Python-GUI zur Steuerung von DALI-Leuchten

In diesem Thread hatte ich vor einiger Zeit ein RS232-DALI-Gateway und später eine GUI zum Schalten von symbolischen Leuchten per Mausklick vorgestellt.

In diesem Beitrag sind beide Ansätze vereint, nun können in einem Grundriss Leuchten angewählt werden, die holen die aktuellen Helligkeitswerte aus der DALI-Leuchte im Raum und öffnen einen Dialog zum Schalten von Einzelleuchten oder Leuchtengruppen.

Das GUI-Projekt ist noch einmal gründlich überarbeitet und aufgeräumt worden.
Die Kommunikation mit den echten DALI-Geräten am DALI-Bus ist implementiert und in einem Modul zusammengefasst.
Und die Definition des "baulichen Projektes" ist in eine XML-Datei ausgelagert.

Um die Arbeitsweise praktisch ausprobieren und nachvollziehen zu können, wird nun allerdings ein DALI-Bus mit realen DALI-Leuchten sowie ein geeignetes Gateway benötigt.
Ok, genaugenommen ist nur eine Serielle Schnittstelle erforderlich, dann startet das Programm zumindest - es tut aber nichts wirklich nützliches.

Im Folgenden bemühe ich mich, die Arbeitsweise der GUI, die Schnittstelle zum DALI-Bus und den Aufbau der Projektdefinition in der XML-Datei ausführlicher zu erhellen.

Definition eines baulichen Projekt

Die Projektdefinition ist ausgelagert in eine XML-Datei.
Das Modul "readxml.py" ist zuständig dafür, diese Datei einzulesen, die Daten zu extrahieren und an die Anwendung zu übergeben.

Der Aufbau einer XML-Datei sollte im Grundsatz bekannt sein, erklärt sich aber beim Ansehen fast von allein: alle Daten sind durch <Tags> gerahmt. Tags können hierarchisch aufgebaut sein.

Die oberste Hierarchieebene ist <Projekt> </Projekt>.

Es folgen zunächst allgemeine Definitionen zum "baulichen Projekt":

```
<projekt>
    <scale>682/2170</scale>
    <floorplanfile>Beispiel_746x500a.gif</floorplanfile>
    <floorplansize>746,500</floorplansize>
    <offset>25,32</offset>
    <floorwindowsize>746,500</floorwindowsize>
    <projektname>Beispielprojekt - Erdgeschoss</projektname>
    ...
    ...
    ...
</projekt>
```

Die allgemeinen Projekttags haben folgende Bedeutung:

<scale> wird benutzt, um die Geometrie der Räume und die Standorte der Leuchten, die in der Einheit Zentimeter eingegeben wird, auf die Einheit Pixel der Bitmap umzurechnen:
682 Pixel entsprechen 2170 cm.

<floorplanfile> ist der Name der Bitmap für die Grundrissdarstellung.

<floorplansize> gibt die Breite / Höhe der Grundrissbitmap in Pixeln an.

<floorwindowsize> gibt an, wie breit / hoch (in Pixeln) das Window für den Grundriss sein soll. Ist das Window kleiner als die Bitmap, dann werden Scrollbars eingeblendet, mit denen der sichtbare Ausschnitt verschoben werden kann.

<offset> gibt den Abstand des Projektursprunges von der oberen, linken Bitmapecke in Pixeln an. Alle Raum(teil)flächen werden bezogen auf diesen Projektursprung in der Einheit Zentimeter angegeben.

<projektname> ist hoffentlich selbstredend.

Innerhalb des Moduls "readxml.py" werden die Tags eingelesen und es erfolgt eine Umrechnung der Maßangaben von der Einheit Zentimeter auf Pixelkoordinaten bezogen auf den Ursprung und die Größe der Grundrissbitmap.

