils vier LEDs. Im ersten Beispiel wurde das Pattern mit 8 Bits pro Wert programmiert. Die LED wird in ihrer Helligkeit zunächst von 0 auf 64, dann von 64 auf 255 und anschließend entsprechend wieder zurück gedimmt. Stellt man diese Helligkeiten nun in Prozent dar statt in absoluten Werten, entspräche das 0%, 25%, 100%, 25%, 0%. Zur Darstellung des Rundumlichtes werden also streng genommen nur drei der zur Verfügung stehenden 256 Helligkeitswerte benötigt (0%, 25% und 100%).

Mit 3 Bits pro Wert stehen acht Helligkeitswerte zur Verfügung (0%, 14%, 29%, 43%, 57%, 71%, 86% und 100%). Der dritte Helligkeitswert mit knapp 29% kommt den 25% von oben am nächsten. Der Unterschied sollte nicht wahrnehmbar sein, reduziert aber den Speicherbedarf um 50 Bytes.

Wem das noch nicht reicht, kann prüfen, ob man einen Unterschied zwischen 25% und 33% Helligkeit wahrnimmt und kann von 3 auf 2 Bits pro Wert und vier Helligkeitswerte reduzieren. Mit vier Helligkeitsstufen können die vier Werte 0%, 33%, 67% und 100 % dargestellt werden. Damit spart man in Beispiel 3 weitere 10 Bytes.

Die maximale Helligkeit der LEDs sollte **NIE** über die Helligkeitswerte in der Tabelle sondern über **„Wert Max.“** eingestellt werden. Damit reduziert man in der Regel deutlich den Speicherbedarf, da weniger Abstufungen benötigt werden.

Beispiel 1: 8 Bits pro Wert, 256 Helligkeitswerte, **93 Bytes**:

Dauer	32											

Flash Bedarf: 93 Bytes

\+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4	5	6	7	8	9	10	11	12
1	LED1	.	.	.	.	.	64	x	x	64	.		
2	LED2	64	.	.	.	.	.	.	64	x	x		
3	LED3	x	x	64	.	.	.	.	.	.	64		
4	LED1	.	64	x	x	64	.	.	.	.	.		
5	LED2	x	x	64	.	.	.	.	.	.	64		
6	LED3	.	64	x	x	64	.	.	.	.	.		
7	LED1	.	.	.	64	x	x	64	.	.	.		
8	LED2	.	.	.	.	.	64	x	x	64	.		

Beispiel 2: 3 Bits pro Wert, 8 Helligkeitswerte, **43 Bytes**:

Dauer	32											

Flash Bedarf: 43 Bytes

\+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4	5	6	7	8	9	10	11	12
1	LED1	.	.	.	.	.	2	x	x	2	.		
2	LED2	2	.	.	.	.	.	.	2	x	x		
3	LED3	x	x	2	.	.	.	.	.	.	2		
4	LED1	.	2	x	x	2	.	.	.	.	.		
5	LED2	x	x	2	.	.	.	.	.	.	2		
6	LED3	.	2	x	x	2	.	.	.	.	.		
7	LED1	.	.	.	2	x	x	2	.	.	.		
8	LED2	.	.	.	.	.	2	x	x	2	.		

Beispiel 3: 2 Bits pro Wert, 4 Helligkeitswerte, **33 Bytes**:

Dauer	32											

Flash Bedarf: 33 Bytes

\+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4	5	6	7	8	9	10	11	12
1	LED1	.	.	.	.	.	1	x	x	1	.		
2	LED2	1	.	.	.	.	.	.	1	x	x		
3	LED3	x	x	1	.	.	.	.	.	.	1		
4	LED1	.	1	x	x	1	.	.	.	.	.		
5	LED2	x	x	1	.	.	.	.	.	.	1		
6	LED3	.	1	x	x	1	.	.	.	.	.		
7	LED1	.	.	.	1	x	x	1	.	.	.		
8	LED2	.	.	.	.	.	1	x	x	1	.		

Wer wie in diesem Beispiel zwei Fahrzeuge statt einem beleuchtet und von 8 Bits auf 2 Bits pro Wert reduziert, spart 120 Bytes!

Analoges Überblenden

Beim analogen Überblenden stehen zwei Möglichkeiten zur Verfügung.

1. Mit „1“ aktiviert man das Überblenden von einem Farb- bzw. Helligkeitswert zum anderen, ohne dabei den Ist-Zustand des Ausgangswerts zu berücksichtigen. Da diese Funktion keinen Speicher im RAM belegt, sollte diese immer als Erstes in Betracht gezogen werden.
2. Mit „X“ aktiviert man das Überblenden von einem Farb- bzw. Helligkeitswert zum anderen inkl. der Abfrage des Ist-Zustands. Somit ist auch ein Wechsel während eines anderen Wechsels möglich, ohne dass es zu einem Sprung kommt. Diesen Mehrwert bezahlt man aber mit einem zusätzlichen Byte RAM je LED (3 Byte je RGB-LED). Man sollte daher immer prüfen, ob die Abfrage des Ist-Zustands nötig ist.

