Extension / Erweiterung

P_ScheduleDuo

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. Da das für mich nicht optimal war, hab ich P_Schedule entwickelt.

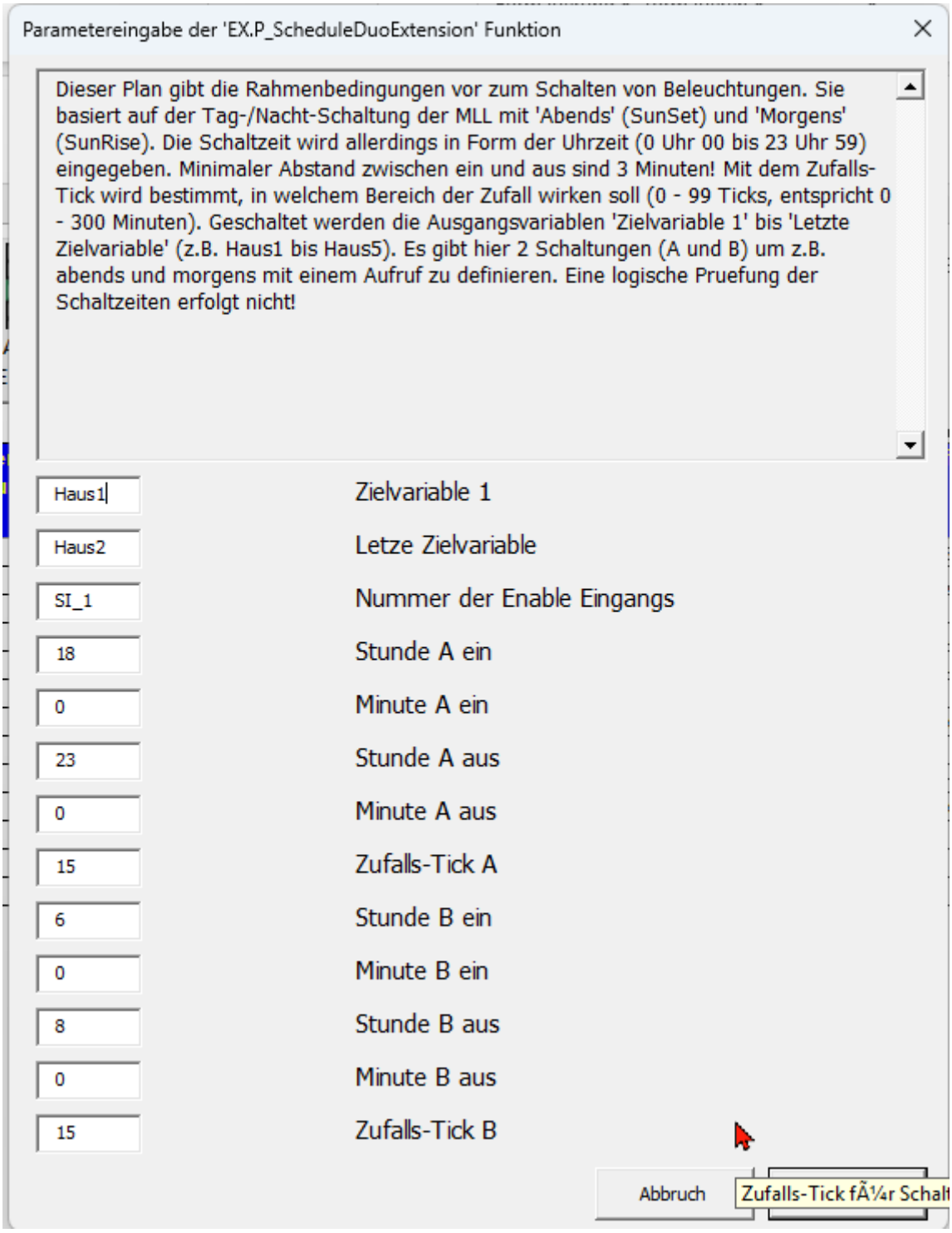
Da ist mir aber aufgefallen, dass meine Preiserchen nur abends aktiv sind und morgens so lange schlafen, bis sie kein Licht mehr benötigen. Klar kann man nach folgendem Beisiel Abhilfe schaffen:

