Système de notification physique

Boule lumineuse qui s'éclaire en fonction des notifications reçues.

Difficulté **Moyen** | Durée **4 heure(s)** | Catégories **Décoration, Électronique, Mobilier, Maison**

Coût **30 EUR (€)**

Sommaire

Introduction

Je ne sais pas vous mais parfois entendre mon téléphone vibrer toutes les 30 secondes a tendance à m'énerver. Cependant, je n'aime pas non plus le mettre en silencieux de peur de rater un message important. C'est comme ça que j'ai eu l'idée de fabriquer un système de notification "physique".
Mais utiliser une simple LED s'allumant ne m'a jamais paru très intéressant. J'ai donc cherché pendant un moment un objet dans lequel installer un tel système. L'occasion s'est finalement présentée quand des amis m'ont offert une lampe de chevet alimentée en USB en forme de "*death star*". **#starWarsFan**

Matériaux

- un objet dans lequel installer votre système (pour moi une "*death star*")
- un ESP8266 (ESP-01)
- des leds de type WS2812
- un régulateur 3,3V , LM1117 (ou équivalent)
- deux condensateurs de 10 µF
- un condensateur de 100 µF
- un bornier
- des connecteurs males/femelles pour circuit imprimé
- une plaque d'essai
- soudure
- fil électrique

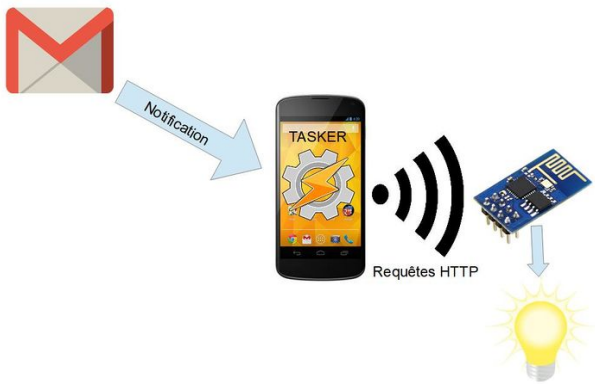
Outils

- fer à souder
- tournevis
- pinces
- convertisseur usb série
- breadboard

Étape 1 - Principe de fonctionnement

Le fonctionnement du système est basé sur l'utilisation de l'application *Tasker* installée sur le téléphone et d'un *ESP8266* connecté au même réseau wifi que le téléphone.
D'un coté l'ESP8266 est programmé pour allumer ou d'animer les leds d'une façon particulière en fonction des requêtes HTTP qui lui sont envoyées sur le port 80.
De l'autre côté Tascker est paramétré pour envoyer une requête http particulière à l'adresse IP de l'ESP8266 à chaque fois qu'une application notifie d'un événement.

> Par exemple si on reçoit un mail, gmail fait une notification, tasker envois alors la requêtes "http://192.168.0.109/gpio/1" et l'EPS allument les leds en rouge.



