# HSV-Farben im Pattern Configurator

Motiviert durch den komplizierten Farbwechsel meiner Burg wollte ich schon immer verstehen, wie der viel einfachere Farbwechsel mit HSV-Farben umzusetzen ist. Dank Norberts Hilfe beim Oktober-Stammtisch 2023 konnte ich das Rätsel nun endlich knacken und lasse euch wie im Forum versprochen daran teilhaben.
— *Michael 2023/11/07*

## Was sind die HSV-Farben?

Der HSV-Farbraum beschreibt den Ort einer Farbe innerhalb eines Kegels. Zum besseren Grundverständnis eignet sich der entsprechende Artikel auf Wikipedia.

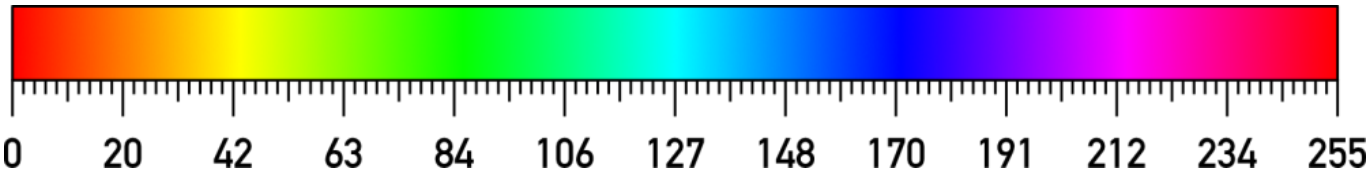
- **H**ue (Farbwert): Farbwinkel innerhalb des Farbkreises. Rot (0°), Grün (120°), Blau (240°) und am Ende wieder Rot (360°)
- **S**aturation (Sättigung): Neutralgrau (0%), wenig gesättigte Farbe (50%), gesättigte bzw. reine Farbe (100%)
- **V**alue (Helligkeit): keine Helligkeit (0%), volle Helligkeit (100%)

Da bei der MobaLedLib keine Werte in Prozent eingeben werden können, wird dieses Farbsystem mit den 8bit (0-255) umgesetzt, die die FastLED Bibliothek* zur Verfügung stellt. Daraus ergeben sich dann folgende Werte:

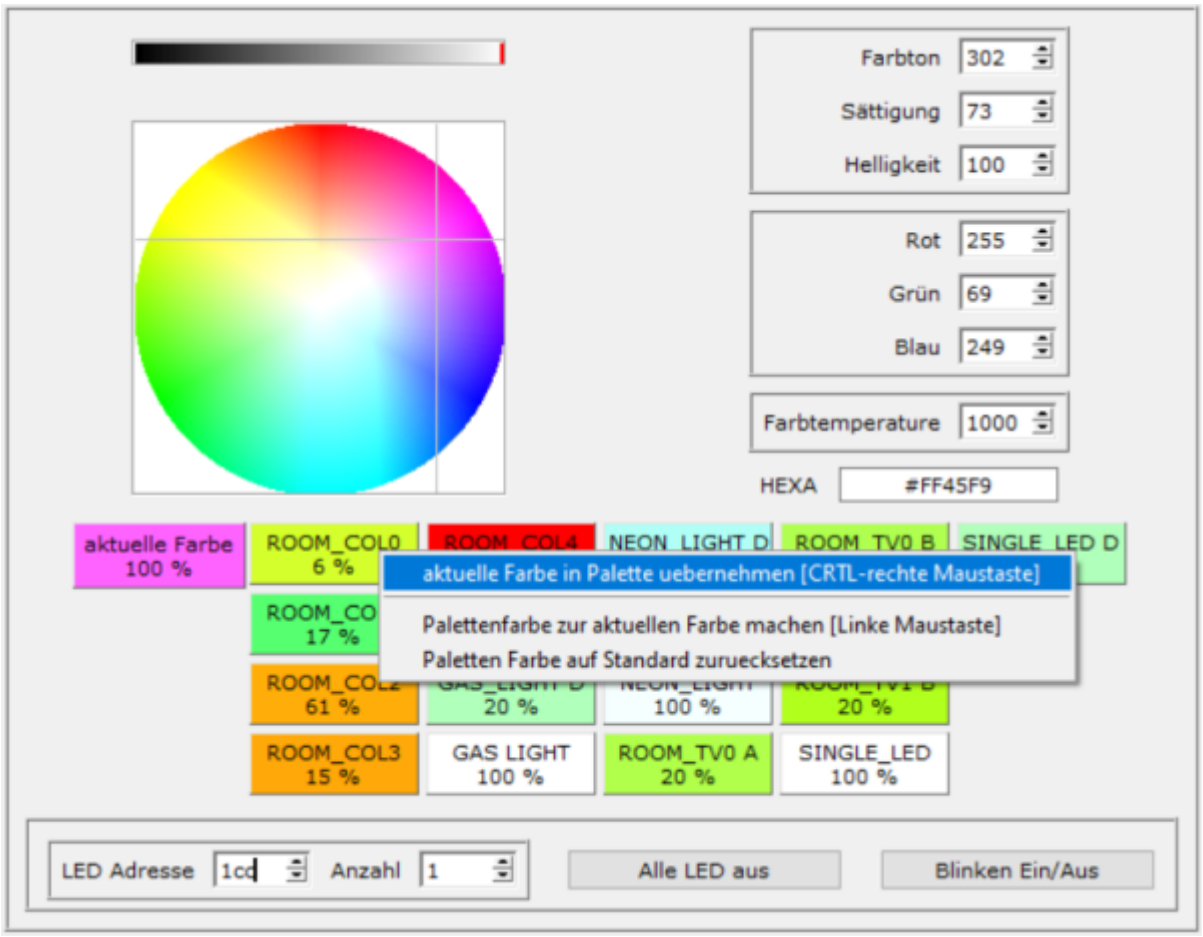
- **H**ue (Farbwert): Farbwinkel innerhalb des Farbkreises. Rot (0), Grün (74), Blau (170) und am Ende wieder Rot (255)
- **S**aturation (Sättigung): Neutralgrau (0), wenig gesättigte Farbe (127), gesättigte bzw. reine Farbe (255)
- **V**alue (Helligkeit): keine Helligkeit (0), volle Helligkeit (255)

*) Link zur FastLED-Beschreibung: FastLED-HSV-Colors

Der wichtigste Wert ist selbstverständlich der Farbwert selbst (Hue), denn Helligkeit und Sättigung beeinflussen jeden Farbwert in gleichem Maße.
Der folgende Verlauf zeigt, mit welchem Wert eine bestimmte Farbe zu erzielen ist. Für alle im Verlauf dargestellten Farben sind sowohl Sättigung als auch Helligkeit auf 255 eingestellt.




Mithilfe der HSV-Werte lassen sich alle Farben für eine RGB-LED sehr schnell definieren. Eine gute Hilfe bietet dabei das Farbtestprogramm von Harold.
Hier kann man Farbe und Sättigung durch Verschieben des Fadenkreuzes und die Helligkeit mithilfe des Reglers über dem Farbkreis einstellen. Die drei Werte kann der Pattern Configurator im Mode PM_HSV dann direkt interpretieren.



