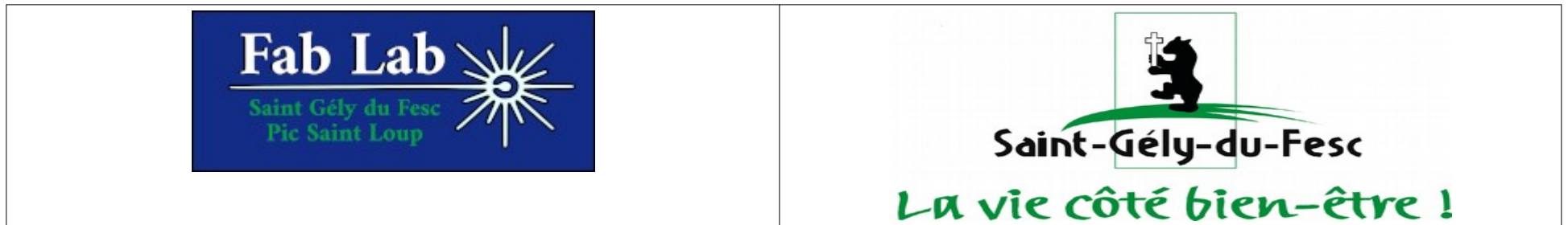


PRINCIPES ET APPLICATIONS DES LEDS ADRESSABLES MULTI-COULEURS.

François Fabre & Maxime Bigot pour le FabLab de Saint Gély du Fesc – Pic Saint Loup.

Le 7 juillet 2022

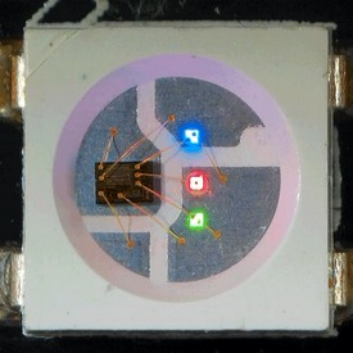

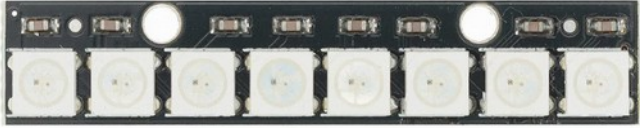

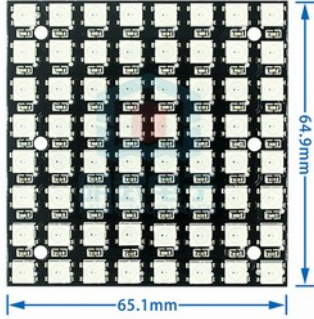



Il existe différents types de led adressables ici nous allons utiliser des [ws2812b](#) ce n'est pas le modèle le plus performant, mais c'est un des modèles le moins cher.

Si vous souhaitez de la performance, je vous conseille de regarder du côté des [apa102](#).

- **FONCTIONNEMENT DES WS2812B**
- **UTILISATION D'UNE BANDE RIGIDE DE 8 LEDS POUR ÉCLAIRER UNE PLAQUE DE PMMA.**
- **UTILISATION D'UN ANNEAU CIRCULAIRE POUR AFFICHER LA VALEUR D'UN POTENTIOMÈTRE.**
- **PILOTAGE PAR WLED DE 144 LEDS.**
- **AFFICHAGE D'UN TEXTE BASÉ SUR LA PERSISTENCE RÉTINIENNE.**

On trouve les ws2812b sur fond blanc ou noir, sous différents formats, à l'unité, en bande rigide, en dalle, en bande souple, en anneaux circulaires...

À l'unité en CMS	À l'unité à souder	Bande rigide
		
Anneau circulaire	Dalle	Bande souple
		

FONCTIONNEMENT DES WS2812B

Les WS2812B sont des composants en boîtier 5050 (5mmx5mm), elles intègrent 3 LEDS mais aussi un circuit intégré logique destiné à piloter les LEDS, ce circuit intégré gère chacune des LEDS via une PWM, les bits sont transmis à une fréquence de 800 kHz avec une résolution de 8 bits, soit 256 niveaux par couleur, pour un total de 16 millions de couleur possibles, le circuit de pilotage gère les 3 leds mais également la communication.

En effet, ces LEDS sont autonomes et disposent de deux broches de communication permettant de leur envoyer une valeur de PWM pour chacune des 3 couleurs, soit 24 bits de données.