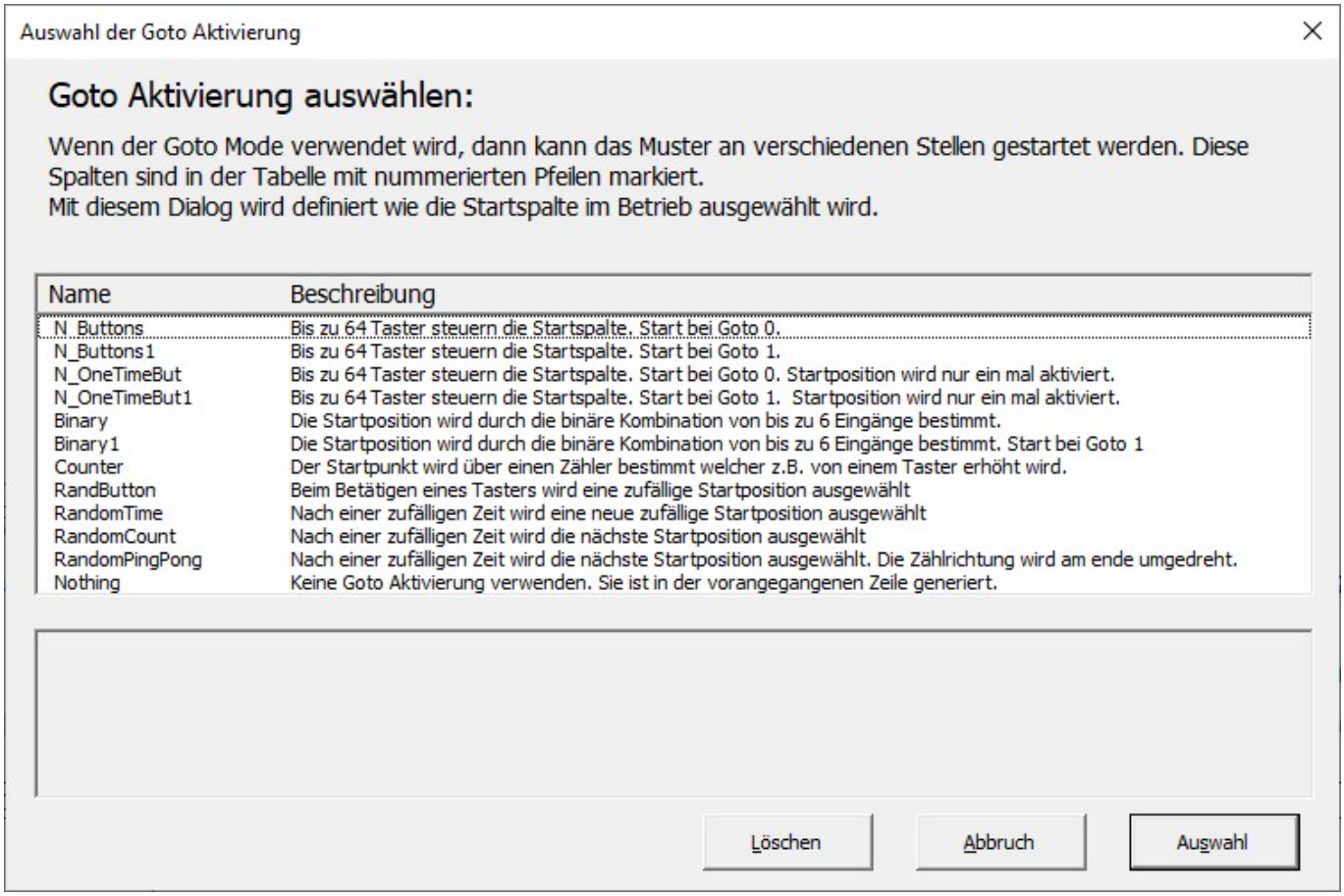
Das analoge Überblenden funktioniert unabhängig von der Anzahl der Helligkeitsstufen. Selbst bei einem Bit pro Wert (2 Helligkeitsstufen) wird sanft von einem zum nächsten Wert überblendet.

Goto Mode

Wenn der Goto Mode verwendet wird, kann ein Muster an verschiedenen Stellen gestartet werden. Diese Start-Spalten sind in der Goto-Tabelle mit nummerierten Pfeilen markiert (bei aktivierter grafischer Anzeige).

Goto Aktivierung

Nach Aktivierung des Goto-Modes über „1“ erscheint folgender Auswahl-Dialog (falls dieser nicht erscheint, erreicht man ihn per Doppelklick im gelben Feld „Goto-Aktivierung):



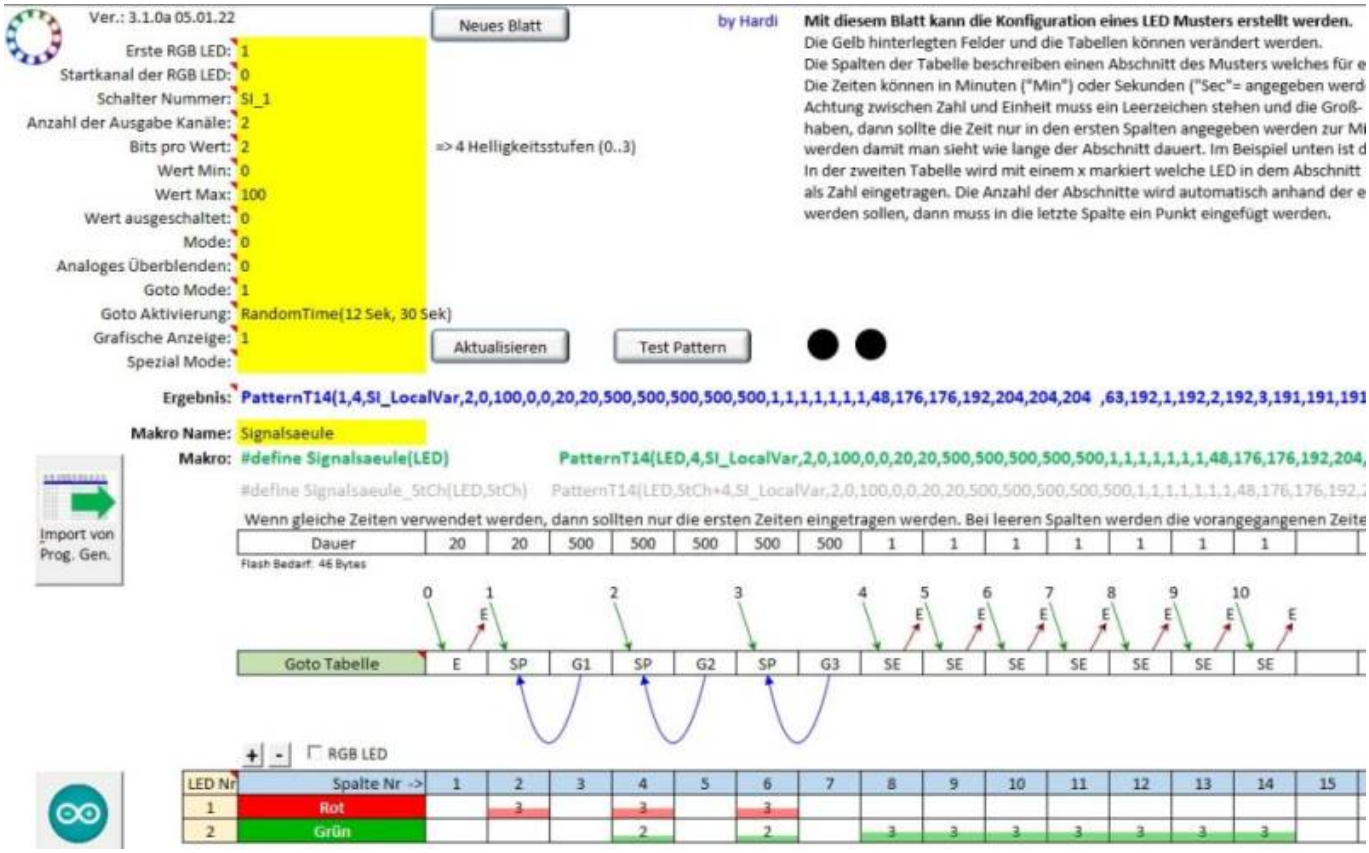
GOTO Aktivierung „N_Buttons“ und „Binary“
Bei der Goto-Aktivierung ist wichtig zu wissen, womit der Effekt später angesteuert werden soll. Soll der Effekt durch einen Tastendruck (Momentschalter) aktiviert werden oder durch einen Schalter (bspw. einer DCC-Adresse an/aus). Für Taster eignet sich die Aktivierung „N_Buttons“, für Schalter die Funktion „Binary“. Die Schalter „Binary“ finden Anwendung bei Selectrix oder im Zusammenhang mit geschalteten DCC-Adressen, wie es beispielsweise die Z21 von Roco/Fleischmann macht. Darüber hinaus stehen noch Taster mit nur einmaliger Abfolge der jeweiligen Sequenz zur Verfügung (N_OneTimeBut).