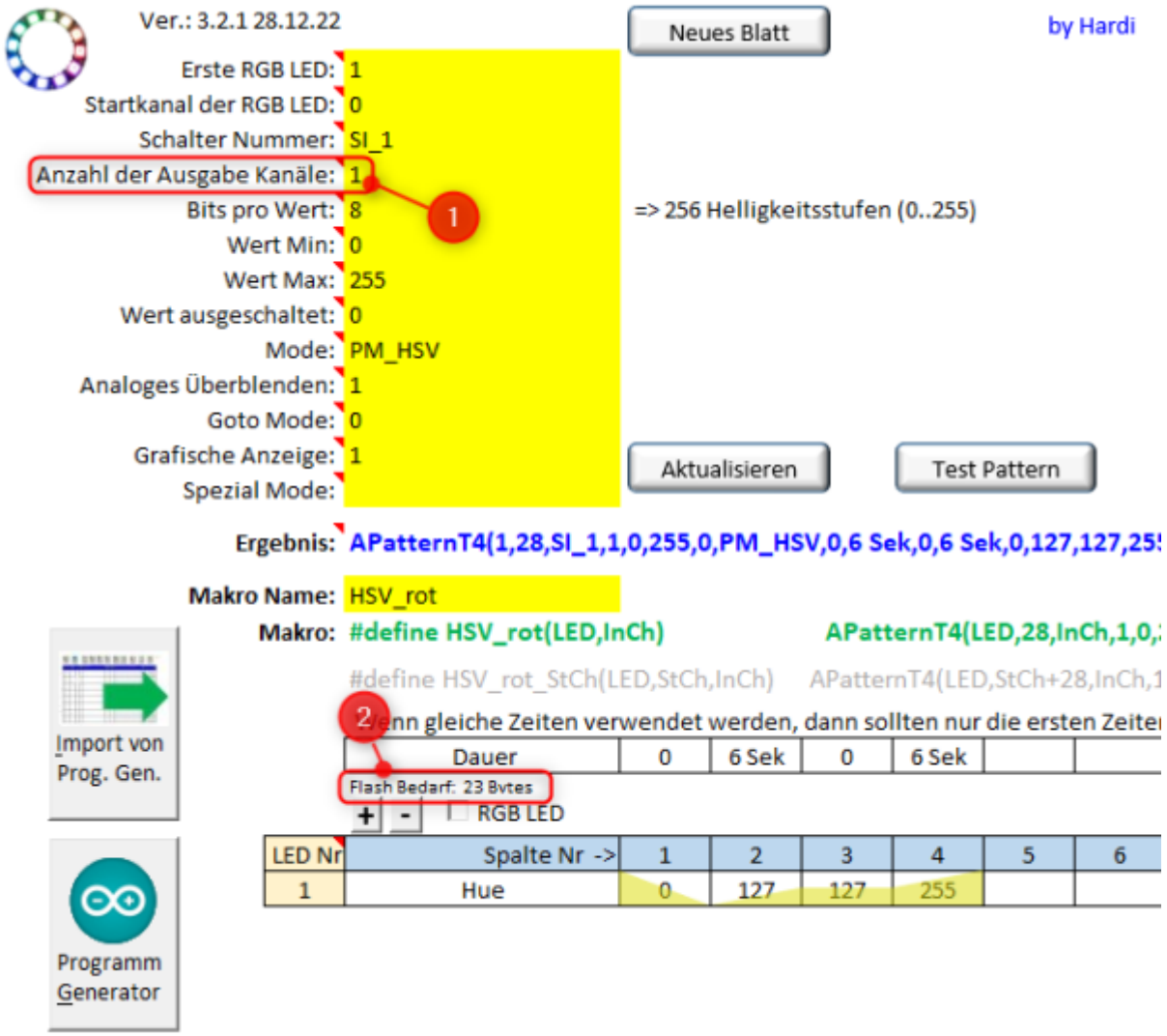

# Wozu braucht man das?

Gerade Farbverläufe, wie sie heute vielerorts zur Effektbeleuchtung eingesetzt werden, belegen mit RGB Farben etwas mehr Speicher auf dem Arduino. Zudem ist das ständige Hantieren mit den ganzen RGB-Werten (jeweils drei pro Farbe) viel umständlicher als mit einem Hue-Wert.

Ein ganz besonderer Vorteil ist es beispielsweise, sechs Einzel-LEDs im Pattern Configurator anzulegen und diese in sechs Zeilen des Programm Generators zu kopieren. Will man beispielsweise 18 LEDs damit ansteuern, kann man die jeweiligen Farben mit dem „Copy LED“-Befehl verfielfachen.