Abends von 18:00 bis 23:00	Schedule von Peter	EX.P_ScheduleExtension(HausA1, HausA11, SI_1, 18, 0, 23, 0,
Morgens von 6:00 bis 8 Uhr	Schedule von Peter	EX.P_ScheduleExtension(HausM1, HausM1, SI_1, 6, 0, 8, 0, 15)
Oder-Verknüpfung	Logische Verknüpfung	Logic(Haus1, HausA1 OR HausM1)
Belebtes Haus	Belebtes Haus	House(#LED, #InCh, 1, 1, ROOM_DARK, ROOM_BRIGHT)

Das war mir aber zu aufwendig. Daher hab ich eine neue Extension P_ScheduleDuo entwickelt.

2 Lösung:

Nachdem das Programm-Paket installiert und der Programm-Generator neu gestartet wurde, ist P_ScheduleDuo unter den Erweiterungen zu finden und ähnlich wie das P_Schedule zu verwenden. Es wird eine A-Zeit und eine B-Zeit eingetragen, damit wird ein Objekt zu zwei unterschiedlichen Zeiten geschaltet. Es können hier einfach die Zeitwerte mit Stunden und Minuten verwendet werden.



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 A ein	Zeitbereich A: Stunde, in der die Zielvariablen aktiviert werden	0 – 23
Minute A ein	Zeitbereich A: Minute, in der die Zielvariablen aktiviert werden	0 - 59

Stunde A aus	Zeitbereich A: Stunde, in der die Zielvariablen deaktiviert werden	0 – 23
Minute A aus	Zeitbereich A: Minute, in der die Zielvariablen deaktiviert werden	0 - 59
Zufalls-Tick A	Zeitbereich A: Größe des Zufalls in MLL_Ticks. Ein Tick entspricht etwa 3 Minuten der MLL-Zeit Mit 0 ist der Zufall abgeschaltet	0 -99
Stunde B ein	Zeitbereich B: Stunde, in der die Zielvariablen aktiviert werden	0 – 23
Minute B ein	Zeitbereich B: Minute, in der die Zielvariablen aktiviert werden	0 - 59
Stunde B aus	Zeitbereich B: Stunde, in der die Zielvariablen deaktiviert werden	0 – 23
Minute B aus	Zeitbereich B: Minute, in der die Zielvariablen deaktiviert werden	0 - 59
Zufalls-Tick B	Zeitbereich B: 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 00 für aus verwendet. Die wirkliche Schaltzeit wird jedoch vom Zufalls-Tick-Wert beeinflusst. Ein Tick entspricht ca. 3 Minuten der MLL-Zeit, 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- und P_ScheduleDuo-Funktionen gleichzeitig zu verwenden. Dadurch können unterschiedliche Beleuchtungen bei unterschiedlichen Zeitwerten angehen. Man kann z. B. eine P_ScheduleDuo 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:

					Tag/Nacht-Modus aktivieren	DayAndNightTimer(#InCh, 16)
					Tageszeiten anzeigen	#define DayAndNightTimer_Debug
		Abends 18:00 - 23:00, morg3ens 6:00 - 8:00			ScheduleDuo von Peter	EX.P_ScheduleDuoExtension(Haus1, Haus2, SI_1, 18, 0, 23, 0, 15, 6, 0, 8, 0, 15)
Haus1		Belebtes Haus			Belebtes Haus	House(#LED, #InCh, 1, 1, ROOM_DARK, ROOM_BRIGHT)
Haus2					Straßenbeleuchtung	GasLights(#LED, #InCh, GAS_LIGHT, GAS_LIGHT)