Nach den allgemeinen Definitionen folgt die Liste der Räume, der Leuchten (und ggf. der Schalter). Geometrische Eingaben für Räume und Standorte von Objekten erfolgen immer in wahre Längen/Abständen in der Einheit Zentimeter - bei Raumfläche bezogen auf den Projektursprung, bei Objekten wie Leuchten und Schaltern bezogen auf den Raumursprung.

Der Projektursprung wird über einen Offset bezogen auf die obere, linke Ecke der Bitmap beschrieben, der Raumursprung wird durch die obere, linke Ecke des ersten Rechteckes innerhalb einer Raumdefinition festgelegt

Die interne Umrechnung in Pixel erfolgt bei der Initialisierung der Räume, Leuchten und Schalter.

Der Vorteil dieser Vorgehensweise ist: sollte die Bitmap skaliert oder ihr Bildrand beschnitten werden, dann muss in der XML-Datei nur an den drei Parametern scale, floorplansize und offset geschraubt werden - die Räume und Leuchten werden anschließend automatisch wieder an den korrekten Positionen eingefügt.

Die Raumdefinition in der XML-Datei

```
    <room>
        <name>Aufenthaltsraum</name>
        <light_group_id>1</light_group_id>
        <area>160,0,680,780</area>
        <area>0,380,850,390</area>
    </room>
```

<name> Jeder Raum sollte eine Bezeichnung, einen Namen besitzen.

<light_group_id> legt fest, welche gemeinsame Gruppen_id alle Leuchten eines Raumes haben. Eine light_group_id > 15 heißt, dass in diesem Raum keine gemeinesame Gruppen_id existiert (bekanntlich gibt es nur 16 DALI-Gruppen).
In diesem Raum können alle Leuchten aber dennoch erkannt werden, wenn im Raumobjekt durch die Leuchtenliste iteriert wird.

Die realen Leuchten müssen selbstverständlich genau über die oben erwähnte Gruppenzuordnung verfügen, denn intern werden die Leuchten über DALI-Gruppen angesprochen.

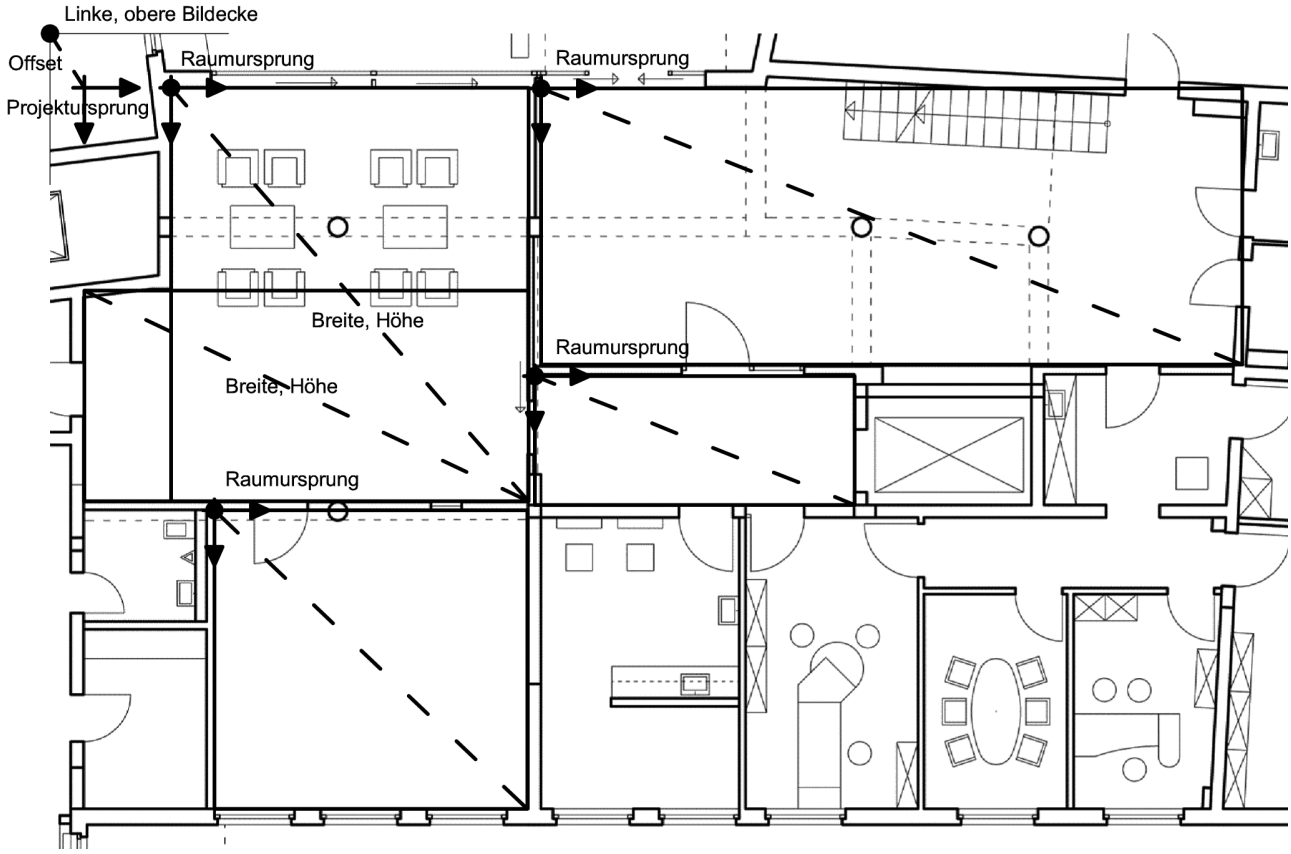
<area> die von Räumen überdeckte Grundfläche muss durch Rechtecke definiert werden:

Ein Rechteck wird durch die Koordinaten seiner oberen, linken Ecke sowie der Angabe der Breite und der Höhe festgelegt. Die Einheit ist in beiden Fällen ist Zentimeter, also die wahre Größe.

Manchmal ist eine Raum nicht einfach nur rechteckig.
Dann muss seine Grundfläche mit Hilfe mehrerer Rechtecke beschrieben werden. Die Verwendung von mehreren Tags <area> ist zulässig.
Die Rechtecke eines Raumes dürfen auch ohne Risiken und Nebenwirkung überlappen.

Das zuallererst angegebene Rechteck eines Raumes hat eine zusätzliche Bedeutung:
seine obere, linke Ecke bestimmt den Raumursprung.

Alle Leuchten und Schalter eines Raumes ermitteln ihre Position relativ zum Raumursprung.
Warum? Nun, verschiebt man einen Raum, dann folgen alle untergeordneten Objekte automatisch nach.



Die Leuchtendefinition in der XML-Datei

```
    <light>
        <name>Rasterleuchte 1</name>
        <room_id>0</room_id>
        <xy>125,250</xy>
        <dali_id>0</dali_id>
        <group>1</group>
        <dimmable>1</dimmable>
        <shape>square</shape>
        <length>20</length>
    </light>
```

<light> jede Leuchte hat einen Namen.

<room_id> jede Leuchte ist logisch über die room_id einem Raum zugeordnet.

Wie lautet nun aber die room_id eines Raumes? Die Räume verfügen ja über kein entsprechendes Attribut.
Die room_id wird einfach über den Index eines Raumes innerhalb der Raumliste gebildet.
Der zuerst angelegte Raum hat den Index 0, der nächste den Index 1.
Die Reihenfolge der Definition entscheidet daher über die room_id.

<xy> Die Position einer Leuchte wird bezogen auf den Raumursprung, der Abstand wird in der Einheit Zentimeter (x, y) angegeben.

<dali_id> jeder Leuchte muss eine eindeutige DALI-id zugewiesen werden, über die sie via DALI-Bus adressiert werden kann. Es versteht sich von selbst, dass diese id mit der DALI-id der realen Leuchte übereinstimmen muss.

