DMX-Szenen

Speichern der beliebten Farben der RGB Lampen ist eine grundsätzliche Anforderung bei der Steuerung der Farbenbeleuchtung. Wir haben eine Lösung erstellt, die das Speichern der beliebigen Anzahl der DMX-Szenen erlaubt.

Was erfahren Sie von diesem Tutorial?

Von diesem Tutorial erfahren Sie wie die Schnittstellen zu erstellen, mit den Sie eine beliebige Anzahl der DMX Szenen mit Hilfe der Menu des **Remote** sowie der Visualisierung speichern/aufrufen können. Um diese Funktionalität zu realisieren, brauchen Sie keine Software. Der ganze notwendige Code befindet sich im Skript `rgbmem.lua`, der in der Aktualisierung der Komponente `resources` in der Version `130308` für das **Base** Modul verfügbar ist. Sie können den Code des Skriptes in **Ressourcen** > **Skripten** > **dmxscene.lua** schauen. Wenn es in Ihrem Modul kein Skript gibt, können Sie es manuell hinzufügen. Zu diesem Zweck klicken Sie auf **Hinzufügen** und dann merken Sie die Datei auf die lokale Festplatte. Das Skript ist im Anhang an diesem Tutorial verfügbar.

1. Deklaration der Gruppen und Szenen

Um die Lichtszenen zu erstellen, importieren Sie das Skript `dmxscene.lua` ins Lesezeichen **Logik**. Dazu dient das Kommando `import '<skript>'`, in diesem Fall `import 'dmxscene'`.

Um die Verwaltung der Szenen noch intuitiver zu machen, gruppiert das Skript dieses Szene automatisch nach von Ihnen angegebenen Namen. Dadurch können Sie mehreren Szenen erstellen, die mit einem Zimmer oder bestimmter Lichtquelle verbunden werden. Zu einer Gruppe können Sie eine beliebige Anzahl der Szenen hinzufügen. Es ist sehr einfach eine neue Gruppe zu erstellen. Fügen Sie im Lesezeichen **Logik** eine einzelne Linie des Codes nach dem folgenden Schema hinzu:

`<variable> = dmxscene('<gruppe>',...)`, wo `<variable>` ein beliebiger Name der Variable ist, die mit einer Gruppe verbunden ist. Im weiteren Teil des Tutorials werden die Variable als "Objekte" genannt. `<gruppe>` ist ein beliebiger Name der Gruppe. Statt `...` geben Sie die Nummer der DMX-Kanäle ein, die einer Gruppe zugeordnet werden sollen. Das Skript ermöglicht es einer Gruppe eine beliebige Anzahl der DMX-Kanäle zuzuordnen.

Beispiele der fertigen Deklarationen der Gruppen:

`wohnzimmer = dmxscene('wohnzimmer',2,3,4,5,6,7,8,9,10)` – Gruppe, die mit einem Zimmer verbunden ist.

`wohnzimmerdecke = dmxscene('decke',2,3,4)` – Szene, die mit einer bestimmten Lampe verbunden ist.

Das Skript ermöglicht es mehrere Szene, die mit denselben Satz der DMX Kanäle verbunden sind, zu speichern. Dazu dient die Funktion `save`. Zur Funktion `save` übergeben Sie als Argument der Name der Szene, die gespeichert werden soll. Die Syntax ist wie folgt: `objekt:save('name')`, z.B. `wohnzimmer:save('erholung')`.

Der Aufruf der Funktion verursacht, dass in der Szene `'erholung'` die Werte der DMX Kanäle gespeichert werden. In diesem Fall sind das die Nummer von 2 bis 10.

Die Szenen werden im nichtflüchtigen Speicher des **Base** Moduls (als MEM-Variablen) gespeichert. Jeder Aufruf der Funktion `save` überschreibt die Szene mit der aktuellen Werten der DMX Kanäle, die einer Gruppe zugeordnet werden. Die Werte aller Szenen können Sie im Lesezeichen **Zustand** anschauen. Jede Szene ist eine separate MEM-Variable mit dem gleichen Namen wie der Name der Gruppe und Szene. Der Wert einer bestimmten MEM Variable ist der Wert der bestimmten Kanäle, die im sedezimalen Code ausgedrückt wurden. Eine allgemeine Form der Darstellung ist wie folgt: `MEM.dmxs.<name>.<szene>=<wert>`.