3 Nebenwirkungen:

3.1 Speicher:

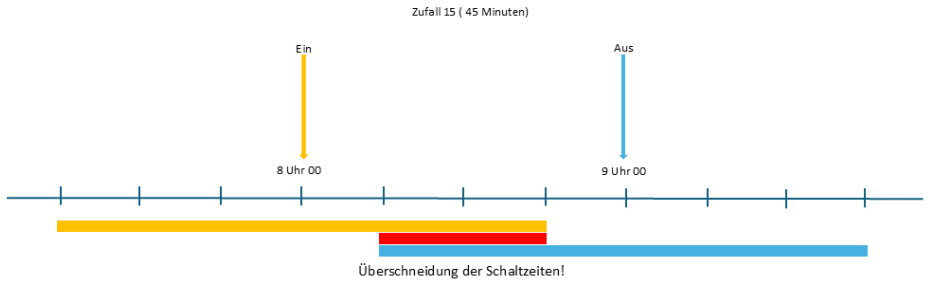
Das Programm braucht Speicher auf dem Arduino, ESP32 oder RP20240. Das ist die einzige sichere Aussage.

Programmspeicher	ca. 200 Byte
Dynamischer Speicher	ca. 4 Byte

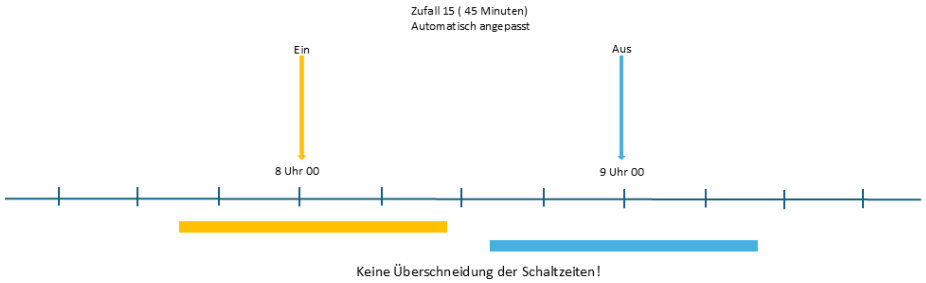
3.2 Einschränkungen:

Der Zufallswert kann programmtechnisch automatisch verkürzt werden, wenn sich Aus- und Einschaltzeiten 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).



Beispiel:

```
EX.P_ScheduleDuoExtension(Haus1, Haus2, SI_1, 6, 10, 10, 0, 30, 12, 0, 14, 0, 30)
```

Das bedeutet:

Schaltzeit A: 6:10 – 10:00 Zufall 30 Ticks, (also 90 Minuten)
Schaltzeit B: 12:00 – 14:00 Zufall 30 Ticks, (also 90 Minuten)

Damit würde sich die Ausschaltzeit A mit der Einschaltzeit B überdecken. Daher wird der Zufall gekürzt. Über die Ausgabe über den seriellen Monitor der Arduino DIE wird das beim Neustart angezeigt (Serieller Monitor mit 115200 Baud):

```
P_ScheduleDuo: Zeit A: 06:10 - 10:00, Zeit B: 12:00 - 14:00
!!! P_ScheduleDuo:  Zufall-Wert B für LED: 0  verkürzt von 30 auf 17 Ticks
!!! P_ScheduleDuo: Zeitbereiche überschneiden sich! (1)
!!! P_ScheduleDuo: Zeit A: 06:10 - 10:00  Zufall: +/- 30 Ticks
!!! P_ScheduleDuo: Zeit B: 12:00 - 14:00  Zufall: +/- 17 Ticks
```

Eine Verkürzung oder Änderung der Schaltzeiten wird nicht durchgeführt, was ggf. zu seltsamen Ereignissen führt.