<group> jede Leuchte kann einer einzigen Gruppe zugeordnet werden (das gilt hier nur für die Python-Umgebung, nicht für die DALI-Leuchte selbst, die Mitglied in bis zu 16 Gruppen sein kann). Alle Leuchten der gleichen Gruppe eines Raumes können aus einem später noch zu beschreibenden Dialog heraus gemeinsam bearbeitet werden.
Die Gruppen_id wird neben dem Leuchtensymbol auf dem Bildschirm eingeblendet.

<dimmable> eine Leuchte kann dimmbar sein - oder auch nicht.
Bei nicht dimmbaren Leuchten ist der Schieberegler für die Helligkeitseinstellung disabled.

<shape> eine Leuchte kann verschiedene Darstellungsformen im Grundriss einnehmen, die verfügbaren shapes sind: "square", "round", "diamond", "rectangle_h" und "rectangle_v"

<size> die Größe des Leuchtensymbols kann über einen Wert justiert werden.

Die Schalterdefinition in der XML-Datei

```
<switch>
    <xy>650,760</xy>
    <room_id>1</room_id>
    <group>5</group>
</switch>
```

<xy> der Standort jedes Schalters ist wieder bezogen auf den Raumursprung definiert, die Abstände werden in der Einheit Zentimeter (x, y) erwartet.
<room_id> der Raum, dem der Schalter zugeordnet ist.
<group> ein Schalter schaltet als Wechselschalter alle Leuchten dieser DALI-Gruppe.

Anmerkung:
Das Konstrukt der 'Schalter' ist vermutlich überflüssig, ich habe es aber vorläufig erst einmal beibehalten.

Wie funktioniert die grafische Oberfläche

Die Räume

Grundlage für die Grafik ist ein Canvas, dem als Hintergrund ein Bitmap zugewiesen wird.
Dieses Bitmap ist sinnigerweise eine geeignete Grundrissdarstellung.

Ein Bitmap ist bekannterweise nicht übermäßig klug, es kennt keine Räume, obwohl es im geistigen Auge des Betrachters Räume darstellt.
Um innerhalb einer Bitmap als Grundriss separate Räume selektieren zu können, muss eine Hilfskonstruktion herangezogen werden:

Räume sind Python-Objekte mit diversen Attributen.
Jeder Raum verfügt eine Liste von (Teil-)Flächen.
Diese (Teil-) Flächen werden durch Rechtecke definiert, deren diagonal gegenüberliegende Eckpunkte innerhalb der Flächenliste des Raumes abgelegt sind.
Die Gesamtheit der rechteckigen Fläche eines Raumes definieren sozusagen einen Teppichboden, der seine gesamte Grundrissfläche bedeckt.

Hinweis: In der Flächenliste sind die Rechtecke durch die Bitmapkoordninaten der oberen, linken und der unteren, rechten Ecke abgelegt, nicht wie in der XML-Datei durch obere, linke Ecke plus Breite und Höhe.

Wenn der Anwender die Rechte-Maustaste betätigen, dann wird ein Event ausgelöst, durch das wiederum eine Funktion aufgerufen wird, der die Mausposition als Parameter übergeben wird.

Nun muss innerhalb der Listen aller Räume deren jeweilige Flächenliste daraufhin untersucht werden, ob der Mausklick innerhalb oder außerhalb der Teilflächen erfolgte.

Lag der Mausklick innerhalb einer Fläche, dann meldet der übergeordnete Raum seine Id als Erfolgsmeldung zurück, damit ist der angeklickte Raum identifiziert.