Étape 2 - Programmation de l'ESP8266

Pour programmer l'ESP82266 on va uiliser l'IDE Arduino. Pour cela, il vous faut tout d'abord ajouter le support de ce type de carte dans l'IDE en ajoutant l'URL suivante dans Fichier>Préférences>URL de gestionnaire de cartes supplémentaires :

```
http://arduino.esp8266.com/package_esp8266com_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

Puis choisir "Generic ESP8266 Module " dans Outils>types de carte.
Et enfin relier l'ESP au convertisseur USB/série comme indiquer sur le schéma.
Pour gérer les leds WS2812 la meilleur solution est la librairie NeoPixel d'Adafruit:

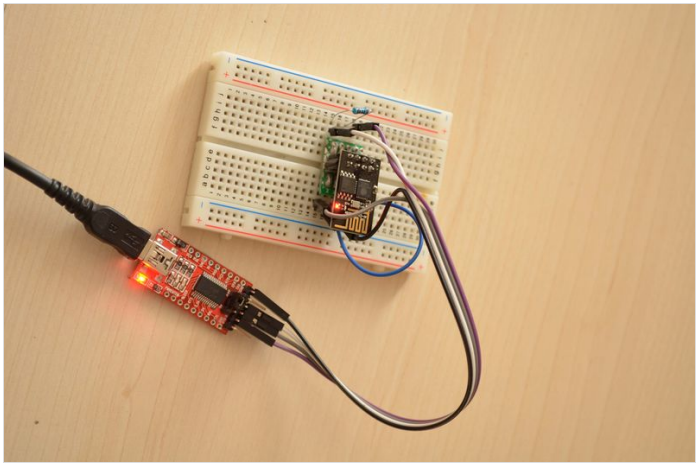
```
https://github.com/adafruit/Adafruit_NeoPixel