Tension d'entrée: DC 5 V

Une Puce Led consomme environ 0.3 W

Exemple : DC 5 V, Si 45 W 5 V 9 A

Puissances au mètre des différents modèles disponibles:

30 LEDS/m 9 W / m

60 LEDS/m 18 W / m

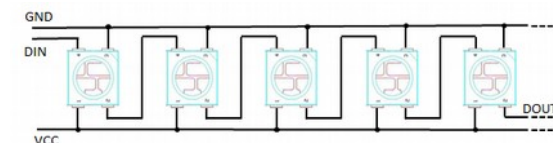
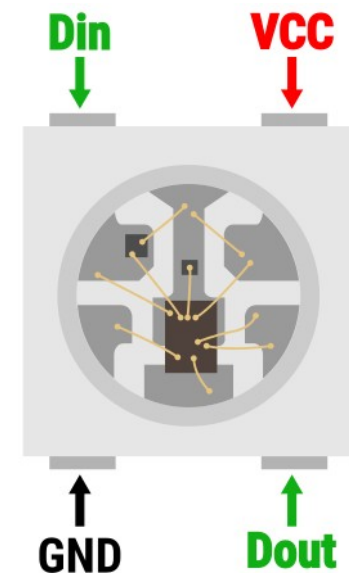
74 LEDS/m 22.2 W / m

96 LEDS/m 28.8 W / m

144 LEDS/m 43.2 W / m

La première broche de communication est l'entrée DIN et la seconde DOUT.

Les LED sont conçues pour être chaînées, le DOUT de la LED amont est connecté au DIN de la LED aval.



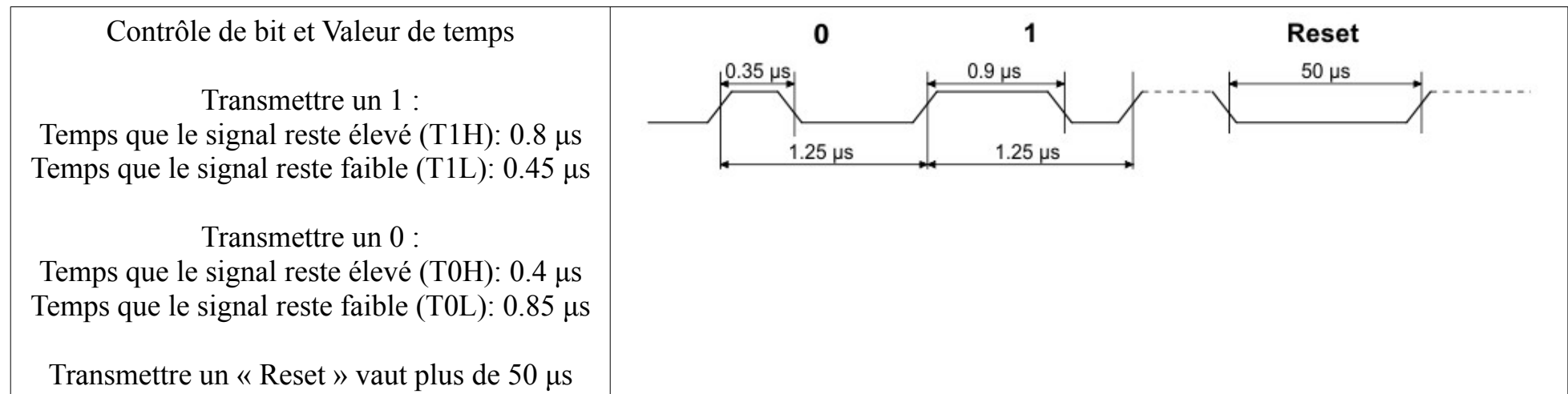
Protocole

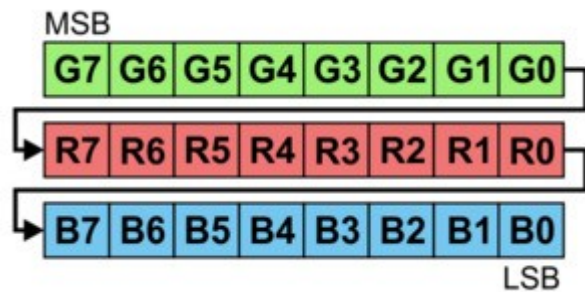
À l'état de repos, c'est à dire si aucune information ne doit être envoyée à la LED, le signal DIN est maintenu à l'état bas.

Le passage de DIN à l'état haut indique à la LED le début d'une transmission d'une série de bits.

Les bits sont transmis à une fréquence de 800 kHz, soit 1,25 μ s par bit.

Il n'y a pas de signal d'horloge séparé, la LED se base sur le signal transportant les données pour se synchroniser.





24 bits représentent la couleur d'une LED SK6812/WS281x dans une bande de LEDS RGB adressables.

24 bits demandent un temps de transmission de $1,25 \mu\text{s} \times 24$, soit $30 \mu\text{s}$.

L'envoi des données à une chaîne de 100 LEDS demande 3 ms.

Chaque LED lit les 24 premiers bits qui lui sont transmis puis transmet le reste de la trame aux autres LEDS sans les 24 bits qu'elle vient de lire.