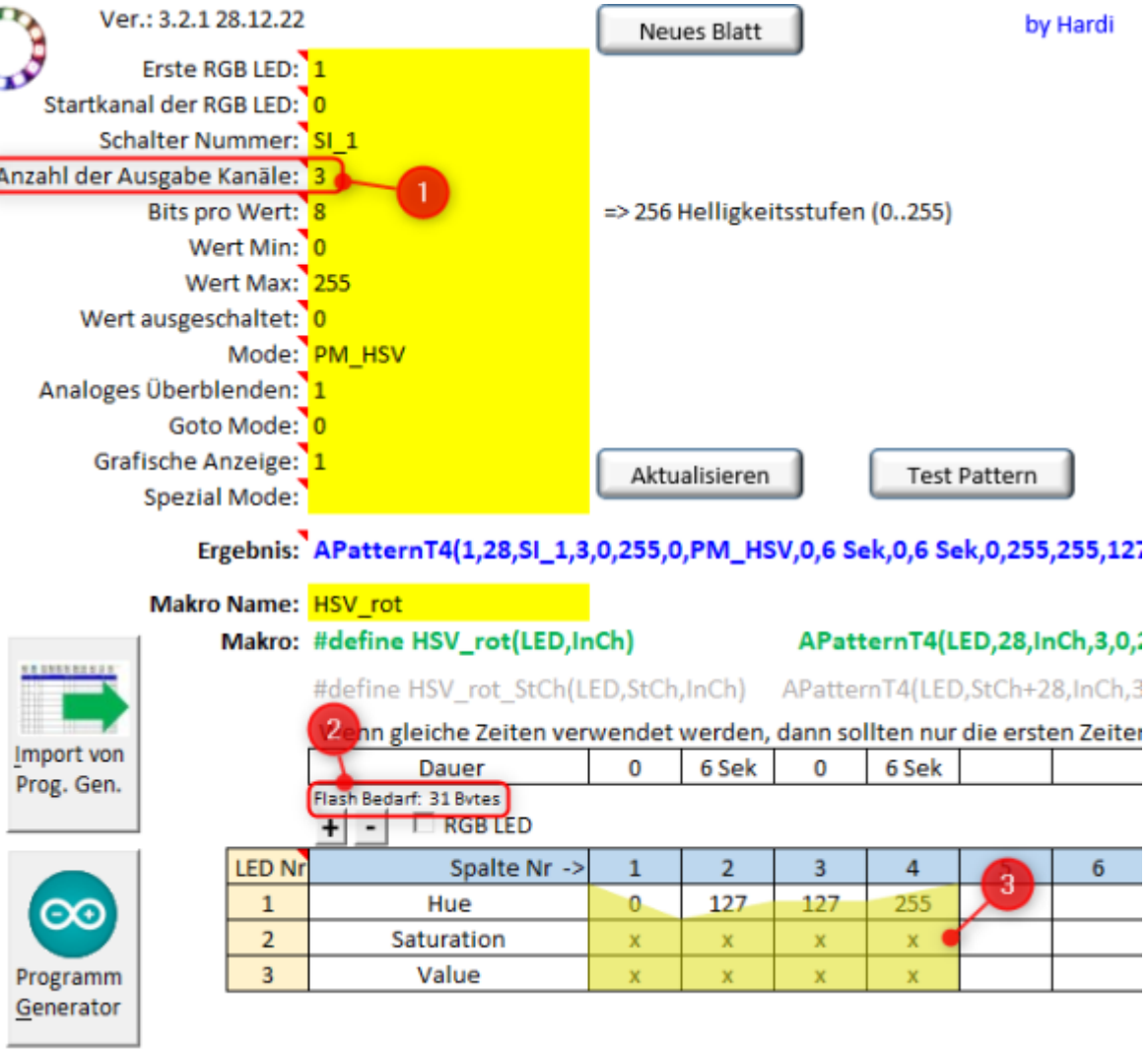
# Wie programmiert man das?