```

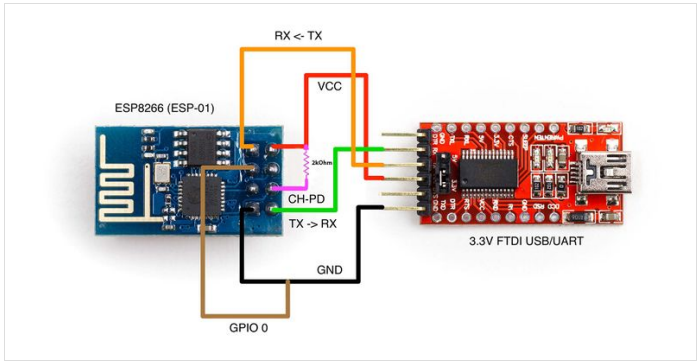
Maintenant que vous avez tous les outils d'installés vous allez pouvoir passez envoyer le programme sur l'ESP:

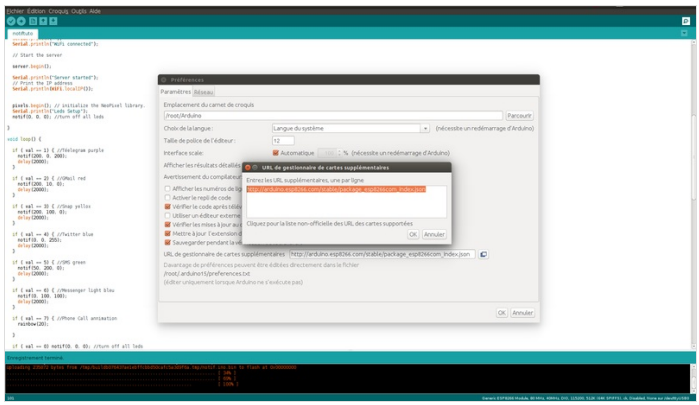
```
https://drive.google.com/open?id=0B8tCTkPLfNNrQTdaYTVYZWQwTGc
```

Vous pouvez ensuite modifier/ajouter des valeurs au début du "void loop" pour choisir la couleur de notification associé à chaque application. La couleur est gérée par la fonction *notif* que prend comme paramètres 3 valeurs comprises entre 0 et 255 correspondant respectivement au taux de rouge, de vert et de bleu.