Le micro-contrôleur n'est connecté qu'à la première LED d'une chaîne.

Le mode opératoire consiste à envoyer à cette première LED autant de séries de 24 bits qu'il y a de LEDS dans la chaîne.

Chaque LED prend pour elle les 24 premiers bits qu'elle reçoit et transmet les suivants à la LED suivante de la chaîne.

De fil en aiguille chaque LED reçoit les 24 bits qui lui sont destinés.

Après la transmission, DIN est maintenu à l'état bas pendant $50 \mu\text{s}$ afin de réinitialiser le mécanisme de transmission de l'information de LED en LED.

De cette manière, à la prochaine transmission, l'échantillonnage des 24 premiers bits reçus sera de nouveau effectué par chaque LED de la chaîne.

Datashet ws2812b: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>

les bibliothèques les plus courantes sont (**ATTENTION** désactivent les interruptions):

[FastLED](#)

et

[Adafruit_NeoPixel](#)

Un bout de code basique pour tester l'illumination de 2 LEDS ws2812b avec la librairie [FastLED](#).

Sur la diapo suivante nous observerons le résultat à l'aide d'un oscilloscope.



```
ws2812b | Arduino 1.8.19
Fichier Édition Croquis Outils Aide
ws2812b
1 #include <M5StickCPlus.h>
2 #include <FastLED.h>
3
4 #define LED_PIN G26
5 #define NUM_LEDS 2
6 #define FRAMES_PER_SECOND 2500 // Default 400 Max 2880
7 CRGB leds[NUM_LEDS];
8
9 void setup() {
10   Serial.begin(115200);
11   delay(500);
12   M5.begin(true, true);
13   FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, NUM_LEDS);
14   FastLED.setMaxRefreshRate(FRAMES_PER_SECOND);
15   FastLED.setBrightness(255);
16 } // End setup
17
18 void loop() {
19   leds[0] = CRGB (0, 0, 0); // GRB
20   leds[1] = CRGB (255, 255, 255); // GRB
21   FastLED.show();
22 } //Fin loop
...
1 M5Stick-C-Plus, Default, 240MHz (WiFi/BT), 1500000, None sur /dev/ttyUSB0
```

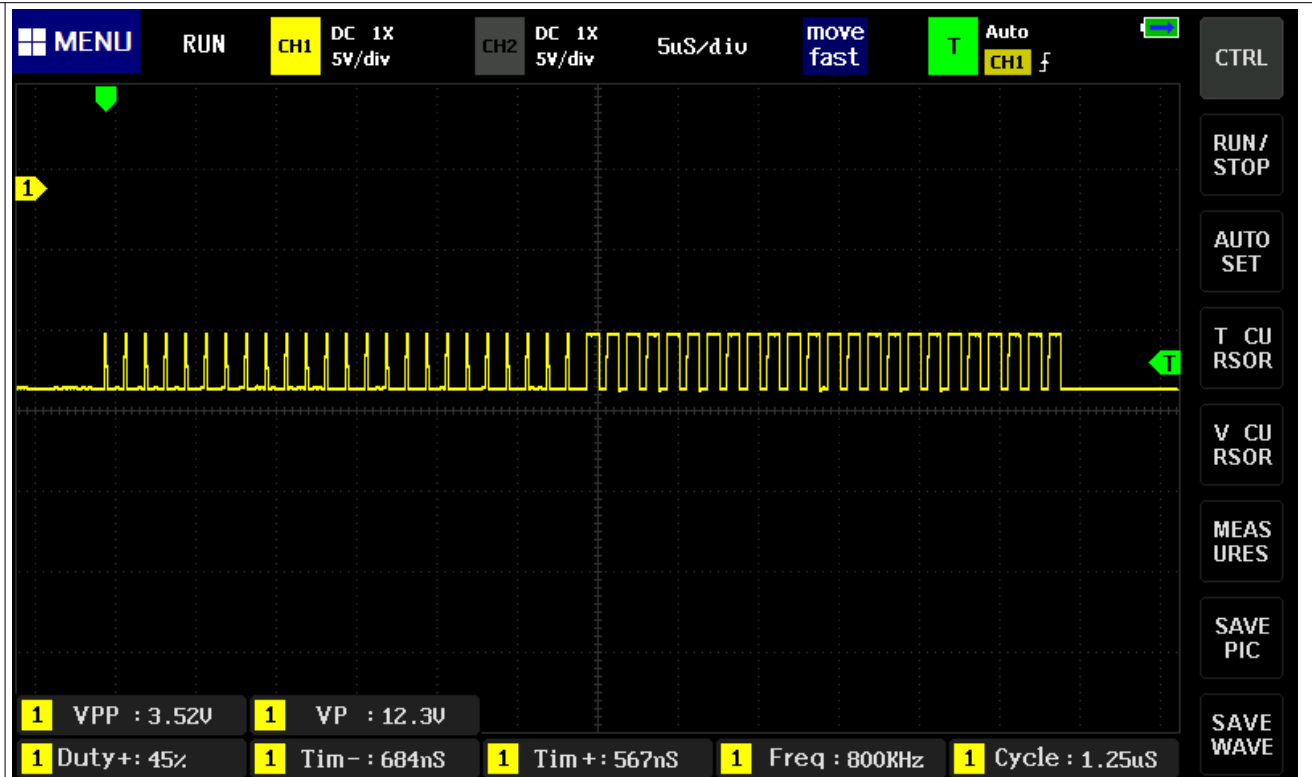
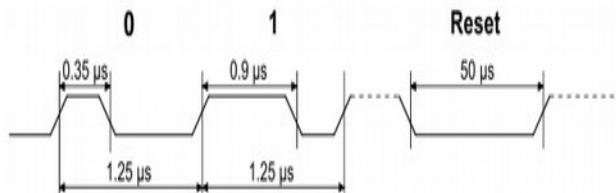
Le résultat du code ci-dessus observé sur mon oscilloscope (DANIU ADS1013D). Il semble avoir disparu mais on le trouve encore sur Aliexpress), apparemment remplacé par cette référence : [FNIRSI-1013D](#)

