Extension / Erweiterung

P_Schedule

1 Vorgeschichte:

Die MobaLedLib beinhaltet eine Zeitfunktion bzw. Schedule-Funktion, die zu bestimmten Tag- und Nacht-Werten z.B. das belebte Haus ein- oder ausschaltet.

Dieser Plan gibt nur die groben Rahmenbedingungen vor. Wann die Ausgänge tatsächlich geschaltet werden, bestimmt das Programm zufällig damit ein realer Eindruck entsteht. Geschaltet werden die Ausgangsvariablen „DstVar1“ bis „DstVarN“. Sie werden zufällig zwischen dem Zeitpunkt „Start“ und „End“ eingeschaltet, wenn es „Abend“ ist und genauso Zufällig wieder am „Morgen“ ausgeschaltet. Ob es „Abend“ oder „Morgen“ ist, bestimmt die globale Variable „DayState“. Sie ist „Abends“ auf „SunSet“ gesetzt und „Morgens“ auf „SunRise“. Die zweite Variable „Darkness“ bestimmt über eine Zahl zwischen 0 und 255 wie „Dunkel“ es ist. Damit repräsentiert sie die Zeit.

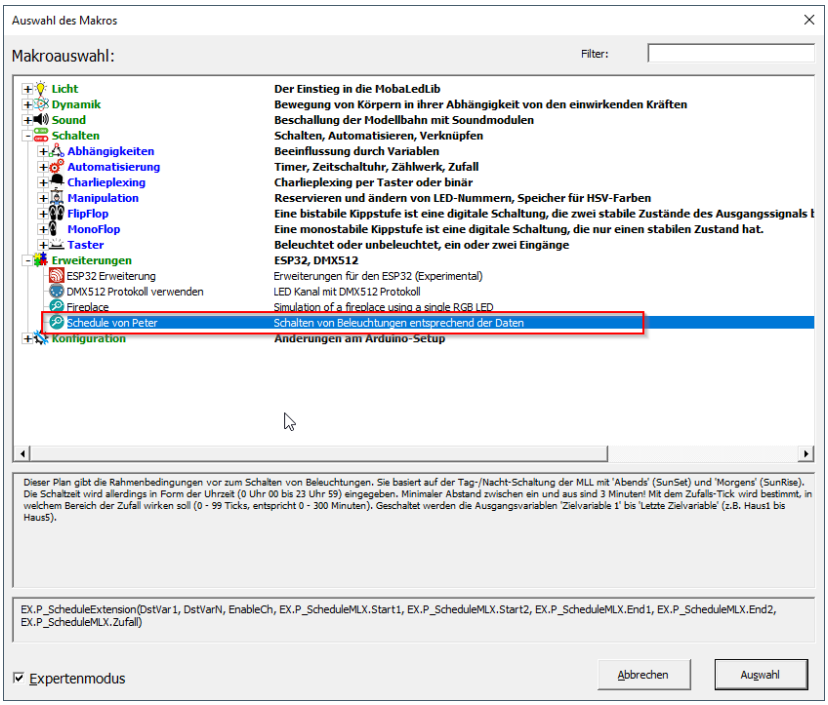
Diese großzügige Schaltvorgang hat mir nicht immer gefallen. Auch waren für bestimmte Zeiten relativ viele zusätzliche Programmierzeilen notwendig. Unten beispielhaft der Aufwand, eine Lampe morgens um 5 Uhr ein- und um 6 Uhr 30 wieder auszuschalten.