Beispiel:

```
EX.P_ScheduleDuoExtension(Haus1, Haus2, SI_1, 6, 12, 14, 0, 0, 11, 0, 12, 0, 0)
```

Das bedeutet:

Schaltzeit A: 6:12 – 12:00 Zufall 0 Ticks, (also kein Zufall)
Schaltzeit B: 11:00 – 14:00 Zufall 0 Ticks, (also kein Zufall)

Schalt-Aus A ist nach dem Einschalten von B. Auch dies wird über die serielle Schnittstelle gemeldet.

```
P_ScheduleDuo: Zeit A: 06:12 - 14:00, Zeit B: 11:00 - 12:00
!!! P_ScheduleDuo: Zeitbereiche überschneiden sich! (1)
!!! P_ScheduleDuo: Zeit A: 06:12 - 14:00  Zufall: +/- 0 Ticks
!!! P_ScheduleDuo: Zeit B: 11:00 - 12:00  Zufall: +/- 0 Ticks
```

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,.....)

Wird hierbei der Zufall auf 0 gesetzt, so wird trotzdem eine kleine Pause von 100mSec. Zwischen den einzelnen Schaltobjekten eingefügt.

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 besserer Wert.

Gleiches gilt zwischen den Schaltzeiten A und B. Hier benötigen wir auch mindestens einen kleinen Abstand.

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.

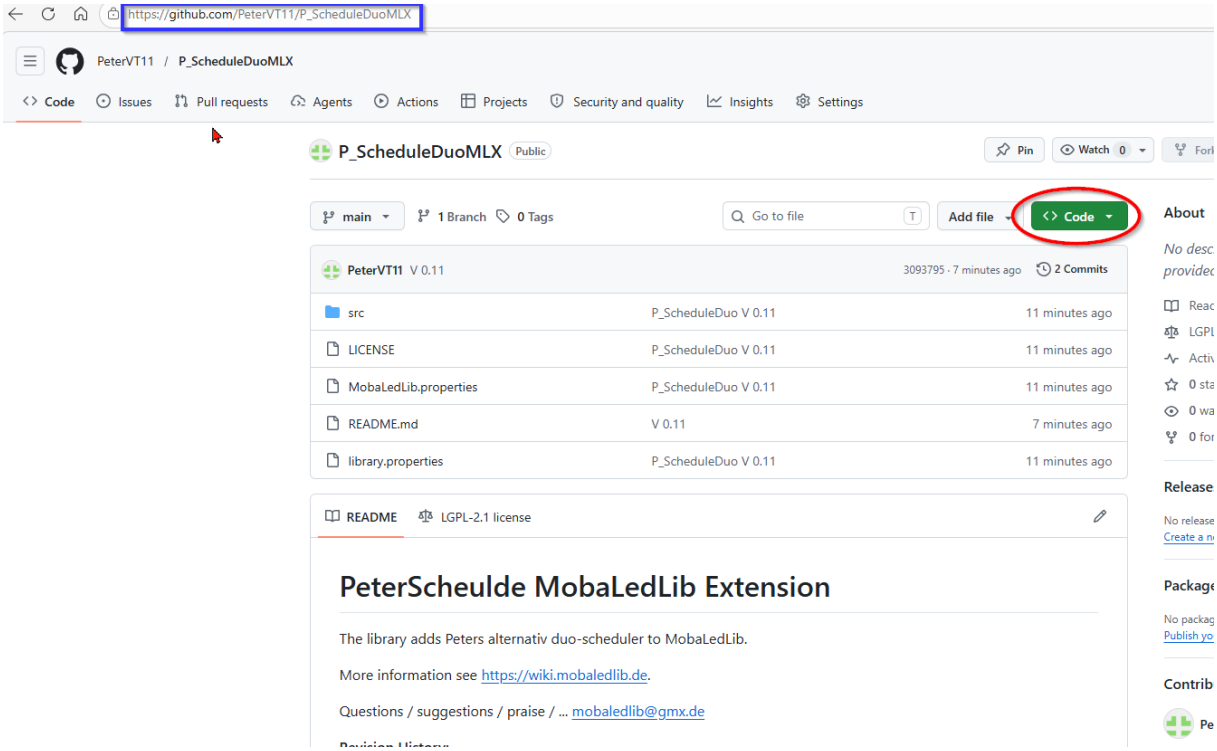
Es wird bei einem Neustart geprüft, ob ein Schaltauftrag in der Zeit von 0 Uhr bis 12 Uhr vorliegt und die Ausschaltzeit nach 12 Uhr ist. In diesem Fall wird der Schaltauftrag sofort ausgeführt.