La première LED est : GRB à zéro, les premiers 24 Bits (3*8)
 0 (Binaire)=00000000 (Décimal 8 bits)

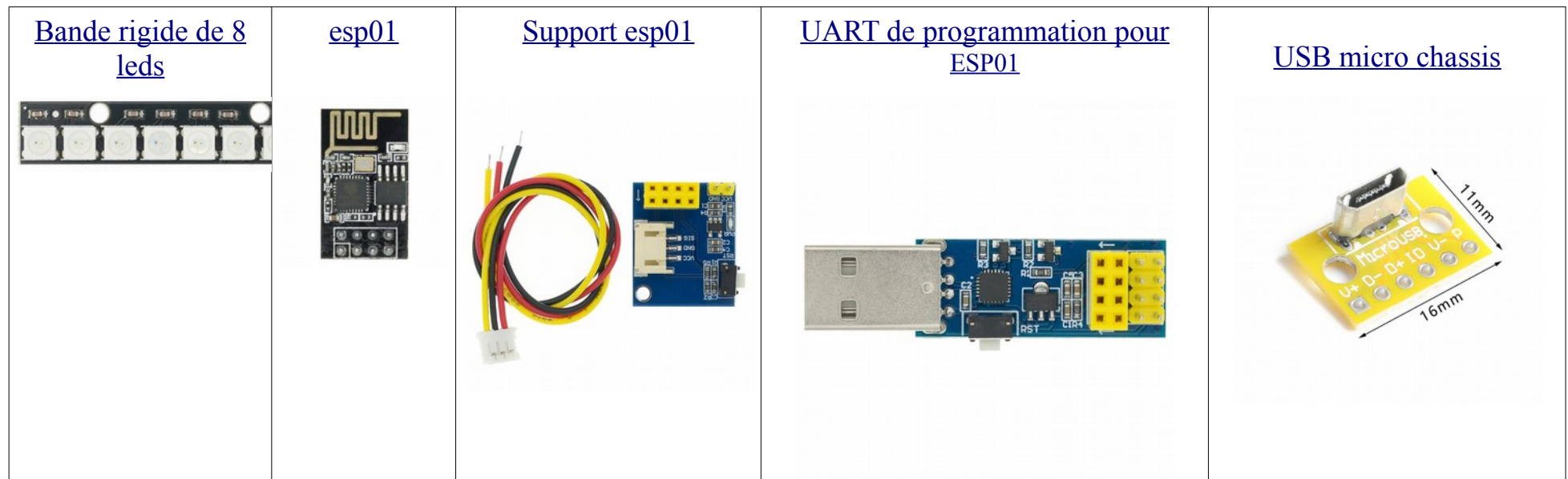
La seconde LED est : GRB à 255, la seconde succession de 24 Bits (3*8)
 255 (Binaire)=11111111 (Décimal 8 bits)

On voit les 24 (3*8) premiers bits à 0
 et les 24 bits (3*8) suivants à 1

Suivant la règle pour 0 et pour 1



UTILISATION D'UNE BANDE RIGIDE DE 8 LEDS POUR ÉCLAIRER UNE PLAQUE DE PMMA.



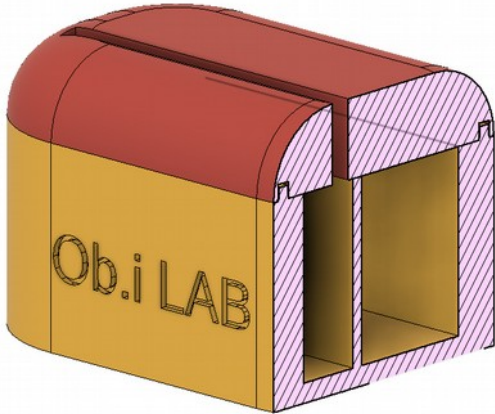
Par mesure d'économie nous allons utiliser un esp01, la totalité des composants nécessaires ne dépasse pas les 5 euro.