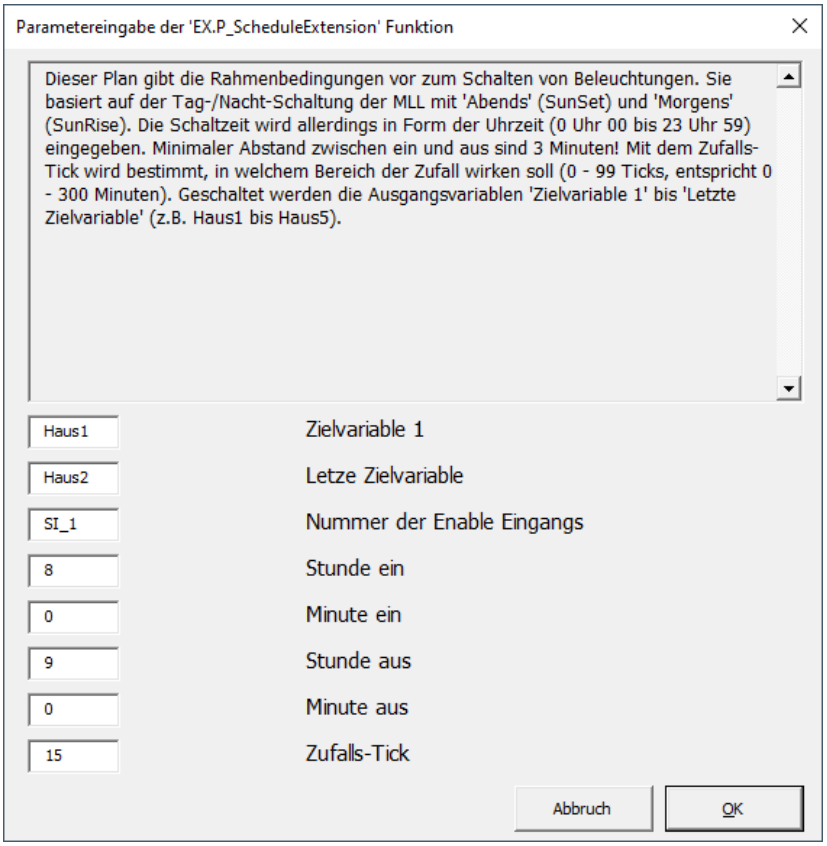
Zeitplan 2: ein 5:00			Zeitplan	Schedule(Hs7_W3, Hs7_W3, SI_1, 149, 149)
Zeitplan3: aus 6:30			Zeitplan	Schedule(Hs7_W4, Hs7_W4, SI_1, 120, 120)
Flanke negieren			Logische Verknüpfung	Logic(Hs7_N3, NOT Hs7_W3)
Flanke negieren			Logische Verknüpfung	Logic(Hs7_N4, NOT Hs7_W4)
Flipflop morgens			RS Flip-Flop	RS_FlipFlop(Hs7_M, #InCh, Hs7_N3)

Daher hab ich mich an dieser Extension versucht.

2 Lösung:

Nachdem das Programm-Paket installiert und der Programm-Generator neu gestartet wurde, ist P_Schedule unter den Erweiterungen zu finden und ähnlich wie das Original Schedule zu verwenden. Nur dass keine Zeitwerte umständlich errechnet und SunRise / SunSet beachtet werden müssen. Es können hier einfach die Zeitwerten mit Stunden und Minuten verwendet werden. Grundlage ist aber trotzdem die MLL-Zeit-Definition.





Variable	Beschreibung	Bereich
Zielvariable 1	Name der ersten Variablen (z.B. Haus1)	
Letzte Zielvariable	Name der letzten Variablen (z.B. Haus2)	
Nummer der Enable Eingangs	Ist immer SI_1	
Stunde ein	Stunde, in der die Zielvariablen aktiviert werden	0 – 23
Minute ein	Minute, in der die Zielvariablen aktiviert werden	0 - 59

Stunde aus	Stunde, in der die Zielvariablen deaktiviert werden	0 – 23
Minute aus	Minute, in der die Zielvariablen deaktiviert werden	0 - 59
Zufalls-Tick	Größe des Zufalls in MLL_Ticks. Ein Tick entspricht etwa 3 Minuten Mit 0 ist der Zufall abgeschaltet	0 -99

Unter Zielvariable 1 gibt man den Namen der ersten Variable, unter letzte Zielvariable den Namen der letzten Variable ein. Diese Namen müssen mit einer Zahl enden. Im Beispiel werden die Variablen Haus1 bis Haus2 verwendet. Diese Variablen werden dann eingeschaltet, wenn die Einschaltzeit erreicht wird. Im Feld „Stunde ein und Minute ein“ wird die Zeit eingegeben, bei der die Häuser eingeschaltet werden sollen. Unter „Stunde aus“ und „Minute aus“ wird der Zeit eingegeben, bei dem die Häuser ausgeschaltet sein sollen. Im Beispiel werden dazu die Zeiten 8 Uhr 00 für ein und 9 Uhr für aus verwendet. Die wirkliche Schaltzeit wird jedoch vom Zufalls-Tick-Wert beeinflusst. Ein Tick entspricht ca. 3 Minuten, d.h. in unserem Beispiel mit dem Wert 15 sind das 45 Minuten. Ist der Schaltzeitpunkt auf 8 Uhr festgelegt, so kann der Zufall zwischen 7 Uhr 15 und 8 Uhr 45 zuschlagen. Hier gibt es allerdings eine Einschränkung. In unserem Beispiel ist der Ausschalt-Zeitpunkt im Bereich von 8 Uhr 15 und 9 Uhr 45. Somit gäbe es eine Überschneidung von 8 Uhr 15 (aus) bis 8 Uhr 45 (ein). Somit könnte das „Aus“ vor dem „Ein“ kommen. Dies wird programmtechnisch aber weitgehend verhindert, in dem der Zufallswert drastisch verkürzt wird (hier auf den Wert 8 (entspricht 24 Minuten).