Um die Szenen aufzurufen, verwenden Sie die Funktion `restore`. In der Funktion `restore` ist der Name der aufzurufenden Szene der Argument. Die Syntax ist wie folgt: `objekt:restore('name')`, z.B. `wohnzimmer:restore('erholung')`.

Wenn Sie die Gruppen deklariert haben, können Sie die Schnittstelle zur Steuerung der Szene erstellen.

Vergessen Sie nicht das Lesezeichen **Logik** zu speichern.

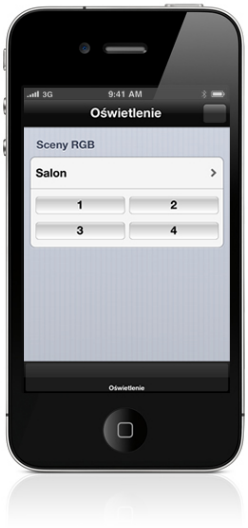
2. Remote

Die Schnittstelle in der Applikation **Remote** bestehen aus mindestens zwei Elemente: **RGB Licht**-Auswahl der Farbe und Helligkeit des Lichts und Steuerelement **Taste**- die Taste zum Speichern und Aufruf der Szene. Ein kurzes Drücken der Taste ruft die Szene auf- ein langes Drücken der Taste speichert die aktuellen Werte der DMX-Kanäle, die in der Deklaration der Gruppe bestimt wurden. Wenn für einen Satz der DMX-Kanäle mehrere Szenen deklariert wurden, dann fügen Sie die entsprechende Anzahl der **Tasten** hinzu.

Die unten beschriebene Prozedur gilt für die Deklaration der Gruppe `'wohnzimmer'`, die im letzten Kapitel dargestellt wurde.

1. Fügen Sie dann das neue Element **RGB Licht** und doppelklicken Sie auf ihm. Ergänzen Sie ihre Eigenschaften. In weiteren Feldern geben Sie die Nummer der DMX Kanäle, die jeder Farbe des Lichts zugewiesen sind, z.B. `DMX.2`, `DMX.3`, `DMX.4`. Zu unserer Gruppe gehören die Kanäle von 2 bis 10, deshalb wiederholen Sie diesen Schritt 3 Mal.
2. Fügen Sie **Taste** hinzu, doppelklicken Sie auf sie und ergänzen Sie seinen Eigenschaften:
   - In der Zelle **Etikett** geben Sie die Beschreibung der Taste ein.
   - Wählen Sie das Lesezeichen **Drücken** aus.
   - Klicken Sie auf **Kommando hinzufügen** und im angezeigten Fenster im Feld **Name** geben Sie: `C.LOGIC` ein, und im Feld **Wert**: `<name>:restore('<szene>')`, in diesem Fall `wohnzimmer:restore('erholung')`.
   - Gehen Sie zum Lesezeichen **Halten** und wiederholen Sie den letzten Schritt. Ändern Sie den Inhalt des Feldes **Wert** auf `<name>:save('<szene>')`, in diesem Fall `wohnzimmer:save(' erholung ')`.

Die Beispielschnittstelle wurde unten dargestellt:



3. Visualisierung

Aufruf der DMX-Szene kann einfach in der Visualisierung implementiert werden. Als Steuerelemente verwenden Sie z.B. **Taste**. Die Definition des Kontrollelementes ist nur auf Zuordnung der **Tasten** den Befehl und optional Etikett beschränkt.

Der Befehl soll die folgende Syntax haben: `LOGIC=objekt:restore('szene')`, in diesem Fall: `LOGIC=wohnzimmer:restore('erholung')`.

```
--
-- DMX Scenes
--
-- Copyright 2013 DOMIQ Sp. z o.o.
--
function dmxscene(group,...)
     assert(group, "Group name required")
     for _,v in ipairs(arg) do
          assert(v >= 0 and v <= 255, "Invalid slot value "..v)
     end
local t = {}
function t:save(name)
     local val = {}
     for _,v in ipairs(arg) do
          table.insert(val,
          string.format('%02x',
          math.ceil((get('DMX.'..v))*2.55)))
     end
     set(string.format('MEM.dmxs.%s.%s',group,name),
     table.concat(val))
end
function t:restore(name)
     local val = get(string.format('MEM.dmxs.%s.%s',group,name))
     assert(val,"Missing scene "..name)
     local c = 0
     for v in string.gmatch(val,"(%x%x)") do
          c = c + 1
          command('C.DMX.'..arg[c],
          math.floor(tonumber(v,16)/2.55+0,5))
     end
end
     return t
end
```