**Nur Farbwert einstellen**
Wenn man nur die Farbwerte programmieren möchte (Hue), kann man im Pattern Configurator etwas Speicher sparen (2), indem man nur einen Ausgabekanal angibt (1). In dem Fall steuert man mit dem ersten Ausgabekanal nicht wie sonst üblich den roten Chip auf einer WS2812, sondern den Farbwert aller drei Chips. Die fehlenden Ausgangskanäle Saturation und Value werden bei nur einem Ausgabekanal automatisch auf 255 (100%) gesetzt. Konkret sieht das so aus:




**Alle Werte einstellen**

Will man den Farbwert (Hue) und die Sättigung (Saturation) einstellen, wählt man zwei Ausgabekanäle, will man alles einstellen, wählt man alle drei Ausgabekanäle (1). Eine Besonderheit stellt die Steuerung von Farbwert (Hue) und Helligkeit (Value) dar. Wer nur diese beiden Werte einstellen will, muss alle drei Ausgabekanäle auswählen, weil die Sättigung (Saturation) immer an Stelle zwei steht. In dem Fall muss bei der Sättigung auch ein Wert eingetragen werden (3), denn wenn die Zeile leer bleibt, geht das Programm vom Wert 0 aus. Die LED wäre also aus.




**Vor- und Nachteile beider Methoden**
Der Vorteil von Variante 1 liegt in der Einfachheit und beim Speicherbedarf der Programmierung. Schwierig wird es, wenn man diese Variante beispielsweise mit einer Goto-Tabelle ein- und ausschalten will.
Da die Ausgabekanäle 2 und 3 nicht programmiert sind, stehen diese immer auf 255, unabhängig vom Schaltzustand der Goto-Tabelle. Man wird die RGB-LED mit Variante 1 also nie ausschalten können. Entweder sie macht den definierten Farbton bzw. Farbwechsel (an) oder sie leuchtet rot (aus). Das liegt daran, dass der Hue-Wert „0“ in der oben gezeigten Skala der Farbe Rot entspricht. Will man erreichen, dass die LED nach einer gewissen Zeit einfach ganz ausgeht, benötigt man die zusätzliche Zeile „Helligkeit“, die man dann entsprechend auf „0“ setzt. Es empfiehlt sich daher in den meisten Fällen, mit Variante 2 zu arbeiten.

# Ablauf des Farbwechsels

Der automatische Farbwechsel startet beim eingegebenen Hue-Wert, läuft dann in der vorgegebenen Zeit zum nächsten Hue-Wert. Hat er diesen erreicht, wechselt die LED in umgekehrter Reihenfolge ihre Farbe zurück zum Startwert. Der Farbwechsel pendelt. Dieser Effekt eignet sich beispielsweise für eine gleichmäßige Beleuchtung eines Objekts, bei der man den Farben nicht „hinterher“ schaut.
Will man einen Farbwechsel erzeugen, bei dem alle Farben nacheinander an einem Objekt „vorbeiziehen“, eignet sich diese Methode nicht, da die Farben auf dem Objekt hin und her wandern würden. In dem Fall bedient man sich eines einfachen Tricks. Nach erfolgtem Farbwechsel von

beispielsweise 0 nach 255 in zehn Sekunden lässt man die Farbe von 255 nach 0 in null Sekunden wechseln. Der Farbwechsel rotiert.
Da in dem Fall die Farben 0 (rot) und 255 (rot) annähernd identisch sind, kann das Auge diesen Sprung nicht erfassen. Dasselbe gilt selbstverständlich für einen Wechsel von 43 (gelb) über 255 (rot) nach 42 (gelb). Damit der Verlauf im Anschluss nicht über 255 zurück nach 43 wandert, definiert man diesen Vorgang mit null Sekunden.

**Beispiel mit identischer Sättigung und Helligkeit:**
Im Folgenden sind die sechs Grundfarben Rot (0), Gelb (42), Grün (84), Cyan (127), Blau (169) und Magenta (211) als Farbwechsel mit jeweils zwei Sekunden pro Farbton und einer Wechseldauer von 12 Sekunden je Durchgang dargestellt.

Das hier gezeigte Prinzip stellt alle Farben mit gleicher Sättigung und mit gleicher Helligkeit über den gesamten Farbwechsel dar. Will man die verschiedenen Farben in ihrer Helligkeit beeinflussen, muss man sie im Pattern Configurator vereinzeln. Ein Beispiel dazu folgt weiter unten.

