照明コントロールサーバ(helvar_server)マニュアル

2014/11/10 IANS3

概要

照明コントロールサーバ（以下 Helvar_Server)は、 外部からの情報・指令に応じて、Helvar社製 DALI router へ 照明制御コマンドを送信する、TCPベースのサーバソフトウェアです。

- UI表示機能(WebI/F): 照明の色を設定するUIを表示します。 スマートフォン(Nexus5　xxhdpi)の ブラウザ(Chrome)に最適化されています。
- センサ(可視光通信、バイタル)情報受信機能:　HTTP(POST)経由でセンサからの可視光IDを受け取り、照明情報を更新する　機能を持っています。（バイタルセンサ情報は現状は使っていません)

ディレクトリ構造

```
├── Helvar_server_spec.txt    サーバー仕様書たたき台
├── MANUAL.md                 本ファイル
├── README.md                 簡易説明(English)
├── helvar_server    [ ヘルバー照明サーバー本体 ]
|   ├── app.js                実装ファイル
|   ├── bin
|   |   └── www                 起動スクリプト
|   ├── handlers                エフェクトソース
|   |   ├── juliana.js          ジュリアナ
|   |   ├── knight2000.js       ナイトライダー
|   |   └── normal.js           デフォルト
|   ├── public                  コンテンツディレクトリ
|   |   ├── javascripts
|   |   ├── main_console.html   スマホUI定義
|   |   └── stylesheets
|   |       ├── BEEP1011.WAV    スタイル・素材など
|   |       ├── common.css
|   |       ├── cool.png
|   |       ├── hot.png
|   |       ├── leaf.jpg
|   |       ├── styles.css
|   |       └── techniqo.ttf
|   ├── routes                  WebI/F定義
|   |   └── index.js
|   ├── directlevel.js
|   ├── readsettings.js
|   ├── server.json
|   ├── timerhandler.js
|   └── views               [各ページのビュー情報1]
|       ├── error.jade
|       ├── index.jade
|       ├── layout.jade
|       ├── npm-debug.log
|       ├── pattern.jade
|       ├── postindex.jade
|       └── scene.jade
└── virtual_helvar          [仮想Helvarルータ]
    ├── public
    └── virtualhelvar.js        実装ファイル
```

./helvar_server

サーバのメインプログラムです。サーバは

- スマホUI　スマホとの通信
- 可視光ID、センサからのXML POSTによる通知の処理
- Helvar ルータへのDirectLevel（照明変更）通信 を行います。サーバのポートは 8000番です。

ツール

Helvar_serverでは以下のツールを使用しています。

- git リビジョン管理　> http://git-scm.com
- github ソースコード共有 > http://github.com
- node.js (v0.10.30で動作確認) > http://nodejs.org

取得

ソースコードはgithub上にあります(https://github.com/tkuro11/helvar_server)

Linux, MacOSX, Cygwin or MinGW(windows)等のUnix-like環境なら、コマンドラインから

```
% git clone https://github.com/tkuro11/helvar_server.git helvar_server
```

または https://github.com/tkuro11/helvar_server.git　の Download ZIP ボタンから。

起動方法

- サーバPCをインターネットに接続した状態で、 helvar_serverディレクトリに移り、コマンドラインから以下のコマンドを 実行して、必要なライブラリを取得します。

```
% npm install
```

- サーバの軌道は以下の通り

```
% ./bin/www
```

- サーバ停止は CTRL-cです。

スマホ UI アクセス方法

サーバを起動後、ブラウザからサーバポート8000につなぐことでアクセスできます。 例えば、照明コントロールであれば、スマートフォン（NEXUS5)のブラウザから

```
http://<サーバのIPアドレス>:8000/<API>
```

で接続できます。

APIは以下の３つがあります。

API	description
/	照明コントロール

API	description
/scene	シーン選択画面
/pattern	エフェクト（アニメーションパターン）選択画面

設定ファイルについて

サーバの各種設定は ./server.jsonに記述します。server.jsonを書き換えると、サーバは自動的に再実行 され、設定が反映されるようになっています（ただし、WebUI等はリロードしないと設定値は反映されない）。

key	description
router_address	address for Helvar router
default_color	color for empty position
MACDICT	MACaddress -> phone id table
ecoselector_colors	color table for eco-selector(in #hex format)
scene_colors	color table for scenes(in #hex format)

route_address: サーバが照明情報を送るHelvar社 DALI ルータのIPアドレスを指定します。

default_color: 可視光IDが検出されていない場所（デフォルト）の色を Web Colorで指定します。

例：

```
"default_color": "#f0f0ff"
```

MACDICT: 使用するスマートフォンのMACアドレスと端末番号(0〜)の組をJSON連想配列として登録する。 端末番号はサーバが内部的に使用するIDであり、重複がなければ任意ですが、なるべく連番であったほうが サーバ実行効率は良くなります。 ここに登録の無いMACアドレスのスマートフォンは、UIに接続しても照明コントロールはできません。 MACアドレスを文字列キー、端末番号を整数としてJSON辞書の形式で記述します。