GOTO Aktivierung „Counter“
Mit der Aktivierung Zählwerk (Counter) lässt bei jedem Tastendruck einer einzelnen Taste einen Goto-Schritt weiter gehen. Um beim Beispiel der Ampel zu bleiben, könnte man somit per Tastendruck zwischen beiden Sequenzen wechseln. Die Goto 0-Sequenz würde mit Gelb starten und auf Rot umspringen. Die Goto 1-Sequenz startet mit Rot/Gelb und endet bei Grün.

GOTO Aktivierung „Random“
Die Zufallsaktivierungen lassen sich wahlweise per Taster (RandButton) **oder** per Zeit zufällig (RandomTime), sequenziell wiederholend (RandomCount) und sequenziell umkehrend (RandomPingPong) aktivieren.

Wahrscheinlichkeit bei Zufallsaktivierungen („RandomTime“ und „RandButton“)
Am Beispiel der Zufallsaktivierungen „RandomTime“ wird im folgenden Beispiel gezeigt, wie man die Wahrscheinlichkeit der zufälligen Aktivierungen beeinflussen kann. Eine Besonderheit dieser Aktivierung ist, dass die Position 0 nur zu Beginn aktiviert wird oder dann, wenn der Eingang auf 0 ist, also die Funktion beispielsweise per DCC deaktiviert ist. Alle anderen Goto-Spalten werden vom Programm zufällig angesprungen, solange der Eingang auf 1 ist, also die Funktion beispielsweise per DCC aktiviert ist.

Soll nun ein Zustand häufiger vorkommen als andere, wird dieser einfach mehrfach angelegt. Am Beispiel einer Signalsäule, wie sie in Fabrikhallen an Automaten benutzt werden, kann man das wie folgt abbilden. Es werden einfach ein Ereignis für Rot (Goto 1), zwei Ereignisse für Gelb (Goto 2 & 3) und sieben Ereignisse für Grün (Goto 4 - 10) angelegt, die zufällig angesteuert werden. Will man Zustand 0 (aus) auch in der zufälligen Reihenfolge haben, legt man diesen einfach als weiteren Zustand oder ein Vielfaches davon an (bspw. Goto 11).



GOTO Aktivierung „Nothing“
Wer die Binärwandlung über "Temporäre 8bit Variable erstellen, binär" selbst machen will, kann die GOTO Aktivierung „Nothing“ wählen. Bin_InCh_toTmpVar speichert den Wert in SI_LocalVar und das Pattern liest den Wert von dort.

Das wohl bekannteste Beispiel für die Verwendung des Goto-Modes sind unsere Signale. Hier werden durch Tasten bzw. durch DCC-Adressen unterschiedliche Zustände mit analogem Überblenden erzeugt. Alle Signale basieren auf dem Pattern_Configurator.

Die Goto-Tabelle

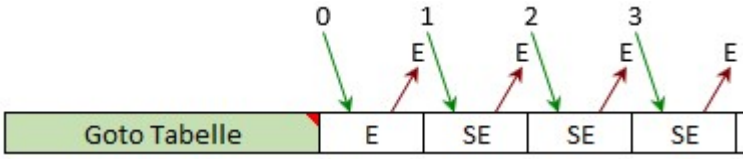
In der Goto-Tabelle können die jeweiligen Positionen der einzelnen Sequenzen mit Start (S), Ende (E), Position (P) und Goto (G) definiert werden. Diese Positionen können innerhalb der Goto-Tabelle auch kombiniert werden. In der nachfolgenden Tabelle sind die entsprechenden Kürzel mit ihrer Funktion aufgeführt.

