Die integrierte Warnleuchte

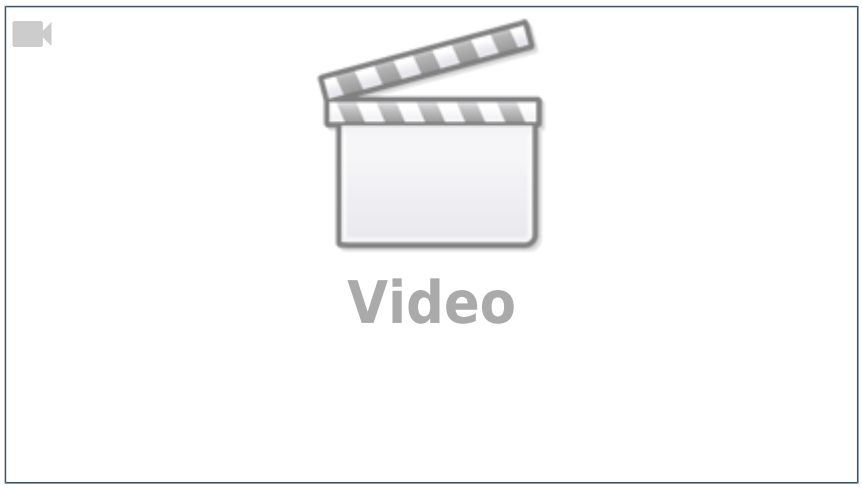
Ausgangsbasis für die folgende Anwendung war eine fixe Idee:
Dort existiert ein Gleisanschluss, der zeitweise zum Programmieren der Lokomotiven genutzt wird, aber zu 99% dem Spielbetrieb dient. Um das zu realisieren, muss dieser Gleisanschluss zweipolig getrennt werden und ganz wichtig: Nach erfolgreicher Programmierung muss er wieder an die Anlage gekoppelt werden. Um diesen letzten Schritt nicht zu vergessen, sollte in unmittelbarer Nähe ein nicht zu übersehendes Warnsignal **blinken**.

Warum also nicht einfach die RGB-LEDs eines in der Nähe stehenden Gebäudes nutzen und z. B. die Neonröhren einer Werkhalle rot blinken lassen?
Zur Generierung des Flackerns einer Neonröhre braucht man jedoch einen Speicher, in dem abgelegt wird, wie viele Zündversuche schon gemacht wurden und ob die Lampe endlich richtig gezündet hat. Diese Daten werden im roten Kanal der LED abgelegt, um Speicher im Arduino zu sparen. Bei jedem Zündversuch wird die Rote LED um ein kleines bisschen heller. Das sieht dann so aus als wäre es die Glimmlampe des Starters. Zur Erkennung, ob die Lampe gerade hell ist, weil ein Zündversuch stattfindet, leuchtet sie nicht mit der vollen Helligkeit, sondern ein kleines bisschen weniger. In diesem „Weniger“ werden wieder die Zündversuche gespeichert. So spart die Programmierung ein zusätzliches Byte. Das ist wichtig, weil der Arduino nur 2000 davon hat und bereits knapp 800 für die LEDs benötigt werden.

Dieser Sparfimmel führt aber zu einem ungewollten Effekt. Die House Funktion (hier mit Neonröhren) prüft die Helligkeit der roten LED, sobald das Licht angeschaltet werden soll. Wenn die LED durch das Blinken bereits leuchtet, dann kommt das Programm durcheinander. Um das zu verhindern, nutzt man als Warnsignal einfach den Fotoblitz. Da dieser nur einen sehr kurzen Impuls sendet, kommt die House-Funktion beim Wiedereinschalten nicht mehr durcheinander. Man kann somit die RGB-LEDs eines Gehäuses sowohl als belebtes Haus, als auch als Warnleuchte in den Farben Rot (C1), Grün (C2), Blau (C3), Gelb (C12), Cyan (C23) und Weiß (C_All) nutzen.

Insgesamt stünden also sechs unterschiedliche Farben für sechs unterschiedliche Warnsignale zur Verfügung. Im Falle des unten gezeugten Programmiergleises sind das:

- Grün für die Programmierung mit der Z21
- Cyan für die Programmierung mit dem ESU LokProgrammer
- Gelb für die Programmierung mit dem Zimo MXDECUP/MXULFA
- Weiß als Reserve für die Programmierung mit einem vierten Programmiergerät
- Rot als Warnleuchte, wenn fälschlicherweise zwei Programmiergeräte aktiv sind



Um das Ganze per DCC zu steuern, werden im Programmgenerator als erstes die DCC-Adressen mit den Schalternamen verknüpft. Im Beispiel ist das Adresse #1 für die Aktivierung des normalen Lichts („Licht_Main“). Die Adressen #11 bis #14 werden zum Umschalten der oben genannten Relais und gleichzeitig zum Aktivieren des Fotoblitzes genutzt. Ihnen werden die Namen „Licht_Z21“, „Licht_ESU“, „Licht_Zimo“ und „Licht_Res“ zugeordnet.