例：

```
"MACDICT": {
    "f8:a9:d0:4e:fd:d7": 0,
    "f8:a9:d0:4f:00:73": 1,
    "10:aa:bb:ff:85:dd": 2,
    "dummy1": 3,
    "dummy2": 4,
    "dummy3": 5,
    "dummy4": 6,
    "dummy5": 7
}
```

ecoselector_colors: スマートフォン照明制御UIの固定カラーを指定します。 JSON配列として記述します。

例：

```
"ecoselector_colors": [
    "#f03030",
    "#f04050",
    "#c05050",
    "#a08080",
    "#8090a0",
    "#75c0c0",
    "#60e0c0",
    "#40f0f0"
]
```

scene_colors: シーンセレクタで使用する照明の色パターンを設定します。 シーン名を文字列キー　照明色パターンをWebColorの配列とした、JSON辞書として記述します。

```
"scene_colors": {
        "scene1": [
            "#1040ff", "#0e38ee", "#0c30dd", "#0a28cc",
            "#2040ff", "#1e38ee", "#1c30dd", "#1a28cc"
        ]
,       "scene2": [
            "#2040ff", "#1e38ee", "#1c30dd", "#1a28cc",
            "#1040ff", "#0e38ee", "#0c30dd", "#0a28cc"
        ]
```

エフェクト（アニメーションパターン）について

エフェクトは一定時間（現実装では1秒間隔) ごとに呼び出されるモジュール関数として実装され、 照明によるアニメーションエフェクトを実現しています。

格納場所

エフェクトの格納場所（パス）は ./handlers/ ディレクトリです。この下の .jsの拡張子を名前を持つ ファイル全てが読み込まれるようになっています。/patterns　で表示される名前はここから生成 されます。また、./handlers/ ディレクトリに新しいファイルを書き込むと、自動的にサーバが再起動し、エフェクトを読み直すようになっています。

特別なファイルとして、 normal.jsという名前のハンドラはデフォルトハンドラとして扱われます。 パターン（アニメーション）を起動しない場合、このnormal.jsに実装されたものが実行されます。

エフェクト実装

エフェクトの実装は以下のようにjavascriptオブジェクトとして実装します。

```
var <effectname> = {
   // エフェクト本体（一定時間間隔で自動的に実行される）
   next: function() {
            :
            :
       },
   // 初期化関数（エフェクトが切り替わった時に一度だけ実行される）
   init: function() {
            :
            :
       },

   // エフェクト内部で使用するローカル変数など
   colors: [0,0,0,0],
   dir: 1,
   idx: 1
};

module.exports = <effectname> //外部から利用可能とするおまじない
```

必須のメソッドは next(), init()で、この２つは予約されています。エフェクトが選択されると、 そのエフェクトオブジェクトのnext()が決められた時間間隔で呼び出されるので、各function()内 部でdirectlevelなどの関数を使い、照明レベルを変更するコードを記述します。

init()はエフェクトが切り替えられた時に最初に一度だけ呼び出されます。照明パターンの初期化などを 行います。

また、その他のメソッド、プロパティは自由で、例のようにエフェクト内部で利用するローカル変数や、 便利ルーチンなどを定義することが可能です。

エフェクトの時間感覚

エフェクトの時間感覚を指定しているのは./app.jsの中の以下の記述です

```
91 // TimerON
92 setInterval(timerhandler, 1000);
```

1000（ms単位）を変更することでエフェクト（アニメーション）の時間間隔を変更することができます。

DirectLevelライブラリ

エフェクト内で使用可能な照明をコントロールするユーティリティルーチンです。 特に ライブラリ読み込みの宣言は必要ありません（メインルーチンで読み込み済みのため）。

raw API

```
directlevel.raw(idx   , v,      <, fadetime>);
```

直接照明をコントロールします。この場合のidxはRGB独立の番号となります (例：照明1のRはidx = 1, 照明1のGが idx=2, 照明2のRが idx=4 、照明2のBが idx=6など)

rgb API

```
directlevel.rgb(RGBidx, [r,g,b], <, fadetime>);
```

各照明に対してRGBを一度に指定できます。RGBidxは照明番号と同じになります。

RGB APIを使うと、簡単に色を設定できますが、要不要にかかわらずかならず３要素を変更します。 DALIプロトコルは低速のため、すべてのRGB要素を書き換える必要がない場合は、raw APIを使ったほうが 良い場合があります。

[参考]XMLデータ形式について

XML POSTによって送られる可視光ID + センサーデータは以下の通りです。

```
<lvdata MACaddr="10:aa:bb:ff:85:bb"> <!--端末IDとしてMACアドレス使用-->
    <!-- 可視光通信で得られたID番号 -->
    <lightID>1</lightID>

    <!-- 可視光通信を取得した時間 JST(+9DST) -->
    <lightTimeStamp>1999-05-31T13:20:00.000+09:00</lightTimeStamp>

    <!-- 血中酸素飽和度 (%) -->
    <vitalSpO2>95.5</vitalSpO2>

    <!-- 体温(摂氏) -->
    <vitalTemp>36.6</vitalTemp>

    <!-- 心拍数(/分) -->
    <vitalPulse>60.3</vitalPulse>

    <!-- バイタル情報取得時間 JST(+9DST) -->
    <vitalTimeStamp>1999-05-31T13:20:00.000+09:00</vitalTimeStamp>
</lvdata>
```

仮装Helvarルータ

デバッグ用に使用する簡易シミュレータです。DirectLevelのみシミュレート可能です。

起動方法

```
% npm install
% node virtualhelvar
```

で動作開始します。終了は CTRL-cです。

仮想ルータ起動後、任意のホストからブラウザで

```
http://<サーバのIPアドレス>:3000/
```

で接続できます。

仮想ルータに接続する場合、 server.jsonの router_addressを仮想Helvarルータを起動しているPCに 設定してください。