Befehl	Beschreibung
S	Start einer neuen Sequenz
E	Ende einer Sequenz **ohne** Wiederholung dieser. Der letzte Wert bleibt aktiv.
P	Position, zu der mit „G“ zurückgesprungen werden kann.
G	„Gehe zu“ oder „Goto“-Wert (G1 = Gehe zu Position 1)
SE	Start und Ende einer neuen Sequenz innerhalb **einer** Spalte
SP	Start einer wiederholbaren Sequenz und Position, zu der mit „G“ zurückgesprungen werden kann.
SPE	Start und Ende einer einer wiederholbaren Sequenz innerhalb **einer** Spalte, zu der mit „G“ zurückgesprungen werden kann.
G1	Ende einer Sequenz **inklusive** Wiederholung dieser. Die „1“ bezieht sich auf die Reihenfolge der Startposition.
SG1*	Ende einer Sequenz **inklusive** Wiederholung dieser und gleichzeitigem Start einer nächsten Sequenz mit gleicher Funktion.

Die Funktion SG1 kann beispielsweise verwendet werden, wenn die jeweiligen Startpunkte nicht über Taster, sondern über DCC Schalter angesprungen werden sollen und beim Goto-Mode der Schalter „Binary“ verwendet wird. Die Anzahl der Goto Punkte muss in dem Fall immer eine gerade Anzahl sein (zwei Zustände eines Schalters). Hat man nur drei Sequenzen, kann man mit dieser Funktion die dritte Sequenz auch als vierte nutzen um Speicher im Pattern_Configurator sparen, da die dritte Sequenz sonst zweimal angelegt werden müsste. Ein Beispiel dazu befindet sich in der schaltbaren Ampel mit Blinkfunktion weiter unten. Im Beispiel „Weiß und Farbwechsel mit einer RGB-LED“ wird alternativ der Start an zwei unterschiedlichen Stellen innerhalb der Sequenz aufgezeigt (SP > SP > G1).

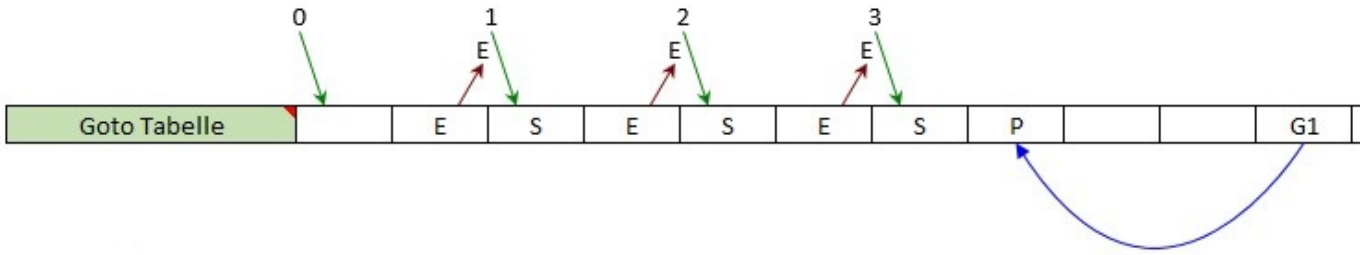
Beispiele für die Goto-Tabelle

Beispiel 1:



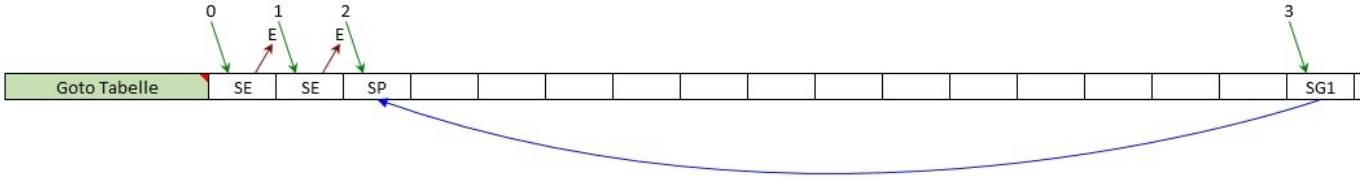
In Beispiel 1 werden über die Goto-Tabelle vier Sequenzen angesprungen, die über Taster (N_Buttons) oder Schalter (Binary) aktiviert werden können. Jede einzelne Sequenz benötigt nur eine Spalte, sodass Start und Ende immer zusammengefasst werden können. Das spart am Ende auch Speicher, da jede zusätzliche Spalte im Pattern_Configurator Speicher auf dem Arduino/ESP32 belegt.

Beispiel 2:



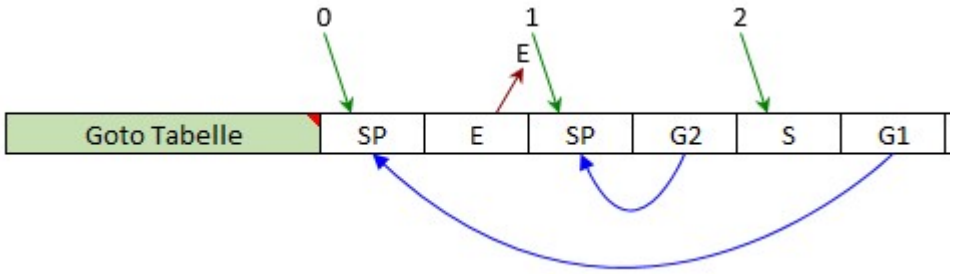
In Beispiel 2 werden über die Goto-Tabelle ebenfalls vier Sequenzen angesprungen, die über Taster (N_Buttons) oder Schalter (Binary) aktiviert werden können. Jede einzelne Sequenz benötigt mehr als eine Spalte, sodass Start und Ende jeweils in einer eigenen Spalte stehen. Eine Besonderheit stellt die dritte Sequenz dar. Hier wird nicht die ganze Sequenz wiederholt sondern nur der Bereich zwischen P und G1. Das kann sinnvoll sein, wenn zwischen S und P eingeblendet wird und das Einblenden nicht wiederholt werden soll oder bei der Ansteuerung von Bewegungen und Sounds.

