Laumio Documentation

Version 1.0.0

HAUM

26 February 2017

Table des matières

Un Laumio est une lampe Ikea bidouillée par le HAUM et incluant 13 LEDs RGB. Celui-ci peut être programmé directement en Wifi via `socat`, l'interprêteur Python or en... regardant l'API.

Tout le code est disponible sur le dépôt Github dédié. Vous êtes libres de l'utilier, de le cloner, de le modifier, et de le publier.

Une version anglaise de la documentation est aussi disponible ici.

CHAPITRE 1

Conception du Laumio

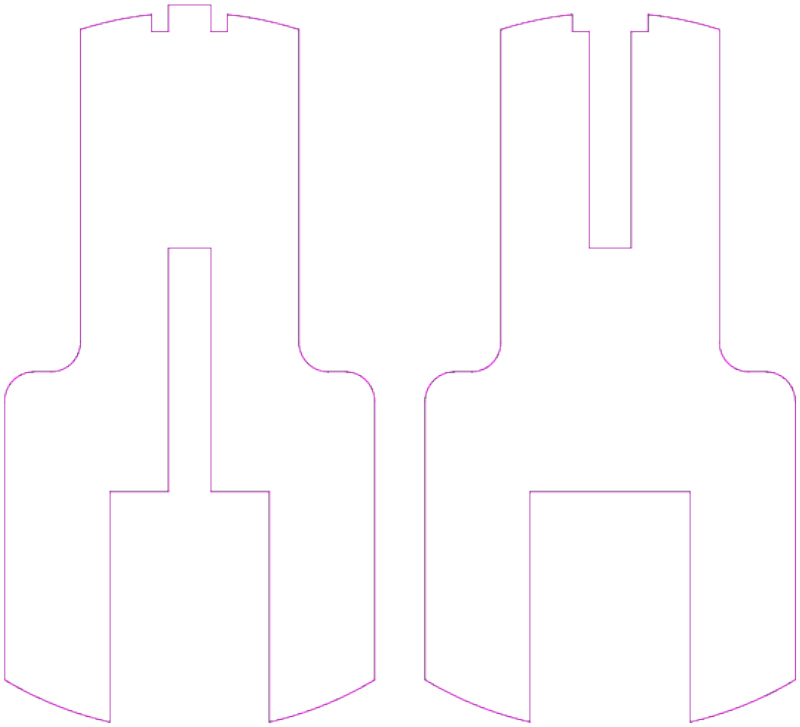
Le Laumio est constitué d'une lampe Ikea FADO et d'une structure "maison" contrôlant 13 LEDs RGB.

Cette structure physique est elle-même constituée d'une partie mécanique supportant les lampes et du système électronique associé.

Partie mécanique

Astuce : Tous les fichiers de conception peuvent être récupérés sur le dépôt Git.

La structure mécanique est composée d'un tronc central conçu pour se tenir sur la douille de l'ampoule. Celui-ci a été usiné grâce à une minifraiseuse et assemblé sans colle.



Deux propositions ont ensuite été portées pour accueillir les LEDs en elles-mêmes.

La première proposition est basée sur des bras pliables, les LEDs étant collées à leurs extrémités. Les problèmes qui en découlent sont principalement la fragilité de la conception (notamment au niveau des soudures) et l'importante manipulation nécessaire pour insérer cet arbre dans le verre.

La seconde proposition est toujours à l'étude. Celle-ci utiliserait des bandes de plastique souple reliées en haut et en bas de la structure.

Système électronique

Le système électronique est construit autour d'un ESP Wemos d1 Mini de de 13 LED WS2812. Afin de réutiliser le Wemos dans d'autres projets, il a été décidé de créer un shield adapter pour y connecter le microcontrôleur. Les fichiers de conception KiCAD design de ce shield peuvent être téléchargés ici.

Le code utilisé dans l'ESP est aussi présent sur le dépôt, merci de lire le fichier README pour être en mesure de compiler et de flasher le programme.

Une fois flashé et démarré, le Laumio va chercher à se connecter au réseau WiFi spécifié (voir `wifi-config.h`) et attendre des instructions UDP ou HTTP.

Démarrage

Au lancement, le Laumio clignote en violet puis démarre une animation de couleur rouge : il est en train de chercher le réseau WiFi spécifié et va tenter de s'y connecter.

Si la connexion est réussie, le Laumio lancera une animation Arc-en-ciel. Sinon, il s'illuminera en orage, signalant par-là que le mode *Access Point* est activé. Ce mode permettra à l'utilisateur de donner les informations nécessaires pour connecter l'appareil à un réseau WiFi valide.

Anatomie du Laumio's et and commandes en valeurs hexadécimales

Le laumio peut être commandé via une API UDP minimale utilisant des commandes simples en hexadécimal envoyées sur le **port 6969**. Celui-ci peut également recevoir via des commandes d'une API REST. Quatre types de sous-ensembles sont utilisables pour commander les LEDs du Laumio : le pixel (une seule led), l'anneau, la colonne et le Laumio dans son intégralité. Il est aussi possible de lancer quelques animations plus spécifiques.

Les LEDs sont disposées en 4 branches de trois LEDs, en plus d'une à son sommet. Celles-ci sont toutes numérotées de 0 à 12 depuis le bas d'une des branches vers sa cîme, évitant l'unique led du sommet, descendant ensuite de l'autre côté de la structure, et enfin remontant du bas d'une des deux branches restantes pour atteindre le bas de la dernière (la LED de sommet étant cette fois-ci reliée entre les deux branches).