4 Installation:

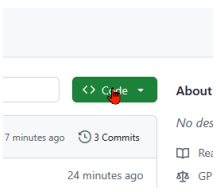
P_Schedule liegt auf Github und kann dort mit der aktuellen Version herunter geladen werden. Hierzu verbindet man sich über den Browser mit Github:

https://github.com/PeterVT11/P_ScheduleDuoMLX

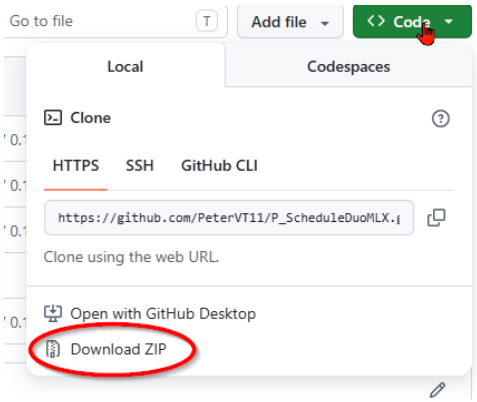
Es öffnet sich die Github-Seite:



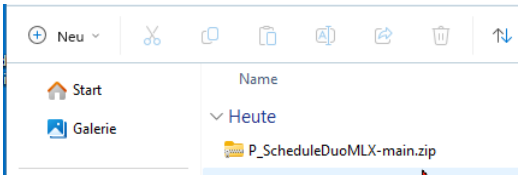
Dort wird auf den grünen Button geklickt:



Dort auf „Download ZIP“ klicken:

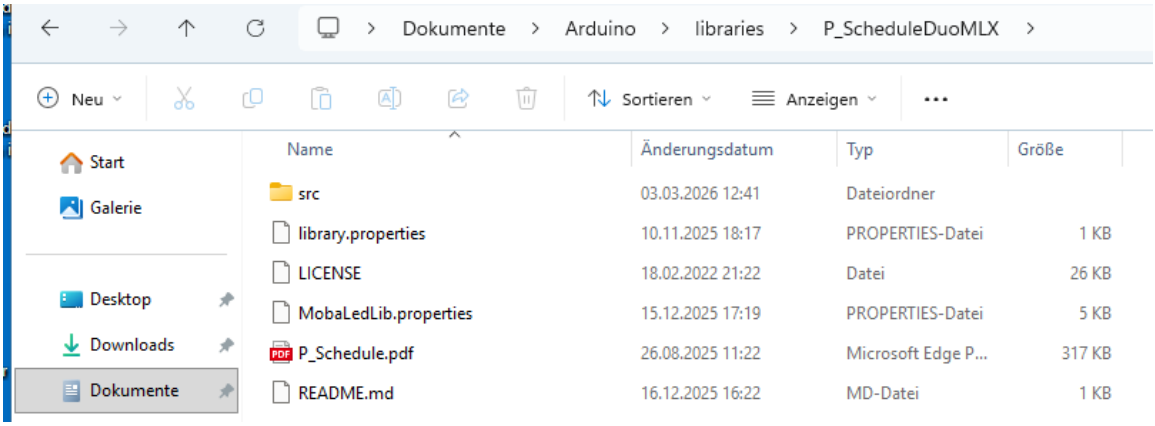


Dann wird die Datei P_ScheduleDuoMLX-main.zip heruntergeladen.



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:

0.10 16.12.2025 Betaversion 1
0.11 06.03.2026 Anpassung my_FirstLedStatus auf P_Schedule 0.21. und Debugausgaben.

6 Autor:

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

Stand 21.04.2026