> Penser à modifier l'attributs NUMLED en fontion du nombre de leds que vous utilisez et à changer le ssid et le password pour vous connecter à votre réseau wifi.







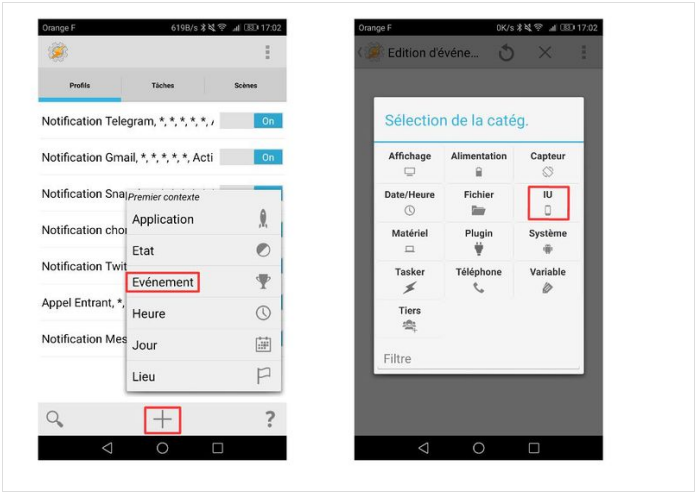
Étape 3 - Préparation du téléphone

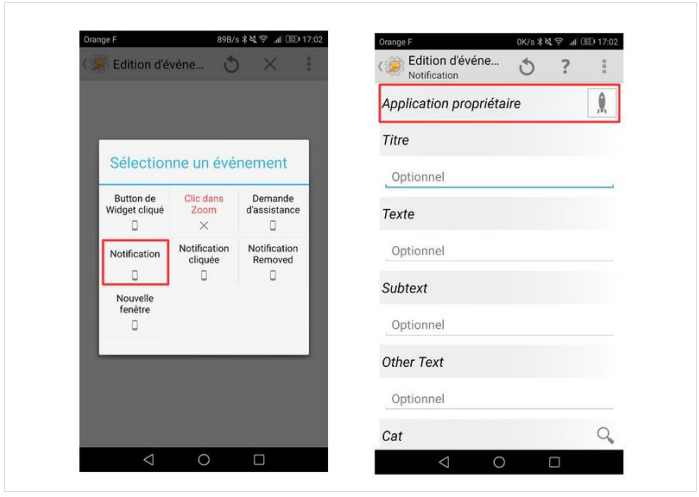
Pour gérer de manière automatique et transparente l'envoi de requêtes à l'ESP8266 nous allons utiliser l'application **Tasker**:

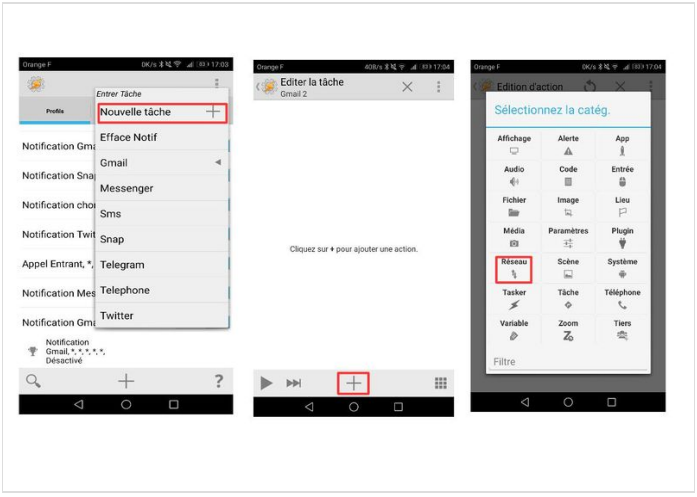
https://play.google.com/store/apps/details?id=net.dinglisch.android.taskerm&hl=fr

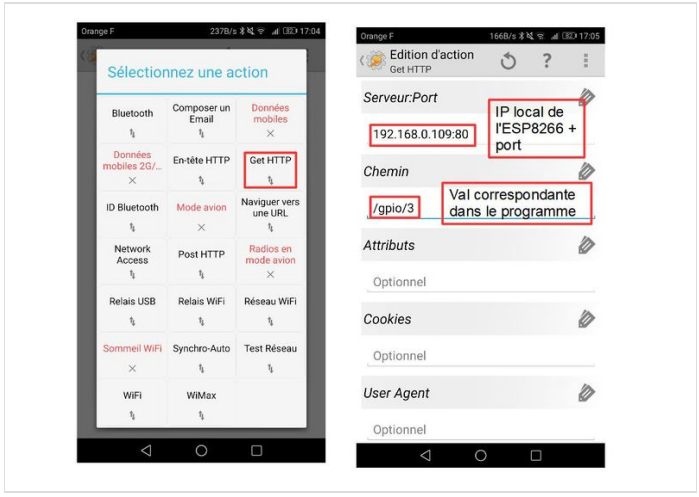
Tasker est une application payante disponible sur le google play store pour quelque euros. Cependant il existe une version d'essai de 2 semaines

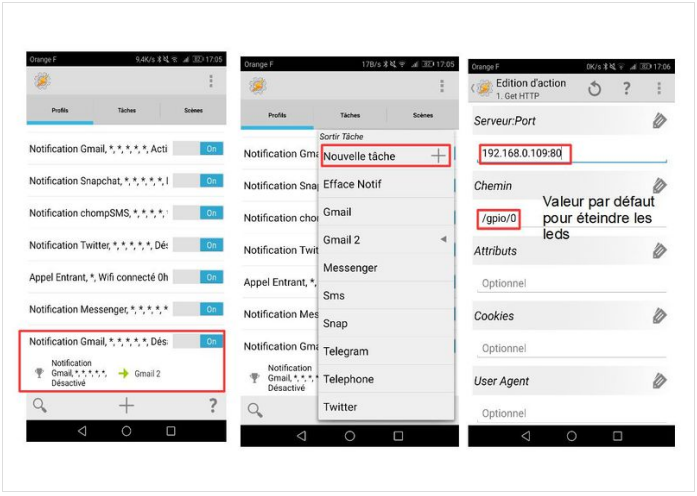