Beispiel 3:



In Beispiel 3 werden über die Goto-Tabelle nur drei Sequenzen angelegt, die über Taster (N_Buttons) oder Schalter (Binary) aktiviert werden können. Um das Aktivieren über zwei Schalter mit insgesamt vier Zuständen zu ermöglichen, wird die dritte Sequenz durch „SG1“ auch zur vierten Sequenz. Bei aktivierter grafischer Anzeige sieht man den Verlauf anhand der Pfeile:
Einschalten der dritten Sequenz mit dem grünen Pfeil 2 oder dem grünen Pfeil 3 und ab da Wiederholung bis zum Ausschalten über Pfeil 0.

Beispiel 4:



Beispiel 4 zeigt exemplarisch die Zusammenhänge der Positionen und der Goto-Sprünge ohne Anspruch auf Sinnhaftigkeit.

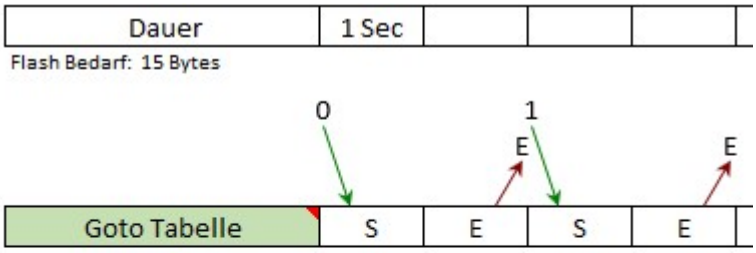
Beispiel einer Goto-Anwendung

Schaltbare Ampel

Bleiben wir beim Beispiel der Ampel. Mit dem Goto-Mode wird es möglich, die Ampelphasen Rot > Grün bzw. Grün > Rot per DCC-Befehl zu steuern.

- Um Bytes zu sparen, wird im Gegensatz zum obigen Beispiel nur noch die Dauer der Gelbphase mit einer Sekunde angegeben.
- Da der letzte Wert jeder Sequenz in den Spalten 2 und 4 (E) bis zum nächsten Tastendruck aktiv bleibt, kommt „Rot“ und „Grün“ jeweils ans Ende.

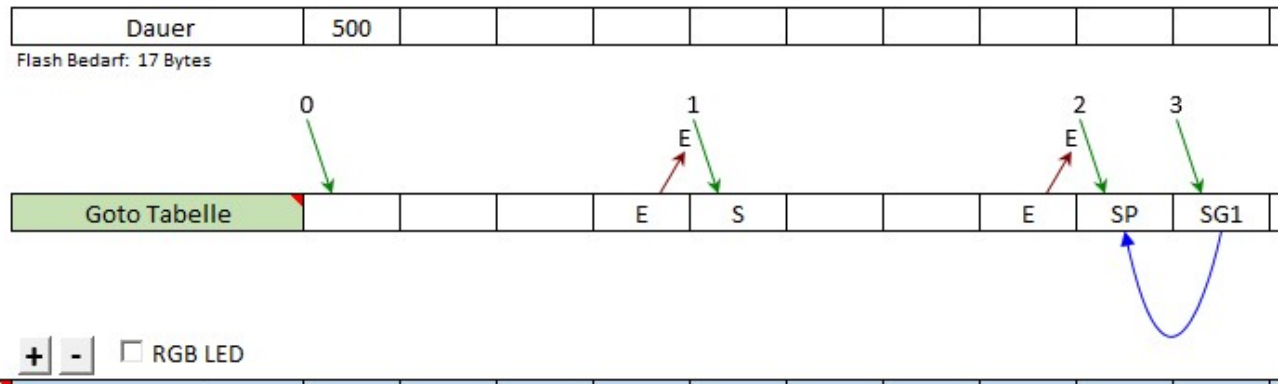
- Den Start beider Sequenzen bilden somit die unterschiedlichen Gelbphasen.
- Mit der Goto-Aktivierung „N_Buttons“ wird mit einem roten Taster der Wechsel von Spalte 4 auf Spalte 1 ausgelöst und mit einem grünen Taster der Wechsel von Spalte 2 auf Spalte 3.
- Mit der Goto-Aktivierung „Binary“ wird mit einer deaktivierten DCC-Adresse der Wechsel von Spalte 4 auf Spalte 1 ausgelöst und mit einer aktivierten der Wechsel von Spalte 2 auf Spalte 3.
- Der Wechsel von Spalte 1 auf 2 bzw. von Spalte 3 auf 4 erfolgt jeweils im Anschluss automatisch nach einer Sekunde, bedingt durch das angelegte Muster.



Schaltbare Ampel mit Blinken

Um der Ampel nun auch noch die Blinkfunktion für die Nacht zu geben, die über eine globale DCC-Adresse für alle Ampeln aktiviert werden kann, müssen wir das Ganze etwas umbauen. Selbstverständlich ließe sich das folgende Beispiel auch mit drei Tasten über „N_Buttons“ lösen. Da aber die Konfiguration mit DCC-Adressen etwas komplizierter ist, nehmen wir **„Binary“** als Beispiel:

- Um Bytes zu sparen, starten wir wieder mit der Gelbphase, reduzieren die Zeiteinheit jedoch auf eine halbe Sekunde.
- Die halbe Sekunde benötigen wir für die Blinkphase.
- Da, wo wir 1,0 bzw. 1,5 Sekunden benötigen, wiederholen wir einfach die Spalten mit den Gelbphasen.
- Diese Vorgehensweise benötigt weniger Speicher als unterschiedliche Zeiten in sechs Spalten.
- Goto 0 startet mit Gelb und wechselt nach 1500 ms. zu Rot, wo es verharrt.
- Goto 1 startet mit Rot/Gelb und wechselt nach 1500 ms. zu Grün, wo es verharrt.
- Goto 2 und Goto 3 starten den Blinkmodus, unabhängig vom vorherigen Zustand (Rot bzw. Grün) und wiederholen diesen bis zur Deaktivierung.
- Achtung: Bitte auf den Punkt in Spalte 10 achten.