**Rotierender Farbwechsel beginnend mit Rot (255):**

| | Dauer | 12 Sek | 0 |
|---|---|---|---|

Flash Bedarf: 21 Bytes
\+ - ☐ RGB LED

| LED Nr | Spalte Nr -> | 1 | 2 |
|---|---|---|---|
| 1 | Hue | 255 | 0 |
| 2 | Saturation | x | x |
| 3 | Value | 127 | 127 |

**Rotierender Farbwechsel beginnend mit Gelb (42):**

| | Dauer | 10 Sek | 0 | 2 Sek |
|---|---|---|---|---|

Flash Bedarf: 26 Bytes
\+ - ☐ RGB LED

| LED Nr | Spalte Nr -> | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | Hue | 255 | 0 | 42 |
| 2 | Saturation | x | x | x |
| 3 | Value | 127 | 127 | 127 |

**Rotierender Farbwechsel beginnend mit Grün (84):**

| | Dauer | 8 Sek | 0 | 4 Sek |
|---|---|---|---|---|

Flash Bedarf: 26 Bytes
\+ - ☐ RGB LED

| LED Nr | Spalte Nr -> | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | Hue | 255 | 0 | 84 |
| 2 | Saturation | x | x | x |
| 3 | Value | 127 | 127 | 127 |

**Rotierender Farbwechsel beginnend mit Cyan (127):**

| Dauer | 6 Sek | 0 | 6 Sek |
|---|---|---|---|

Flash Bedarf: 26 Bytes

\+ - ☐ RGB LED

| LED Nr | Spalte Nr -> | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | Hue | 255 | 0 | 127 |
| 2 | Saturation | x | x | x |
| 3 | Value | 127 | 127 | 127 |

### Rotierender Farbwechsel beginnend mit Blau (169):

| Dauer | 4 Sek | 0 | 8 Sek |
|---|---|---|---|

Flash Bedarf: 26 Bytes

\+ - ☐ RGB LED

| LED Nr | Spalte Nr -> | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | Hue | 255 | 0 | 169 |
| 2 | Saturation | x | x | x |
| 3 | Value | 127 | 127 | 127 |

### Rotierender Farbwechsel beginnend mit Magenta (211):

| Dauer | 2 Sek | 0 | 10 Sek |
|---|---|---|---|

Flash Bedarf: 26 Bytes

\+ - ☐ RGB LED

| LED Nr | Spalte Nr -> | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | Hue | 255 | 0 | 211 |
| 2 | Saturation | x | x | x |
| 3 | Value | 127 | 127 | 127 |

**Beispiel mit unterschiedlicher Helligkeit:**
Im folgenden Beispiel sind die Hue-Werte in 42er Schritte aufgeteilt, um jedem Farbwert eine andere Helligkeit zuzuordnen. Selbstverständlich ließen sich in dem Beispiel einige Spalten wieder zusammenlegen. Das Beispiel soll aber verdeutlichen, wie man parallel zum Farbverlauf einen angepassten Helligkeitsverlauf anlegen kann. So kann man beispielsweise einen Ausgleich zwischen intensivem Rot und schwachem Blau schaffen.
Mit diesem Verfahren ließe sich auch die Heartbeat-LED erstellen. Dazu müsste lediglich die Helligkeit pulsieren, während der Farbwert wechselt.

| Dauer | 2 Sek | 0 | 2 Sek | 2 Sek | 2 Sek | 2 Sek | 2 Sek |
|---|---|---|---|---|---|---|---|

Flash Bedarf: 46 Bytes

\+ - ☐ RGB LED

| LED Nr | Spalte Nr -> | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | Hue | 255 | 0 | 42 | 84 | 127 | 170 | 212 |
| 2 | Saturation | x | x | x | x | x | x | x |
| 3 | Value | 170 | 170 | 150 | 170 | 212 | 255 | 212 |