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
✓									Heartbeat LED	RGB_Heartbeat(#LED)	0	1	0	0	0
✓					Objekt: Hallenbeleuchtung mit Warnleuchte										
✓		1	AnAus 0		Neonröhren in belebtem Haus				Logische Verknüpfung	Logic(Licht_Main, #InCh)			1	0	
✓		11	AnAus 0		Warnleuchte rot bei Z21 Programmierung				Logische Verknüpfung	Logic(Licht_Z21, #InCh)			1	0	
✓		12	AnAus 0		Warnleuchte blau bei ESU LokProgrammer				Logische Verknüpfung	Logic(Licht_ESU, #InCh)			1	0	
✓		13	AnAus 0		Warnleuchte gelb bei Zimo MXULFA				Logische Verknüpfung	Logic(Licht_Zimo, #InCh)			1	0	
✓		14	AnAus 0		Warnleuchte weiß als Reserve				Logische Verknüpfung	Logic(Licht_Res, #InCh)			1	0	
✓															
✓		Licht_Main			Licht Neonröhre				Logische Verknüpfung	Logic(Licht_OutN, #InCh AND NOT Licht_Z21 AND N			1	0	
✓		Licht_Z21			Licht Grün - Z21				Logische Verknüpfung	Logic(Licht_OutG, #InCh AND NOT Licht_ESU AND N			1	0	
✓		Licht_ESU			Licht Blau - ESU				Logische Verknüpfung	Logic(Licht_OutB, #InCh AND NOT Licht_Z21 AND N			1	0	
✓		Licht_Zimo			Licht Gelb - Zimo				Logische Verknüpfung	Logic(Licht_OutY, #InCh AND NOT Licht_Z21 AND N			1	0	
✓		Licht_Res			Licht Weiß - Reserve				Logische Verknüpfung	Logic(Licht_OutW, #InCh AND NOT Licht_ESU AND N			1	0	
✓					Licht Rot - Fehler				Logische Verknüpfung	Logic(Licht_OutR, Licht_Z21 AND Licht_ESU OR Li			1	0	
✓															
✓		Licht_OutN			Licht Neonröhre				Belebtes Haus	HouseT(#LED, #InCh, 6, 6, 0, 1, NEON_LIGHTM, NE	1	6	1	0	0
✓					LEDs doppelt zuweisen				LED Nummer manipulieren	// Next_LED(-2)	7	-2	0	0	0
✓		Licht_OutG			Licht Grün - Z21				Blitzlicht	Flash(#LED, C2, #InCh, #LocInCh, 500, 500)	5	C1-1	1	1	0
✓		Licht_OutG							Blitzlicht	Flash(#LED, C2, #InCh, #LocInCh, 500, 500)	6	C1-1	1	1	0
✓									LED Nummer manipulieren	// Next_LED(-2)	7	-2	0	0	0
✓		Licht_OutB			Licht Blau - ESU				Blitzlicht	Flash(#LED, C23, #InCh, #LocInCh, 500, 500)	5	C1-2	1	1	0
✓		Licht_OutB							Blitzlicht	Flash(#LED, C23, #InCh, #LocInCh, 500, 500)	6	C1-2	1	1	0
✓									LED Nummer manipulieren	// Next_LED(-2)	7	-2	0	0	0
✓		Licht_OutY			Licht Gelb - Zimo				Blitzlicht	Flash(#LED, C12, #InCh, #LocInCh, 500, 500)	5	C2-3	1	1	0
✓		Licht_OutY							Blitzlicht	Flash(#LED, C12, #InCh, #LocInCh, 500, 500)	6	C2-3	1	1	0
✓									LED Nummer manipulieren	// Next_LED(-2)	7	-2	0	0	0
✓		Licht_OutW			Licht Weiß - Reserve				Blitzlicht	Flash(#LED, C_ALL, #InCh, #LocInCh, 500, 500)	5	1	1	1	0
✓		Licht_OutW							Blitzlicht	Flash(#LED, C_ALL, #InCh, #LocInCh, 500, 500)	6	C2-3	1	1	0
✓									LED Nummer manipulieren	// Next_LED(-2)	7	-2	0	0	0
✓		Licht_OutR			Licht Rot - Fehler				Blitzlicht	Flash(#LED, C1, #InCh, #LocInCh, 500, 500)	5	1	1	1	0
✓		Licht_OutR							Blitzlicht	Flash(#LED, C1, #InCh, #LocInCh, 500, 500)	6	C2-3	1	1	0

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

Permanent link:
https://wiki.mobaledlib.de/anleitungen/spezial/codevorlagen/warnleuchte?rev=1647875191

Last update: **2022/03/21 16:06**