LED Nr	Spalte Nr ->	1	2	3	4	5	6	7	8	9	10
1	rot				x	x	x	x			
2	gelb	x	x	x		x	x	x		x	.
3	grün								x		

Nun werden im Programm Generator mit der Logik-Funktion zwei aufeinanderfolgende Variablen definiert.

- „DCC-Adresse n“ wird mit der 1. Variable „Bsp_Ampel_n_1“ verknüpft.
- „DCC Adresse a“ wird mit der 2. Variable „Bsp_Ampel_n_2“ verknüpft.
- „n“ ist die DCC Adresse der zu schaltenden Ampel, aber nur die Rot-Grün und Grün-Rot-Phase.
- „a“ ist die globale DCC-Adresse, die für alle Ampeln einer Kreuzung verwendet wird, um diese blinken zu lassen.
- „a“ muss Dank der Logik-Verknüpfung nicht aufeinanderfolgend zu „n“ sein.

DCC n	DCC a	Bsp_Ampel_n_1	Bsp_Ampel_n_2	Goto-Wert	Zustand der Ampel
aus	aus	0	0	0	Gelb für 1,5 Sek. mit Wechsel auf Rot
ein	aus	1	0	1	Rot/Gelb für 1,5 Sek. mit Wechsel auf Grün
aus	ein	0	1	2	Umschalten von Rot auf gelbes Blinken
ein	ein	1	1	3	Umschalten von Grün auf gelbes Blinken

Im Programm Generator sieht das Ganze dann so aus:

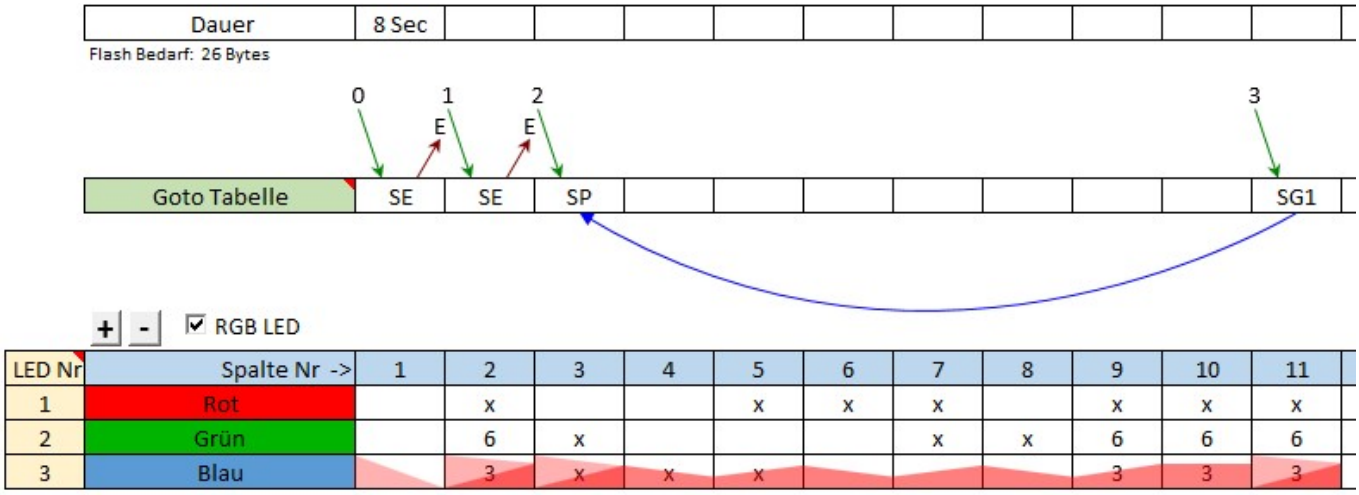
Aktiv	Filter	Adresse oder Name	Typ	Start wert	Beschreibung	Verteiler-Nummer	Stecker-Nummer	Icon	Name	Beleuchtung, Sound, oder andere Effekte	Start LedNr	LEDs	InCnt	Loc InCh	LED/ Sound Kanal
✓		47	AnAus 0		Adresse 47 schaltet Goto 0 oder Goto 1				Logische Verknüpfung	Logic(Bsp_Ampel_n_1, #InCh)			1	0	
✓		11	AnAus 0		Adresse 11 schaltet Goto 2 oder Goto 3				Logische Verknüpfung	Logic(Bsp_Ampel_n_2, #InCh)			1	0	
✓		Bsp_Ampel_n_1			Ampel_DCC (pc)				Muster Pattern_Configu	// Activation: BinaryBin_InCh_to_	1	1	2	0	0

Zum besseren Verständnis sei zu erwähnen, dass sich der Zustand der LED durch einen Tastendruck verändert. Bei der Ampel werden die drei an einen WS2811 angeschlossenen LEDs in ihren jeweiligen Zuständen verändert. Auf gleichem Weg lässt sich selbstverständlich auch die Farbe einer WS2812-LED durch Tastendruck ändern (beispielsweise von weiß auf Farbwechsel).

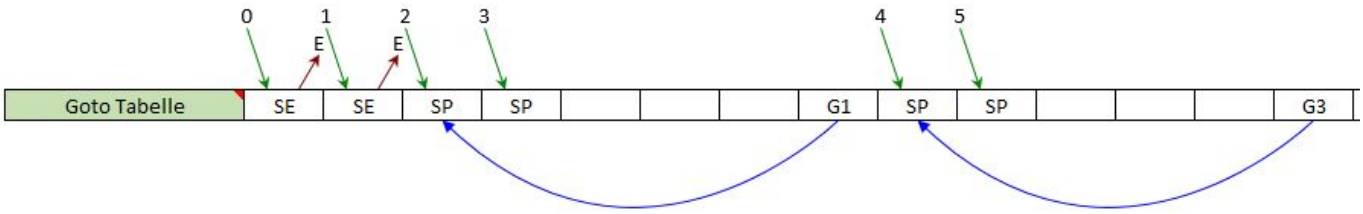
Weiß und Farbwechsel mir einer RGB-LED:
Wer beispielsweise eine Burg, ein Stadttor oder ein Viadukt mit einem RGB Farbwechsel beleuchten möchte, dies aber nur als Knopfdruckaktion als Alternative zur weißen Beleuchtung ausführen möchte, kann beide Abläufe im Pattern_Configurator vereinen.