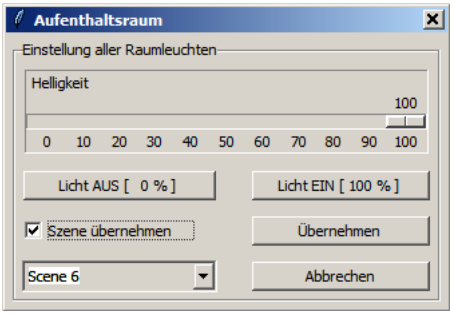
Berücksichtigt werden muss dabei noch, dass ein Mausevent die Position bezogen auf den aktuellen Fensterausschnitt liefert, nicht bezogen auf den Ursprung der Bitmap.

Der Unterschied kommt zum Tragen, wenn die Grundrissgrafik größer ist als der Fensterausschnitt. In diesem Fall kann die Grafik mittels Scrollbars verschoben worden sein.
Glücklicherweise kennt ein Canvas die beiden Methoden canvasx() und canvasy().
Werden diesen Methoden die Windows-Koordinaten übergeben, dann erhält man die Bitmap-Koordinaten zurückgeliefert.

Nachdem der ausgewählte Raum bekannt ist, wird ein Fenster geöffnet, in dem für alle Leuchten eines Raumes gewählt werden kann:

- Einstellung der Helligkeit für aller Leuchten des gewählten Raumes
- Auswahl einer Szene für alle Leuchten des gewählten Raumes

Raumdialog



Die Leuchten

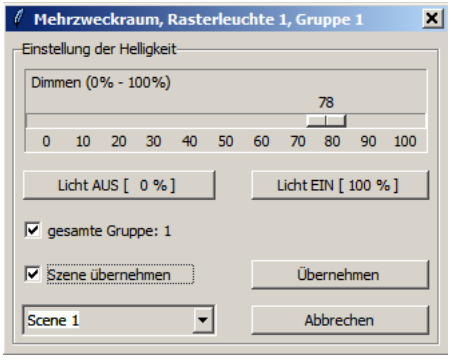
Jede Leuchte ist eine eigenständiges Python-Objekt, hat einen Namen, ist einem Raum zugeordnet, kennt ihre DALI-id und ist einer DALI-Gruppe zugeordnet.

Wird eine Leuchte mit der Linke_Maustaste ausgewählt, dann kann sie in ihren Attributen nachsehen, zu welchem Raum sie gehört, kann mittels ihrer eigenen DALI-id die aktuelle Helligkeit aus der realen Leuchte auslesen und anschließend einen Dialog öffnen.

In den Dialogen sind Einstellungen veränderbar, wobei die Veränderung auf eine einzelne Leuchte, auf alle Leuchte, die der Gruppe der gewählten Leuchte zugehören oder gar für alle Leuchten eines Raumes bezogen sein können.

Abhängig davon, ob die Leuchte mit Linke_Maustaste oder mit Shift_Linke_Maustaste ausgewählt wurde, werden unterschiedliche Dialoge aufgerufen.

Leuchtendialog



Die <Linke Maustaste> ruft den Leuchtendialog auf, hier kann verändert werden:

- Einstellung der Helligkeit der gewählten Leuchte
- Einstellung der Helligkeit für die gesamte Gruppe der gewählten Leuchte
- Auswahl einer Szene für die gewählte Leuchte
- Auswahl einer Szene für die gesamte Gruppe der gewählten Leuchte

Um alle Leuchten eines Raumes zu schalten gibt es zwei unterschiedliche Ansätze:

1.) Jeder Raum kennt alle ihm zugeordneten Leuchten. Es kann daher durch die Liste der Leuchten iterierte werden, alle Leuchten werden nacheinander geschaltet.
Der DALI-Protokoll zeichnet sich nicht durch hohe Geschwindigkeit aus, daher kann diese Methode zu sichtbaren Verzögerung zwischen den Schaltvorgängen führen.

2.) Jedem Raum kann eine light_room_id zugewiesen werden (Werte > 15 werden ignoriert).