**Richtung des Farbwechsels:**
Im Programm Generator werden nun die sechs Pattern untereinander mit derselben Adresse angesteuert. Beginnt man mit Rot, so läuft der Farbwechsel von rechts nach links.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | 1 | AnAus | 1 | | HSV_rot (pc) | | | HSV | Speicher für HSV reservieren | New_HSV_Group() | | |
| ✓ | | 1 | AnAus | 1 | | HSV_gelb (pc) | | | | Muster Pattern_Configurator | APatternT2(#LED,28,#InCh,3,0,255,0,PM_HSV,12 Sek,6 | 1 | 1 |
| ✓ | | 1 | AnAus | 1 | | HSV_gruen (pc) | | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,10 Sek,6 | 2 | 1 |
| ✓ | | 1 | AnAus | 1 | | HSV_cyan (pc) | | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,8 Sek,0, | 3 | 1 |
| ✓ | | 1 | AnAus | 1 | | HSV_blau (pc) | | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,6 Sek,0, | 4 | 1 |
| ✓ | | 1 | AnAus | 1 | | HSV_magenta (pc) | | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,4 Sek,0, | 5 | 1 |
| ✓ | | 1 | AnAus | 1 | | | | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,2 Sek,0, | 6 | 1 |

Stellt man Rot ans Ende, so läuft der Farbwechsel von links nach rechts.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | 1 | AnAus 1 | | HSV_magenta (pc) | | | Speicher für HSV reservieren | New_HSV_Group() | | | |
| ✓ | | 1 | AnAus 1 | | HSV_blau (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,2 Sek,0, | 1 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_cyan (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,4 Sek,0, | 2 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_gruen (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,6 Sek,0, | 3 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_gelb (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,8 Sek,0, | 4 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_rot (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,10 Sek,0 | 5 | 1 |
| ✓ | | 1 | AnAus 1 | | | | | Muster Pattern_Configurator | APatternT2(#LED,28,#InCh,3,0,255,0,PM_HSV,12 Sek,0 | 6 | 1 |

**Farbwechsel von der Mitte aus in beide Richtungen:**

Achtung: Hier gibt es noch einen kleinen Fehler, den ich noch ausfindig machen muss. Die per Copy-LED-Befehl kopierten LEDs gehen zwar mit an aber nicht wieder aus.

Soll der Farbwechsel beispielsweise auf elf LEDs verteilt werden, bei der die Farbe von der mittleren in beide Richtungen nach außen läuft, so kann man den Copy-LED Befehl nutzen.
Dazu nimmt man den Verlauf von rechts nach links und setzt die Copy-Befehle in umgekehrter Reihenfolge drunter. Selbstverständlich lassen sich „Copy-LED“-Befehle auch zwischen die einzelnen LEDs setzen, um beispielsweise immer ein Pärchen mit derselben Farbe anzusteuern.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | 1 | AnAus 1 | | HSV_rot (pc) | | | Speicher für HSV reservieren | New_HSV_Group() | | | |
| ✓ | | 1 | AnAus 1 | | HSV_gelb (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,12 Sek,0 | 1 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_gruen (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,10 Sek,0 | 2 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_cyan (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,8 Sek,0, | 3 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_blau (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,6 Sek,0, | 4 | 1 |
| ✓ | | 1 | AnAus 1 | | HSV_magenta (pc) | | | Muster Pattern_Configurator | APatternT3(#LED,28,#InCh,3,0,255,0,PM_HSV,4 Sek,0, | 5 | 1 |
| ✓ | | | | | Kopiere LED 5 | | | LED-Werte kopieren | CopyLED(#LED, #InCh, 5) | 6 | 1 |
| ✓ | | | | | Kopiere LED 4 | | | LED-Werte kopieren | CopyLED(#LED, #InCh, 4) | 7 | 1 |
| ✓ | | | | | Kopiere LED 3 | | | LED-Werte kopieren | CopyLED(#LED, #InCh, 3) | 8 | 1 |
| ✓ | | | | | Kopiere LED 2 | | | LED-Werte kopieren | CopyLED(#LED, #InCh, 2) | 9 | 1 |
| ✓ | | | | | Kopiere LED 1 | | | LED-Werte kopieren | CopyLED(#LED, #InCh, 1) | 10 | 1 |
| | | | | | | | | | | 11 | 1 |

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

Permanent link:
**https://wiki.mobaledlib.de/anleitungen/spezial/hsv_mode?rev=1699401359**

Last update: **2023/11/08 00:56**