Seul le WiFi est disponible sur les esp01, l'interface de pilotage sera donc en WiFi par point d'accès.

Le boîtier sera réalisé en impression 3D et le PMMA découpé et gravé à la découpe/graveuse laser Co2.



Vue 3D avec Fusion360



Interface de pilotage
SSID : Ob.i LAB_esp8266
PASS : 0123456789

🏠 ▲ 192.168.4.1

Programme

Rouge

Vert

Bleu

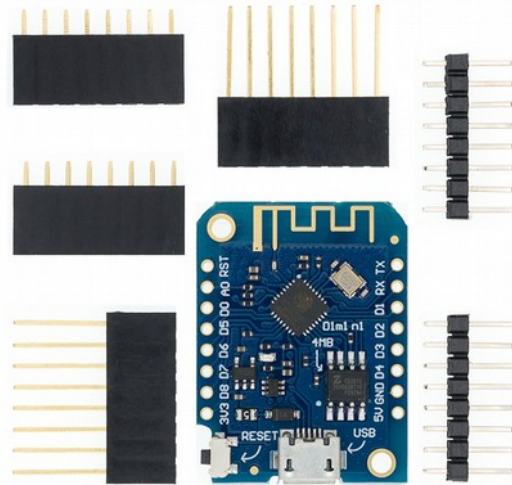
Color Picker

UTILISATION D'UN ANNEAU CIRCULAIRE POUR AFFICHER LA VALEUR D'UN POTENTIOMÈTRE.

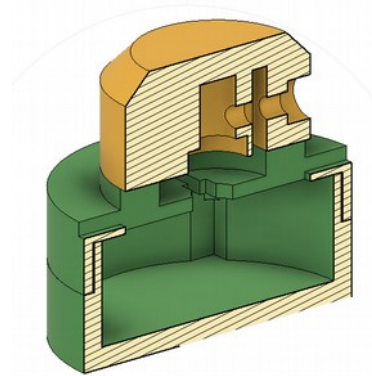
Anneau circulaire de 16 leds



WeMos D1 Mini PRO V3.0.0



Vue 3D avec Fusion360



Potentiomètre 10 K Ω



Visualisation de la rotation d'un potentiomètre



PILOTAGE PAR WLED DE 144 LEDS.



Installation par navigateur web: <https://install.wled.me/>

Github de WLED : <https://github.com/Aircoookie/WLED>

Documentation d'installation :

<https://www.instructables.com/Flash-WLED-to-ESP8266-Based-Controllers/>

Github de Wled :

<https://github.com/Aircoookie/WLED/releases>

ESPHome-Flasher :

<https://github.com/esphome/esphome-flasher/releases>

ESP_Easy_Flasher :

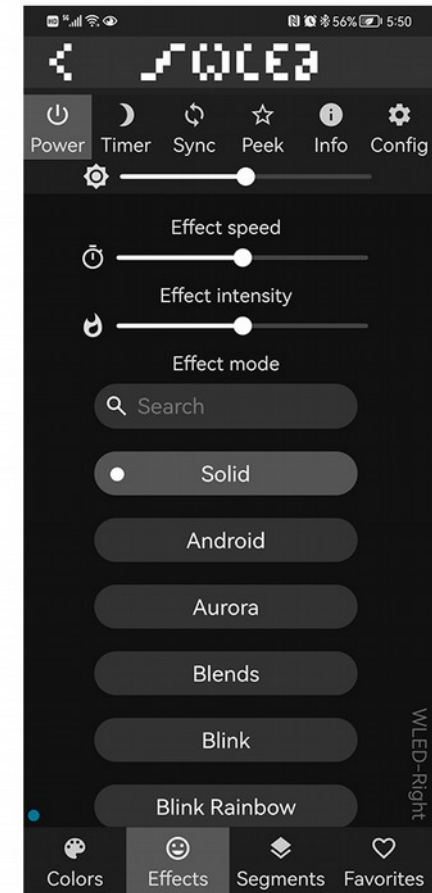
https://github.com/Grovkillen/ESP_Easy_Flasher/releases

Flash Download Tools :