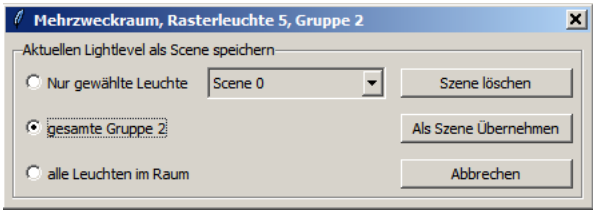
Ist eine id < 16 zugewiesen, dann kann der gesamte Schaltvorgang durch ein einziges DALI-Kommando an die gesamte Gruppe (= alle Leuchten eines Raumes) übermittelt werden.

Voraussetzung: Das Attribut room_group_id < 16 (d.h. auf eine gültige DALI-Gruppe verweisend) und dass es reale Leuchten gibt, die Mitglied dieser Gruppe sind.

Ist die room_group_id > 15 definiert, dann wird die Suche nach den Leuchten des Raumes über die Methode 1.) durchgeführt.

Mit <Shift Linke Maustaste> auf ein Leuchtensymbol erscheint der

Szenendialog



Hier können Szenen definiert oder gelöscht werden:

- Speichern der aktuellen Helligkeitseinstellung als Szene für eine einzelne Leuchte
- Speichern der jeweils aktuellen Helligkeitseinstellung als Szene für die gesamte Gruppe
- Speichern der jeweils aktuellen Helligkeitseinstellung als Szene für alle Leuchten des Raumes
- Löschen einer Szene für die gewählte Leuchte
- Löschen einer Szene für die gesamte Gruppe der gewählten Leuchte
- Löschen einer Szene für alle Leuchten im Raum

Darstellung des Leuchtensymbols

Die Leuchtensymbole sind durch ein schwarzes Rechteck oder einen schwarzen Kreis gerahmt. Die innere Fläche des Leuchtensymbols verändert ihre Helligkeit in Abhängigkeit von der aktuellen Helligkeit der zugeordneten Leuchte.

Wird eine einzelne Leuchte mit der Linke_Maustaste angewählt, dann wird zuerst der Helligkeitswert aus der realen DALI-Leuchte ausgelesen.
Gelingt das nicht (weil die Leuchte nicht antwortet), dann wird das Leuchtensymbol rot hinterlegt.

Nach dem Aufruf von Szenen werden immer alle Szenenwerte aus der Leuchte ausgelesen und den Symbolen auf dem Bildschirm zugewiesen. Bei diesem Vorgang werden nicht antwortende Leuchten erkannt und rot markiert.

Warum werden Szenenwerte ausgelesen und nicht die aktuelle Helligkeitseinstellung der Leuchte ? Weil der Übergang vom Ist-Wert zum Soll-Wert (=Szenenwert) durch die Faderate bestimmt wird und sich über einen längeren Zeitraum hinziehen kann.

Bei der Zuweisung von Helligkeiten über DALI-Gruppen werden die Helligkeitswerte den Symbolen zugewiesen, ohne dass dabei die Helligkeit noch einmal aus den Geräte ausgelesen wird. Dadurch werden auch Geräte, die vorher als fehlerhaft erkannt waren, wieder mit einem Helligkeitswert dargestellt - obwohl sie weiterhin nicht präsent sind.

Die Schalter

Auch die Schalter sind Objekte. Sie werden als gefüllte Kreise dargestellt.
Sie sind einer DALI-Gruppe zugeordnet und schalten den Status aller Leuchten dieser Gruppe um. Wobei alle Leuchten als gemeinsamen Status entweder "ON" oder"OFF" kennen.

Der aktuelle Status wird innerhalb des Schalters gespeichert, er nimmt keine Rücksicht auf den aktuellen Status der zugeordneten Leuchtengruppe.

Die Schalter werden z.Z. nur auf dem Bildschirm genutzt, sie schalten keine Leuchten.