Maintenant, vous pouvez paramétrer Tasker (CF capture d'écran) en ajoutant un *contexte* de notification entraînant une *tâche* qui envoie une requête *http* sur l'IP local de l'ESP8266 sur le chemin "/gpio/NumAssociténAL'application" pour allumer les leds puis une seconde qui envoie "/gpio/0" pour les éteindre. Il peut aussi être intéressant de rajouter un *contexte* "wifi" pour que les requêtes aient lieu uniquement lorsque vous êtes connecté au même réseau que l'ESP8266.





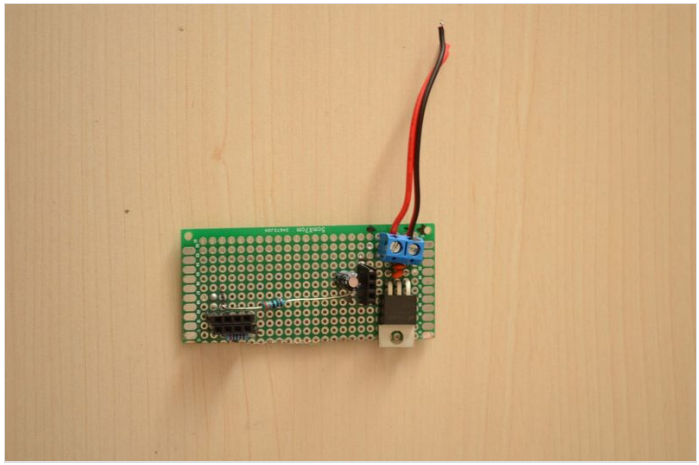


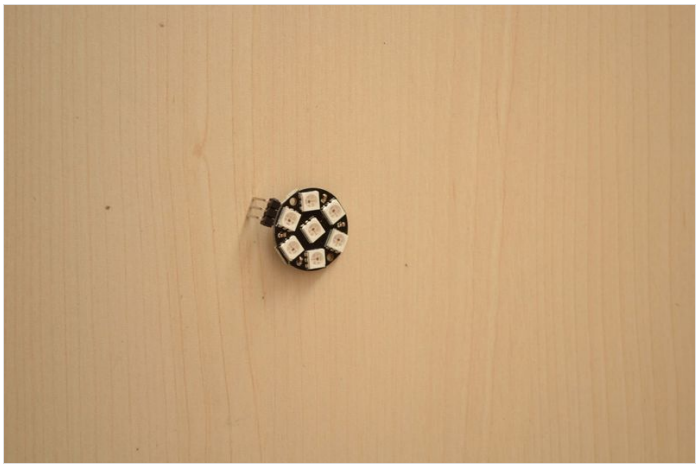


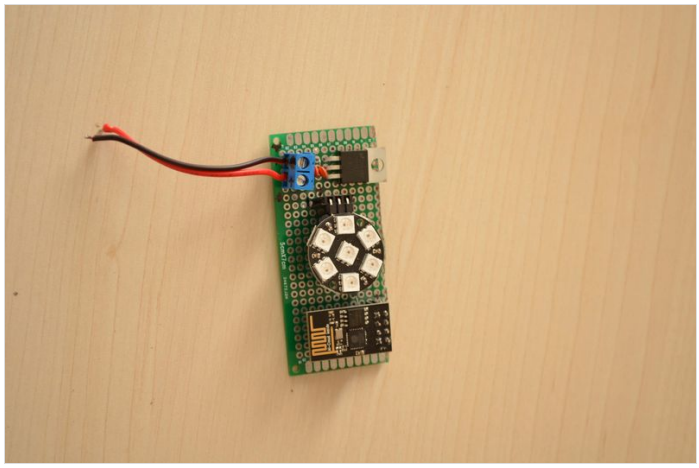


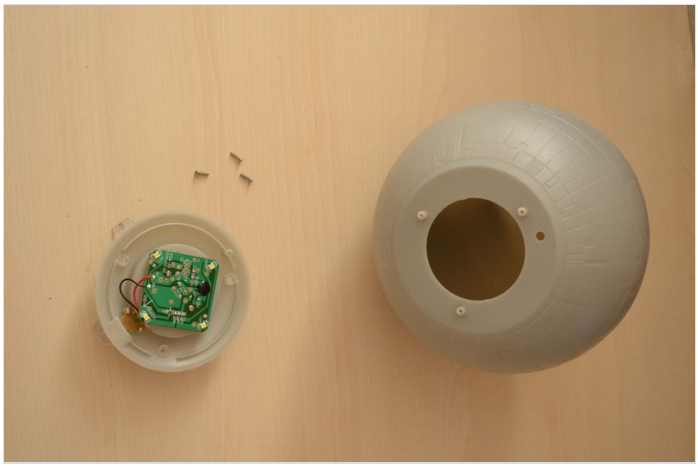
Étape 4 - Installation de l'électronique