Im folgenden Beispiel sind die drei Sequenzen zu sehen:

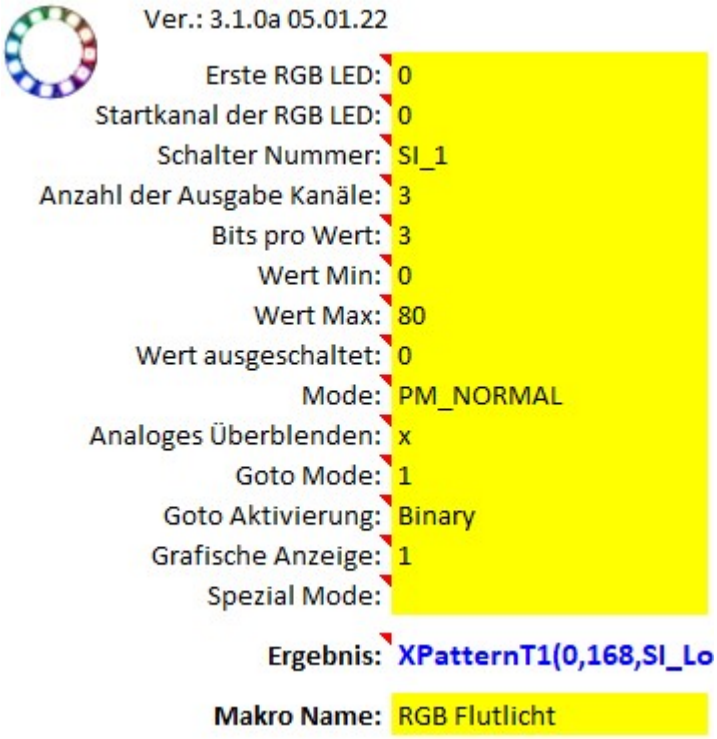
- Goto 0: Alle LEDs sind deaktiviert.
- Goto 1: Die Werte x, 6 und 3 ergeben bei 3 Bits pro Wert ein kaltweißes Licht (siehe oben).
- Goto 2: RGB-Farbwechsel (je 8 Sekunden Cyan, Blau, Magenta, Rot, Gelb, Grün, 3x Weiß).
- Goto 3: Macht dasselbe wie Goto 2 und wird nur benötigt, wenn die Aktivierung über „Binary“ gewählt wurde (siehe Tipp bei den Goto-Sprüngen).



Alternativ kann der zweite Startpunkt auch innerhalb einer Sequenz gesetzt werden.
Im folgenden Beispiel sind bei Goto 0 alle LEDs aus, bei Goto 1 leuchten sie weiß, Goto 2 und drei aktivieren den Farbwechsel und Goto 4 und 5 ein wildes Blinkmuster.



Die Konfiguration zum Schalten per DCC-Adresse sieht in dem Fall so aus:



Aktiviert wird das Ganze über zwei frei wählbare DCC-Adressen mit Logik-Verknüpfung.

Schalten mit der Aktivierung „Binary“:

Beim Schalten über DCC-Adressen ist folgendes zu beachten. Die DCC-Adressen müssen aufeinander folgend sein, andernfalls ist mit aufeinander folgenden Variablen zu schalten, die als logische Verknüpfung definiert werden müssen (siehe Beispiel „Weiß und Farbwechsel“). Bleiben wir also bei aufeinander folgenden DCC-Adressen

DCC n	DCC n+1	Goto-Wert
aus	aus	0
ein	aus	1
aus	ein	2
ein	ein	3

Grafische Anzeige

Die grafische Anzeige wird über eine „1“ im gelben Feld aktiviert und erleichtert es, die Abläufe im Pattern_Configurator zu verstehen.

Beispiel 1:
In Beispiel 1 ist die oben gezeigte DCC-Ampel ohne aktivierte grafische Anzeige abgebildet.

Dauer	1 Sec			

Flash Bedarf: 15 Bytes

Goto Tabelle	S	E	S	E

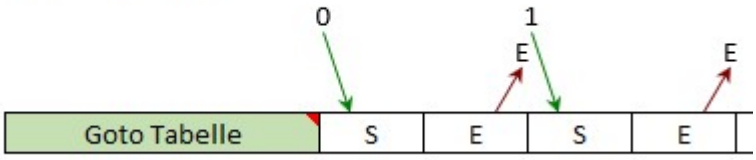
+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4
1	rot		x	x	
2	gelb	x		x	
3	grün				x

Beispiel 2:
In Beispiel 2 ist die oben gezeigte DCC-Ampel mit aktivierter grafische Anzeige abgebildet.