<https://www.espressif.com/en/support/download/other-tools>

Tasmotizer : <https://github.com/tasmota/tasmotizer/releases/>

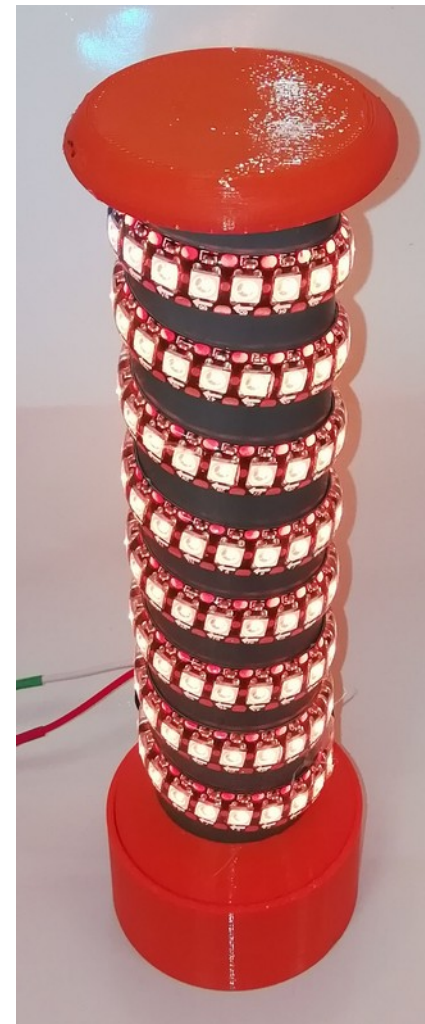
Installez Wled par la méthode de votre choix
Recherchez le point d'accès WiFi nommé : WLED-AP
Mot de passe de connexion : wled1234



Vue 3D avec Fusion360



Bande souple de 144 LEDS pilotées par WLED.



AFFICHAGE D'UN TEXTE BASÉ SUR LA PERSISTENCE RÉTINIENNE.

Exemple d'utilisation d'une bande rigide de 8 leds pour afficher une lettre par persistance rétinienne ou POV en Anglais (persistence of vision).

Nous allons générer l'apparition d'une image ou de texte avec seulement une rampe en ligne de 8 LEDS !

Pour afficher du texte on va créer une matrice par exemple de 8 * 5 pixels

Exemple pour un « B » en tableau de 8 * 5

Si je retranscrit l'image dans un tableau pour Arduino cela donne pour un B :

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0}, // B
```

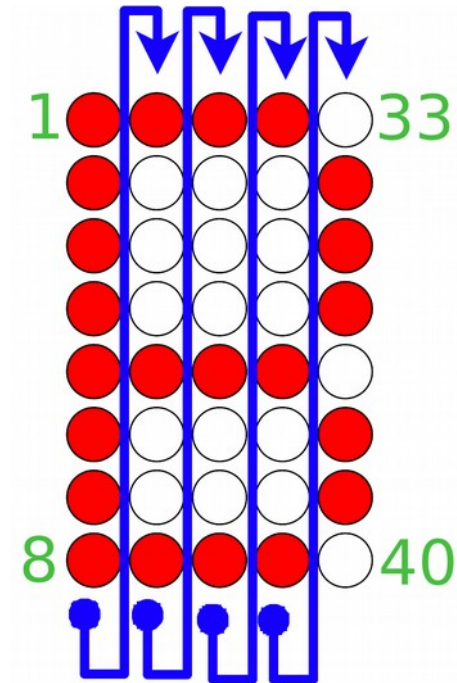


Tableau à 2 dimensions (40 lignes et 40 colonnes) pour chiffres, lettres et quelques éléments de ponctuation adaptés à l'alphabet français.

```
const String charsdispo = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ. :-";
```