Ist der Zufalls-Tick auf 0 gesetzt, so gibt es keinen Zufall. Die Aktivierung bzw. Deaktivierung erfolgt dann zur entsprechenden Uhrzeit.

Im Feld Enable_Pin muss SI_1 eingetragen bleiben. Dort wird **nicht** der Pin eingetragen, an dem der Helligkeitssensor angeschlossen ist.

Es können auch mehrere P_Schedule Funktionen gleichzeitig zu verwenden. Dadurch können unterschiedliche Beleuchtungen bei unterschiedlichen Zeitwerten angehen. Man kann z. B. eine P_Schedule Funktion für Häuser und eine eigene Schedule Funktion für Straßenlaternen verwenden. So können dann z. B. die Straßenlaternen vor den Häusern angehen. Es sind beliebige Kombinationen möglich. Man kann damit z.B. auch erreichen, dass Ampeln gelb blinken, wenn es spät abends ist. Nun trägt man im Programm Generator noch die Häuser ein. Für das Beispiel habe ich zwei Häuser mit je einer RGB Leds verwendet. In der Spalte Adresse oder Name trägt man nun bei den beiden Häusern die Variablen Haus1 und Haus2 ein. Nun sollte die Konfiguration so aussehen:

Aktiv	Filter	Adresse oder Name	Typ	Start wert	Beschreibung	Verteiler-Nummer	Stecker-Nummer	Name	Beleuchtung, Sound, oder andere Effekte	Start LedNr	LEDs	InCnt	Loc InCh	LED Sound
✓								Heartbeat LED, einstellbar	RGB_Heartbeat2(#LED, 5, 50)	0	1	0	0	0
✓								Tageszeiten anzeigen	#define DayAndNightTimer_Debug			0	0	
								Tag/Nacht-Modus aktivieren	DayAndNightTimer(#InCh, 16)			1	0	
✓				8:00 - 9:00				Schedule von Peter	EX_P_ScheduleExtension(Haus1, Haus2, SI_1, 8, 0, 9, 0, 15)			0	0	
✓		Haus1						RGB-LED einstellbar	ConstRGB(#LED, #InCh, 0, 0, 0, 127, 127, 127)	1	1	1	0	0
✓		Haus2						RGB-LED einstellbar	ConstRGB(#LED, #InCh, 0, 0, 0, 0, 0, 127)	2	1	1	0	0

3 Nebenwirkungen:

3.1 Speicher:

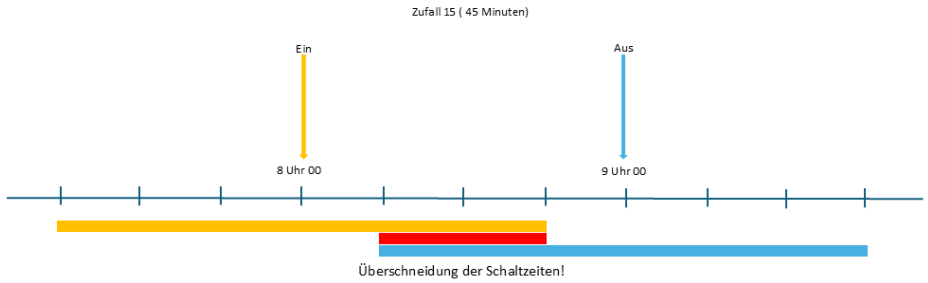
Das Programm braucht Speicher auf dem Arduino oder ESP32. Das ist die einzige sichere Aussage.