Dauer	1 Sec			

Flash Bedarf: 15 Bytes

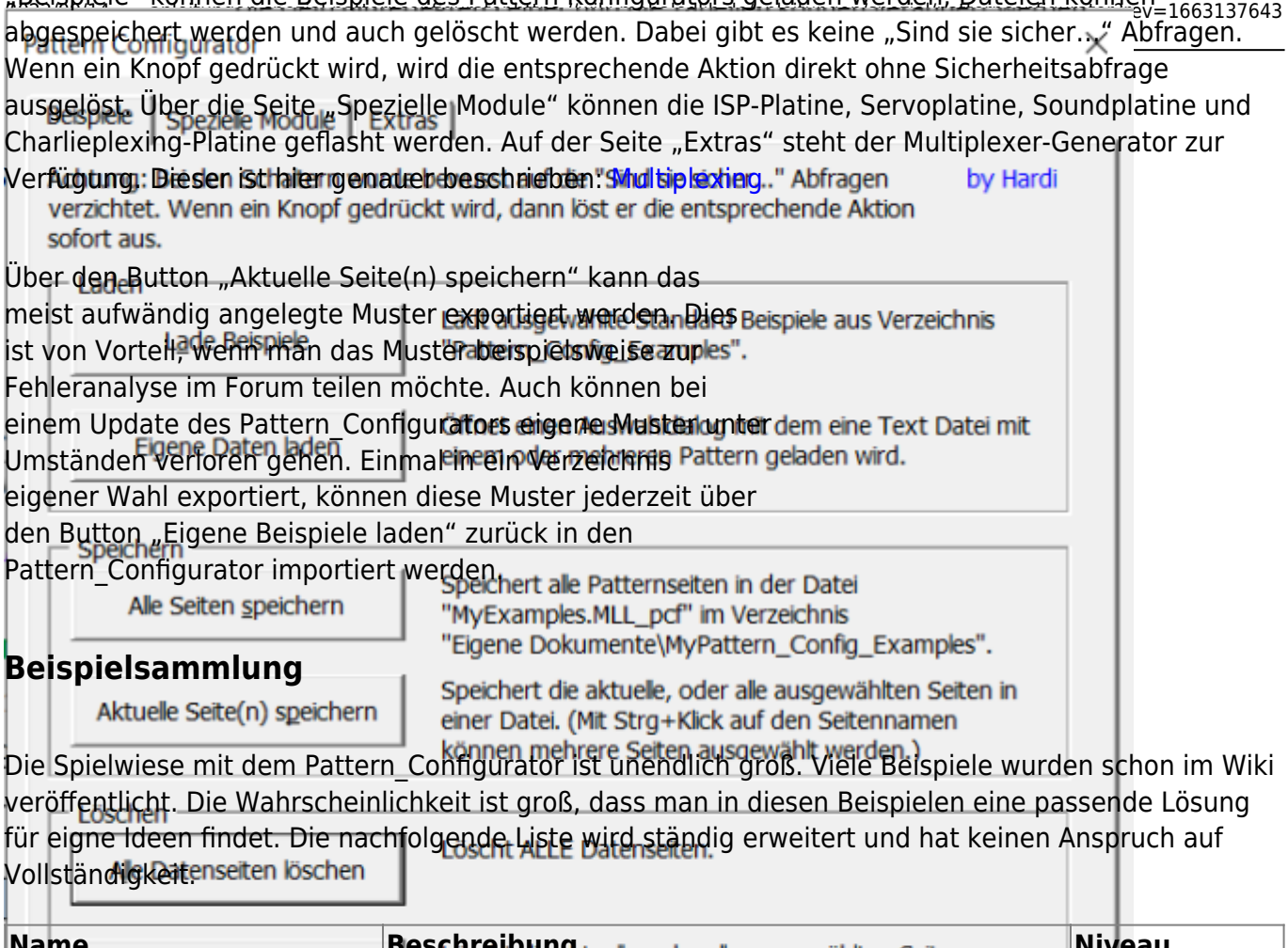


+ - ☐ RGB LED

LED Nr	Spalte Nr ->	1	2	3	4
1	rot		x	x	
2	gelb	x		x	
3	grün				x

Beispiel 3:
In Beispiel 3 ist die oben gezeigte DCC-Ampel mit aktivierter grafische Anzeige und analogem Überblenden abgebildet.

Dieser ist in 3 Seiten unterteilt, die am oberen Rand ausgewählt werden können. Über die Seite „Beispiele“ können die Beispiele des Pattern Konfigurators geladen werden, Dateien können abgespeichert werden und auch gelöscht werden. Dabei gibt es keine „Sind sie sicher…“ Abfragen. Wenn ein Knopf gedrückt wird, wird die entsprechende Aktion direkt ohne Sicherheitsabfrage ausgelöst. Über die Seite „Spezielle Module“ können die ISP-Platine, Servoplatine, Soundplatine und Charlieplexing-Platine geflasht werden. Auf der Seite „Extras“ steht der Multiplexer-Generator zur Verfügung. Dieser ist hier genauer beschrieben: Multiplexing.



Über den Button „Aktuelle Seite(n) speichern“ kann das meist aufwändig angelegte Muster exportiert werden. Dies ist von Vorteil, wenn man das Muster beispielsweise zur Fehleranalyse im Forum teilen möchte. Auch können bei einem Update des Pattern_Configurators eigene Muster unter Umständen verloren gehen. Einmal in ein Verzeichnis eigener Wahl exportiert, können diese Muster jederzeit über den Button „Eigene Beispiele laden“ zurück in den Pattern_Configurator importiert werden.

Beispielsammlung

Die Spielwiese mit dem Pattern_Configurator ist unendlich groß. Viele Beispiele wurden schon im Wiki veröffentlicht. Die Wahrscheinlichkeit ist groß, dass man in diesen Beispielen eine passende Lösung für eigne Ideen findet. Die nachfolgende Liste wird ständig erweitert und hat keinen Anspruch auf Vollständigkeit.

Name	Beschreibung	Niveau
Rundumlicht & Sicherungsanhänger	Rundumlicht mit vier LEDs, die sich nahtlos drehen und Sicherungsanhänger mit typischem Blinkverhalten.	Anfänger
Schweißlicht mit Sound	Gleichzeitiges Abspielen einer 3-sekündigen Sounddatei mit passendem Schweißlicht-Flackern.	Anfänger
Formsignalen mit Ministeppern	Stepper ein- und ausschalten sowie gleichzeitig die Drehrichtung beeinflussen.	Fortgeschrittene
Holzfäller	Holzfäller bewegt sich, Baum fällt, alles mit synchronisiertem Sound.	Fortgeschrittene
Lagerfeuer und Holzhacker	Wenn das Feuer langsam ausgeht, hackt der Holzhacker frisches Holz. Nur im Stammtisch-Video.	Fortgeschrittene
Laubbläser	Gleichzeitige Steuerung von Servo-Bewegung und Soundmodul.	Fortgeschrittene
Ungewollt belebtes Haus	Schrittschaltwerk steuert kompletten Ablauf einer nachgestellten Einbruch-Szene.	Experten

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

Permanent link:
https://wiki.mobaledlib.de/anleitungen/spezial/patternconfigurator?rev=1663137643

Last update: **2022/09/14 07:40**