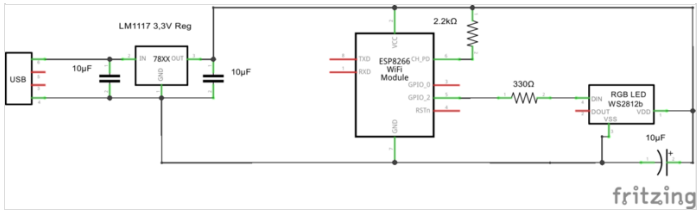
Une fois toute la partie logiciel terminée vous pouvez passer à l'électronique.
On retrouve rien de bien compliqué. L'électricité est fournie par le port USB. La tension est ensuite abaissée de 5V à 3,3V pour alimenter l'ESP8266. Ce dernier contrôle ensuite les LEDs avec sa broche 2 par l'intermédiaire d'une résistance de 330 Ohms. Enfin il est aussi intéressant d'ajouter un condensateur de 100 µF aux bornes des WS2812 pour *lisser* la tension d'alimentation.







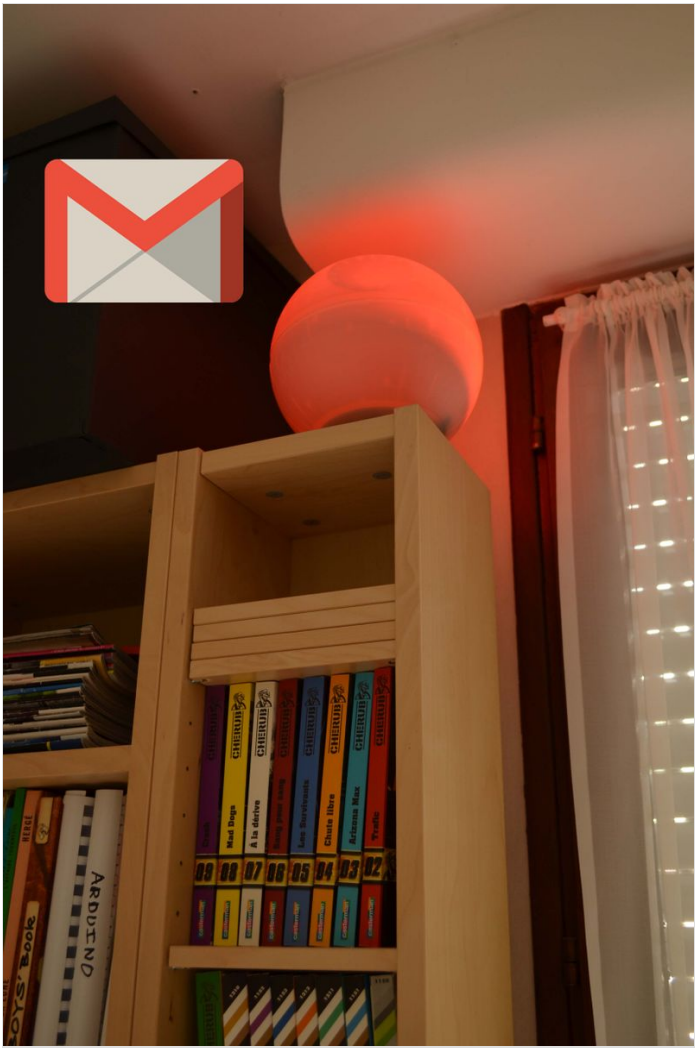




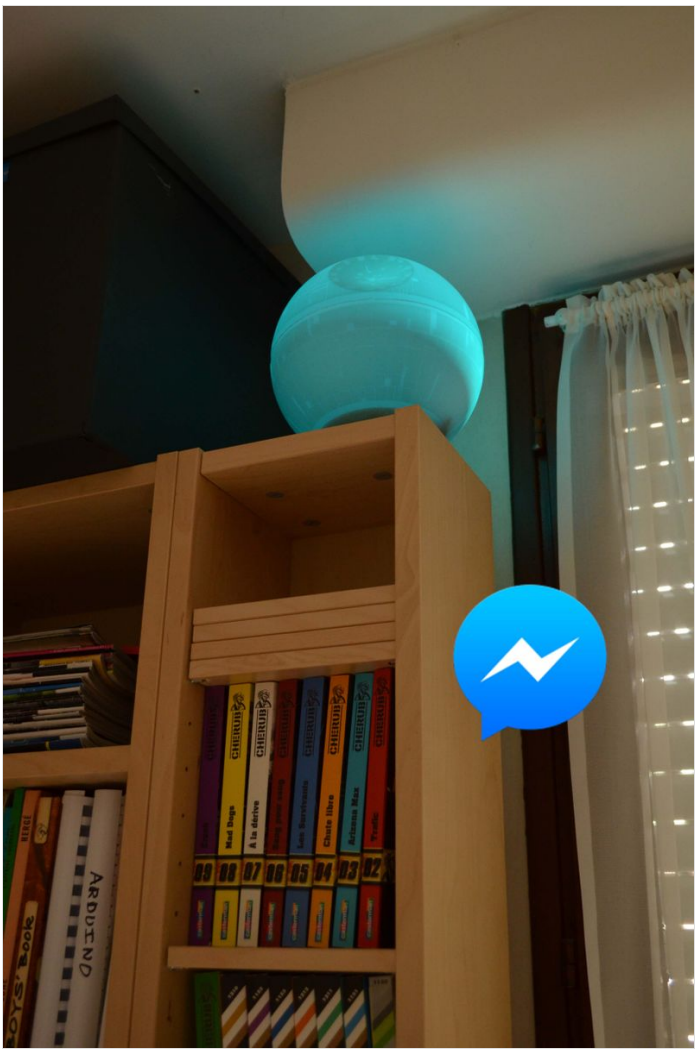
Étape 5 - Conclusion

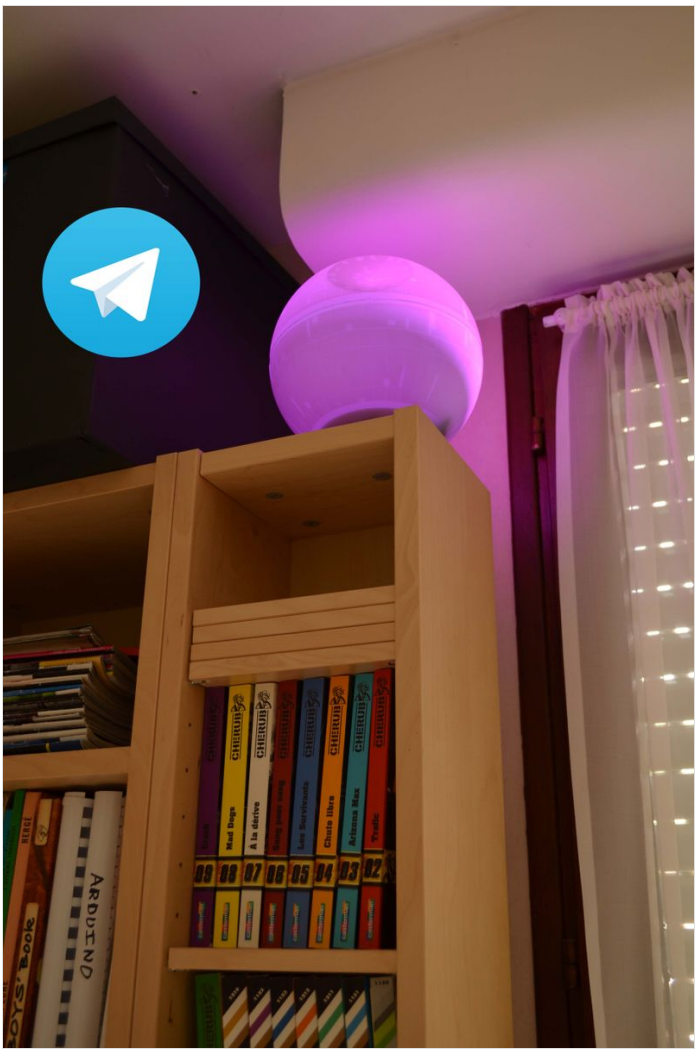