Kommunikation mit den DALI-Ballasts

Das Modul "daligateway.py" ist die Schnittstelle zum DALI-Bus.
Das Objekt "DaliGateWay()" stellt Methoden bereit, um mit den Slaves zu kommunizieren:

```
set_light_by_adr(adr, level) - Sendet den Lightlevel an eine einzelne Adr
set_light_by_grp(grp, level) - Sendet den Lightlevel an eine ganze Gruppe
set_dtr(value) - Schreibt den Wert value in das Register DTR
set_light_2_dtr(adr) - Schreibt den Lightlevel von Adr ins Register DTR
set_dtr_2_scene(scene) - Schreibt den Wert aus DTR in ein Szenenregister
def recall_scene_by_adr(adr, scene) - Ruft in einer Adr die Szene auf
def recall_scene_by_grp(grp, scene) - Ruft in einer Gruppe die Szene auf
def get_light(adr) - Holt den aktuellen Lightlevel von Adr
def get_scene(adr, scene) - Holt den Szenenwert von scene aus Adr
```

Als Hardware verwende ich das in diesem Thread beschrieben "RS232-DALI-Gateway".
In der dortigen Readme.pdf gibt es eine Beschreibung dazu. Hier nur kurz, aber hoffentlich hinreichnd prägnant zusammengefasst:

Das Gateway erwartet einen Datensatz mit genau 4 Byte:

0xA5 konstantes Startbyte
0xXX DALI-Command
0xXX DALI-Data
0xAX (0xA1 = 1x senden oder 0xA2 = 2x senden)

Die beiden mittleren Bytes werden unverändert auf dem DALI-Bus gesendet.
Im Hintergrund lauscht das Gateway auf eine Antwort des adressierten Slaves.

Nach spätestens 25ms wird eine Antwort bestehend aus drei Byte zurückgeliefert:

0xA5 konstantes Startbyte
0xXX Statusflag: 0 = Bytes empfangen, 1 = keine Antwort, >1 Fehler
0xXX Datenbyte, aber nur dann gültig, wenn Statusflag = 0

DaliGayteWay startet einen Hintergrundthread, der an der Seriellen Schnittstelle lauscht.
Werden Daten im beschriebenen Format empfangen, dann liefert der Thread via Callback ein Tuple aus 2 Byte zurück (siehe oben):

0xXX Statusflag: 0 = Bytes empfangen, 1 = keine Antwort, >1 Fehler
0xXX Datenbyte, aber nur dann gültig, wenn Statusflag = 0

Diejenigen Methoden von DaliGateWay(), die eine Rückgabe liefern, geben genau 1 Byte zurück:

0 .. 255 als gültiges Datenbyte oder aber -1 bei einem Fehler.

Die Zuweisung von -1 an die Helligkeit der Leuchtensymbolik läßt das Symbol rot anlaufen ...

09. April 2016

Michael S.

PS.

Wenn sich der Leser fragt, wozu ich diese readme schreibe - mir ist bewusst, dass es nur wenige Interessenten für die Kombination Python + DALI + Mikrocontroller geben wird.

Dieses Dokument dient einmal dazu, dass ich in einem halben Jahr schnell wieder verstehen kann, was ich hier programmiert habe.

Zum anderen führt die Beschreibung der Zusammenhänge dazu, dass Denkfehler, unsinnige Funktionbezeichnungen und unsystematischer Programmablauf aufgedeckt werden.

Während ich hier am Schreiben bin, werden permanent noch Änderungen im Programmcode vorgenommen (und neue Fehler eingebaut).

Quellen zu DALI-Protokoll und DALI Command Set:

NXP Semiconductors - AN10760 - USB-DALI master using the LCP2141
Microchip - AN 1465 - Digitally Addressable Lighting Interface (DALI) Communication
Microchip - AN 1487 - DALI Control Gear
Tridonic, Dokumentation zu masterConfigurator 2.20, Dali Command Set Seite 247 - 252 (masterConfigurator_de.pdf)