Programmspeicher	ca. 930 Byte
Dynamischer Speicher	ca. 30 Byte

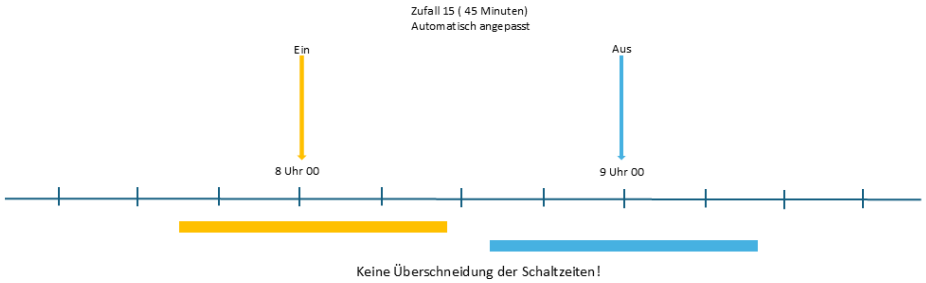
3.2 Einschränkungen:

Der Zufallswert kann programmtechnisch automatisch verkürzt werden, wenn sich Aus- und Einschaltzeit in die Quere kommen.

Beispiel für die Einschaltzeit 8 Uhr 00 und Ausschaltzeit 9 Uhr bei einem Zufallswert von 15 sprich 45 Minuten.



Damit würde es eine Überschneidung von Ein- und Ausschaltzeitpunkt geben und das Ganze zum wirklichen Zufall machen. Das ist aber nicht gewollt. Daher passt das Programm den Zufallswert an (ggf. sogar auf 0).



Eine ähnliche Problematik ergibt sich, wenn mehrere Objekte geschaltet werden sollen (Haus1 bis Haus6 zum Beispiel). Welches Haus (Objekt) geschaltet wird, wenn mehrere Objekte angegeben sind, ist zufällig. Der einzelne Zufall für jedes einzelnes Haus ergibt sich durch den Zufallswert geteilt durch die Objekte (hier z.B. 6). Im ungünstigen Fall kann auch hier der Zufall quasi abgeschaltet sein, wenn die Zeit zu kurz wird. Abhilfe wären dann, mehrere Instanzen von P_Schedule zu verwenden:

P_Schedule(Haus1,Haus1,……)
P_Schedule(Haus2,Haus2,……)
P_Schedule(Haus3,Haus3,……)
P_Schedule(Haus4,Haus4,……)

Bedingt durch die Verwendung der MLL-Ticks muss die Einschalt- und Ausschaltzeit einen Abstand von mindestens 3 eher 4 Minutenhaben. Es geht also nicht um 13 Uhr 10 einzuschalten und um 13 Uhr 12 wieder auszuschalten. Hier wäre ein Aus um 13 Uhr 14 ein guter Wert.

Die Schalt-Uhrzeit kann ggf. von der in der MLL Debugausgabe angezeigten Uhrzeit etwas abweichen. Das ist ein Rundungsproblem, das nur mit wesentlich größerem Speicherverbrauch zu lösen wäre.

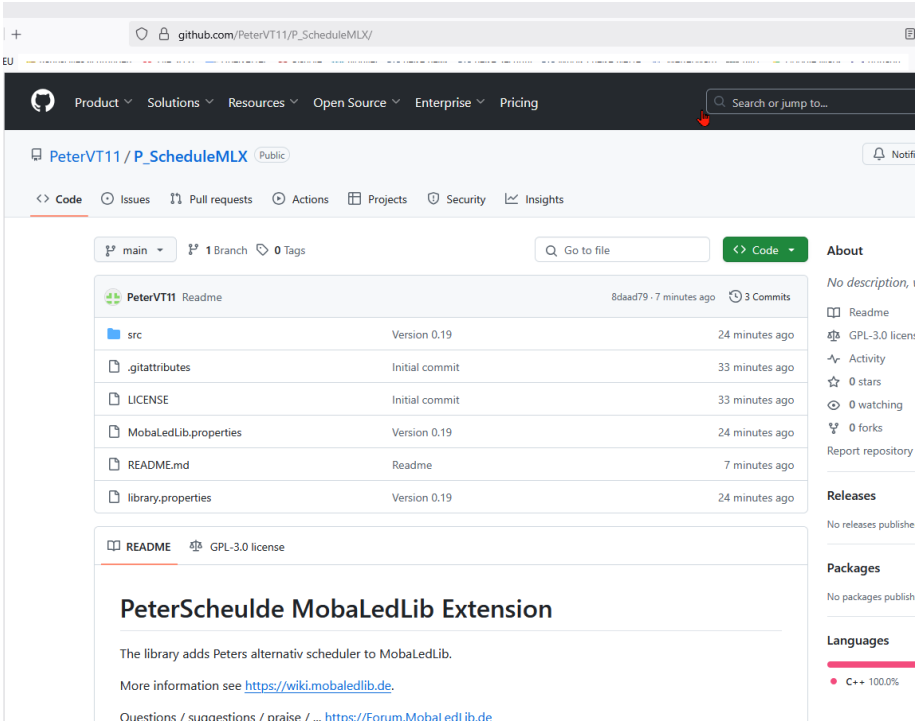
Zu beachten ist auch, dass nach einem Neustart der MobaLedLib die Uhrzeit bei 12 Uhr beginnt. D.h. alle Schaltaufträge davor werden erst am anderen Tag ausgeführt.