Ce tutoriel arrive à son terme, j'espère avoir été clair dans mes explications. Cependant, si jamais vous ne comprenez pas un point n'hésitez pas à engager la conversation. ☺

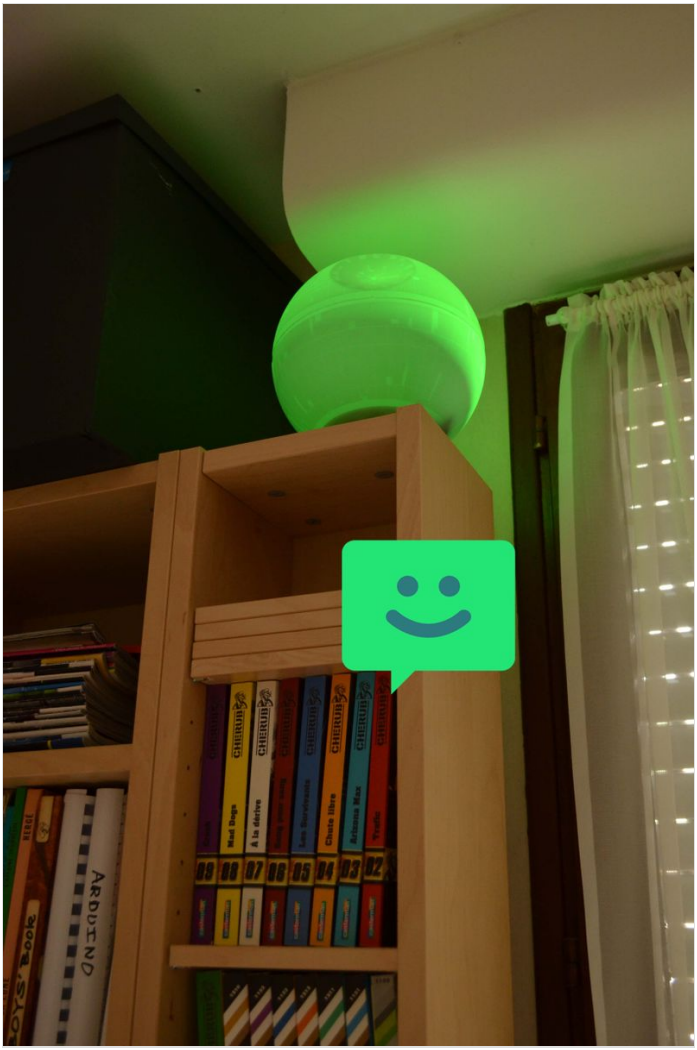
> Si vous intégrer un système de notif similaire dans un autre objet partagez le sur cette page, ça pourrait être cool pour donner des idées aux autres ☺


ARDUINO


ARDUINO







Notes et références

Pour les led WS8266: https://learn.sparkfun.com/tutorials/ws2812-breakout-hookup-guide
Pour l'ESP8266: Hackable magazine numéro 7
Pour tasker: http://tasker.dinglisch.net/
Je tiens également remercier *Team Biz'* pour m'avoir offert la *death star* qui a permis à ce projet de naître 😘 #INSA