Les anneaux correspondant quant à eux à une ligne horizontale de LEDs. Il y en a trois numérotées de 0 à 2, de bas en haut.

Enfin, dans le cas des colonnes, celles-ci se rapportent aux quatre branches de LEDs, bien entendu sans la LED du sommet.

API UDP minimale

Commande pixel par pixel

La commande est `0x00`. Les données à envoyer sont d'abord l'ID du pixel considéré et les trois octets de la couleur (RGB) choisie :

```
0x00 PixelID R G B
```

Commande anneau par anneau

La commande est `0x01`. Il faut d'abord spécifier l'ID de l'anneau (`0x00`, `0x01` or `0x02`) puis préciser les trois octets de couleur :

```
0x01 RingID R G B
```

Commande colonne par colonne

La commande est `0x02`. Ici, il faut indiquer l'ID de la colonne (`0x00`, `0x01`, `0x02` or `0x03`) puis les trois octets de couleur :

```
0x02 ColumnID R G B
```

Remplissage intégral

La commande est `0xff`, il suffit ensuite de specifier la couleur RGB :

```
0xff R G B
```

Animations

Les animations prédéfinies sont encore en développement mais deux d'entre elles ont déjà implémentées dans le Laumio.

Remplissage progressif

L'animation de Remplissage progressif ("color wipe") change passe le Laumio intégralement dans une seule couleur, mais LED après LED. Elle prend deux paramètres, que sont les trois octets de la nouvelle couleur et l'intervalle de temps entre de deux LEDs :

```
0x0b R G B Delay
```

Arc-en-ciel

L'animation Arc-en-ciel ("rainbow") ne prend pas de paramètres. En l'occurence, c'est celle utilisée par le Laumio quand celui-ci réussit à se connecter à un réseau WiFi :

```
0x0a
```

API JSON

L'API JSON est toujours en développement et toutes les requêtes UDP actuellement possibles n'ont pas encore été réimplémentées .

Statut

Le statut du Laumio peut être obtenu en utilisant un endpoint de l'`/api/` :

```
GET http://<laumio's ip>/api/
```

La réponse reçue et de la forme suivante :

```
{"name":"laumio","version":"devel"}
```

Commande pixel par pixel

Le Laumio est contrôlable vie de simples requêtes POST, précisant à la fois les clefs `led` et `rgb` :

```
{
  'led': PixelID,
  'rgb': [R, G, B]
}
```

Remplissage intégral

Pour commande toutes les LEDs du Laumio en une seule fois, il suffit de passer le paramètre `led` à la valeur 255 dans la bribe de JSON précédente :

```
{
  'led': 255,
  'rgb': [R, G, B]
}
```

CHAPITRE 3

Interface bash minimale

Le Laumio peut être commandé très simplement en utilisant socat ou n'importe quel autre outil réseau bien conçu.

Nous vous proposons le code suivant en guise d'exemple et de base de travail :

```
#!/bin/bash

# Consts
IP=              # Laumio's IP
ANIM_TIME=0.05
PAUSE_TIME=0.3

# Utils
fill() {
  echo -en "\xff\x$1\x$2\x$3"
}

led() {
  echo -en "\x00\x$1\x$2\x$3\x$4"
}

ring() {
  echo -en "\x01\x$1\x$2\x$3\x$4"
}

column() {
  echo -en "\x02\x$1\x$2\x$3\x$4"
}

# Program
(
fill 00 00 00
sleep $ANIM_TIME

while true; do

  # write your animation here...

done
) | socat - udp-sendto:$IP:6969
```

En outre, il est également possible d'envoyer plusieurs commandes en une seule requête, comme dans l'exemple suivant :

Ici, le Laumio est d'abord intégralement passé en rouge, puis la LED du sommet est passée en blanc.

CHAPITRE 4

Librairie Python

La classe `Laumio` est construite comme une encapsulation des commandes minimales présentées dans la page de l'API.

class Laumio

__init__(*ip*)
Constructeur pour contrôler un Laumio répondant à l'adresse IP *ip*.

wipeOut()
Eteint toutes les LEDs.
En réalité juste un alias pour :py :method :'fillColor'.

fillColor(*r*, *g*, *b*)
Remplit toutes les LEDs du Laumio avec la couleur (*r*, *g*, *b*).

fillRing(*ringid*, *r*, *g*, *b*)
Remplit un anneau d'une seule couleur. *ringid* est dans l'intervalle 0~3 et *r*, *g*, *b* précisent la couleur. La LED du sommet n'est pas commandable ici.

fillColumn(*colmunid*, *r*, *g*, *b*)
Remplit une colonne d'une seule couleur. *columnid* est dans l'intervalle 0~4 et *r*, *g*, *b* précisentla couleur. La LED du sommet n'est pas incluse dans les colonnes.

setPixelColor(*pixel*, *r*, *g*, *b*)
Passe un pixel *pixel* (0~12) à la couleur (*r*, *g*, *b*) demandée.

colorWipe(*r*, *g*, *b*, *delay*)
Lance une animation de remplissage progressif avec la color (*r*, *g*, *b*) et un intervalle de temps *delay* entre deux LEDs.

rainbow()
Lance une animation arc-en-ciel.

status()
Return the JSON **string** for the Laumio's status

_send(*payload*)
Conventionnaly private, this method is used to transmit a raw `bytearray` payload to a Laumio. It can be used to trigger custom animations.

Symbols

C

F

L

R

S

W