4 Installation:

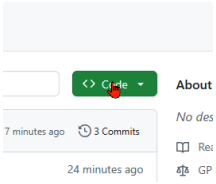
P_Schedule liegt auf Github und kann dort mit der aktuellen Version herunter geladen werden. Hierzu verbindet man sich über den Browser mit Github:

https://github.com/PeterVT11/P_ScheduleMLX

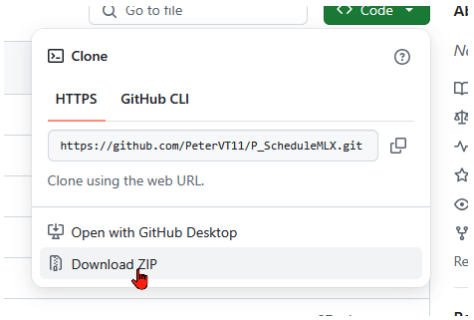
Es öffnet sich die Github-Seite:



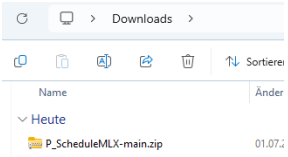
Dort wird auf den grünen Button geklickt:



Dort auf „Download ZIP“ klicken:

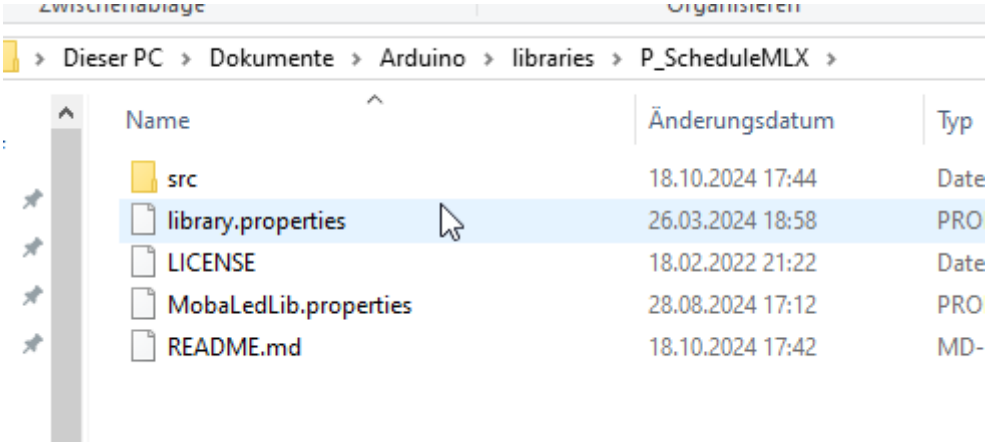


Dann wird die Datei P_ScheduleMLX-main.zip heruntergeladen.



Diese Datei muss dann ins Verzeichnis im Ordner C:\Users\<Benutzername>\Documents\Arduino\libraries entpackt werden.

Der Inhalt sieht dann etwa so aus:



Danach ist nach einem Neustart des Programm-Generators die Funktion unter Erweiterungen zu finden.

5 Version:

0.16	21.10.2024	Betaversion 1
0.16a	22.10.2024	Korrektur Anleitung.
V 0.17	25.10.2024	Fehler bei der Abfrage nach Objekten (beginnen mit 0 nicht 1). Dadurch schalten 2 Objekte gleichzeitig. Debugausgabe, wenn Zufall programmtechnisch gekürzt wird. Beta-Version 2
V 0.18	05.01.2025	Debuging: Dank Jürgen (jueff) gibt es jetzt keinen Kompilerfehler mehr mit Streaming. Debuging: Wenn Zufall verkürzt wird, wird die LED mit ausgeben. Variablendefinition aus Modulen herausgenommen, da Probleme mit ESP32. Ergänzung Doku: Nebenwirkungen. Korrektur für Debug beim ESP32. Ergänzung Doku: Verhalten 12 Uhr nach Reset.
V 0.19	29.06.2025	DstVar global definiert. Raspberry Pico hinzugefügt. Danke Gerald.

6 Autor:

PeterVT11 (https://forum.MobaLedLin.de)

Stand 01.07.2025