```
const int Tab2DFULL[40][40] = {
{0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0}, // 0
{0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // 1
{1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1}, // 2
{1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, // 3
{0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1}, // 4
{1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1}, // 5
{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1}, // 6
{1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0}, // 7
{0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0}, // 8
{1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, // 9
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1}, // A
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0}, // B
{0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0}, // C
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0}, // D
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0}, // E
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0}, // F
{0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0}, // G
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1}, // H
{1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0}, // I
{0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0}, // J
{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0}, // K
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0}, // L
{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1}, // M
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1}, // N
{0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0}, // O
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0}, // P
{0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1}, // Q
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0}, // R
{0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0}, // S
{1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0}, // T
{1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0}, // U
{1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0}, // V
{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1}, // W
{1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1}, // X
{1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0}, // Y
{1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1}, // Z
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // Point
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // Espace
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // :
{0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // -
};
```

L'objectif est d'observer un texte sur un disque virtuel de 220 mm de diamètre basé sur la persistance rétinienne.

Sur un disque virtuel modélisé avec Fusion 360, j'arrive à placer 100 colonnes de 8 LEDS sans que cela ne soit pas trop serré.

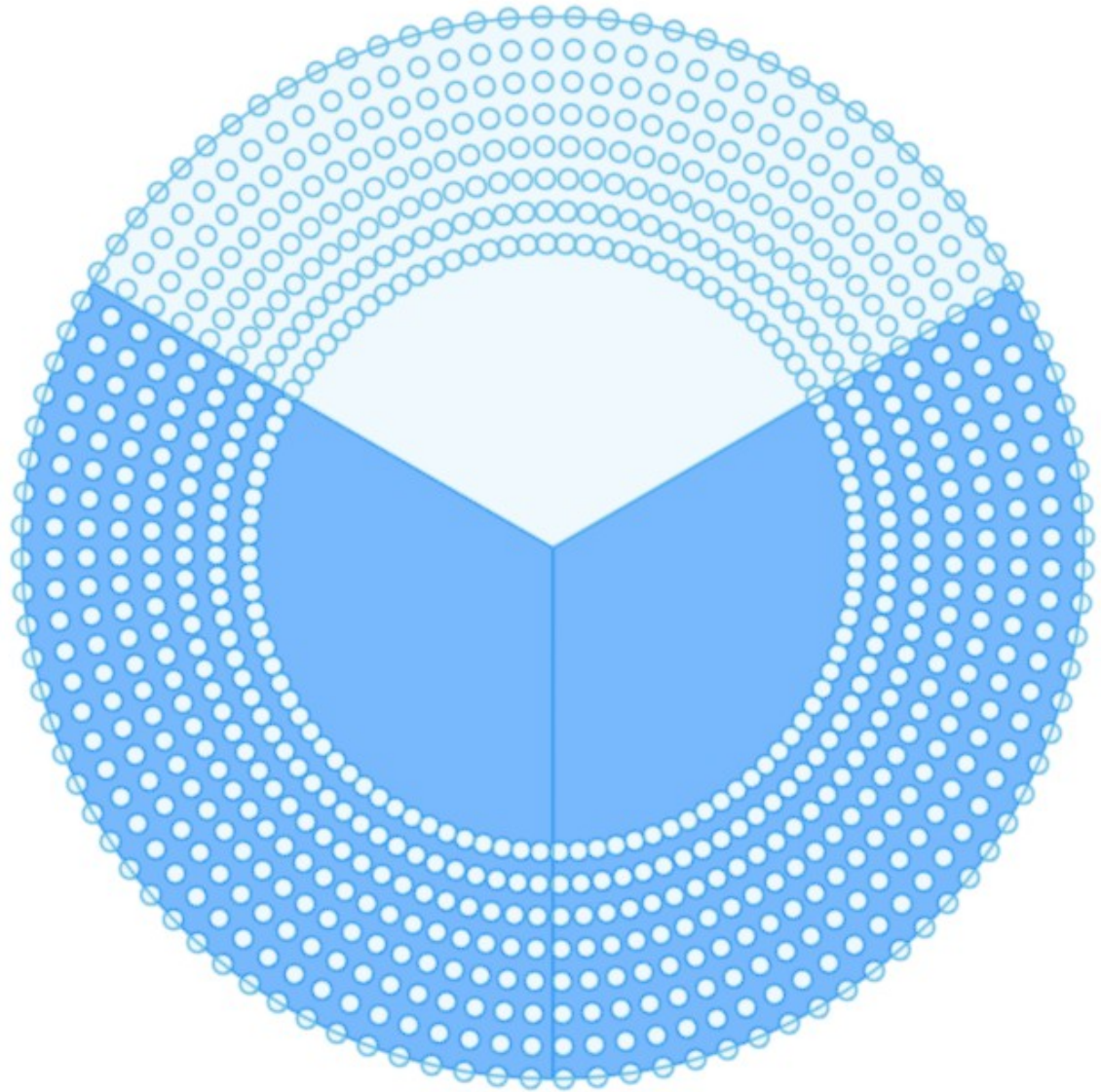
Donc $99 * 8$ (leds en ligne) = 792 LEDS virtuelles sur 360° grace à la persistance rétinienne !

$100 / (5+1)$ (nombre de LEDS en largeur par lettres + 1 pour l'espace entre lettres) = 16,6 caractères peuvent rentrer sur 360° .

Pour que cela soit plus facile à lire, on va réduire la surface du disque sur une plage de 240° .

Ainsi, sur 240° on dispose de $100x (240:360) = 66$ colonnes ce qui permet de faire apparaître sur le disque virtuel 11 caractères ($66/6 = 11$).

La zone virtuelle en bleu de 66 colonnes représente la zone sur laquelle on peut faire apparaître un texte de 11 lettres ou tout autre image.



Il convient :

De fixer la vitesse de rotation du moteur (10 tours /s)

De synchroniser le début du texte en un point précis.

Pour obtenir ce point de synchronisation on ajoute un capteur à effet Hall pour gérer avec précision l'instant correspondant à l'apparition de la première lettre du texte pendant la période des 100 000 μ s.

Remarque : 11 caractères à afficher c'est peu.

Pour allonger le texte, il suffit de coder un système de texte défilant ou seuls 11 caractères sont visibles. En agissant ainsi, on n'est plus limité par la longueur de texte ou de l'image...

Il existe plusieurs librairies pour gérer les leds adressables les plus courantes sont :

Adafruit NeoPixel Library

https://github.com/adafruit/Adafruit_NeoPixel

et

FastLED

<https://github.com/FastLED/FastLED>

Un M5stickC-Plus sera utilisé



Nous allons utiliser la librairie [FastLED](#), pour piloter les LEDS.

Le capteur à effet Hall est un A3144, il est facile à mettre en œuvre.
Un simple `digitalRead(pin)` permettra de lire son état.



Pour alimenter le microcontrôleur qui se trouve sur la partie mobile, nous allons utiliser pour transmettre l'énergie un module à induction sans contact.

<https://fr.aliexpress.com/item/4001171058433.html>

Comme je n'avais pas sous la main un moteur alimenté en continu genre 775, j'ai utilisé ma plateforme de test pour moteur pas à pas.



Pour rendre le code plus lisible dans l'IDE Arduino j'ai créé 4 fichiers
M5_POV_FastLED_v4.ino ⇒ setup et loop

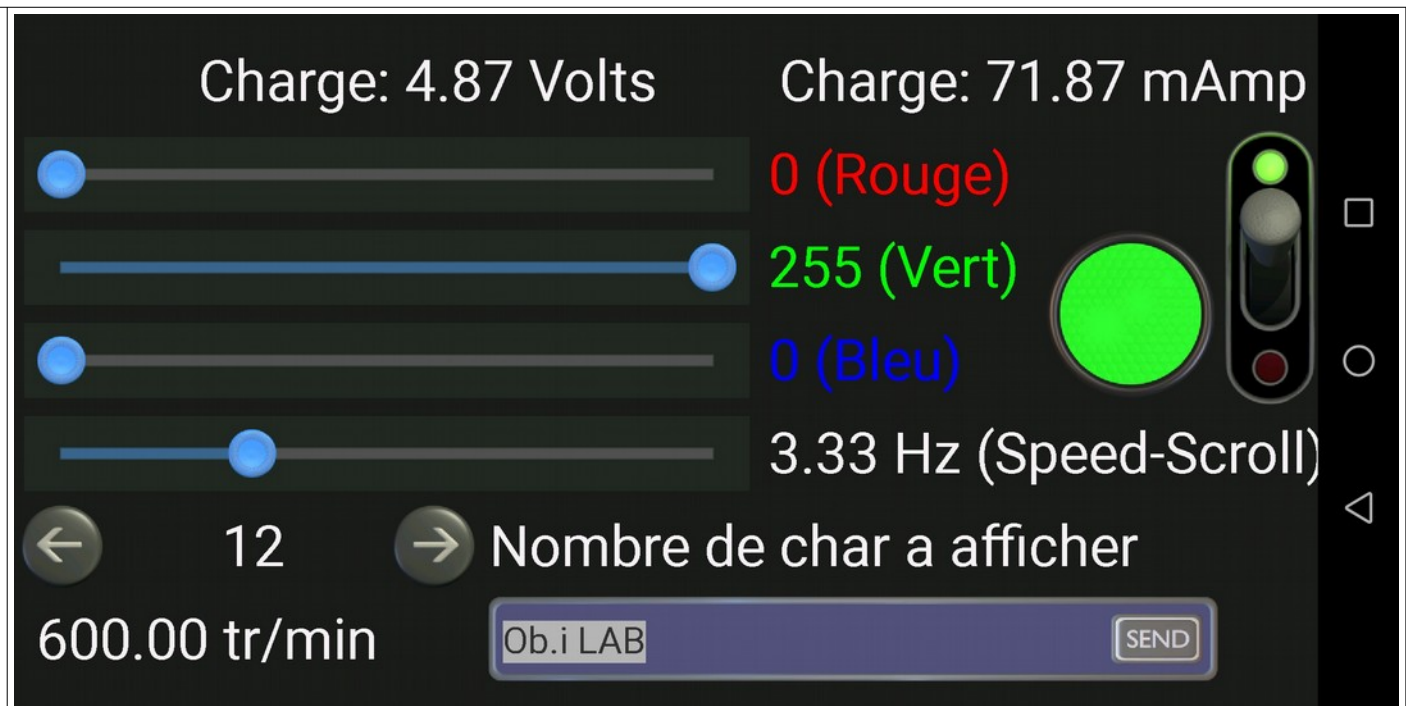
bluet.h ⇒ la partie Bluetooth

fonctions.h ⇒ La fonction qui pilote les envois à FastLED

lib.h ⇒ les tableaux

Le [M5stickC-Plus](#) dispose d'un émetteur Bluetooth une interface est disponible pour piloter le module.

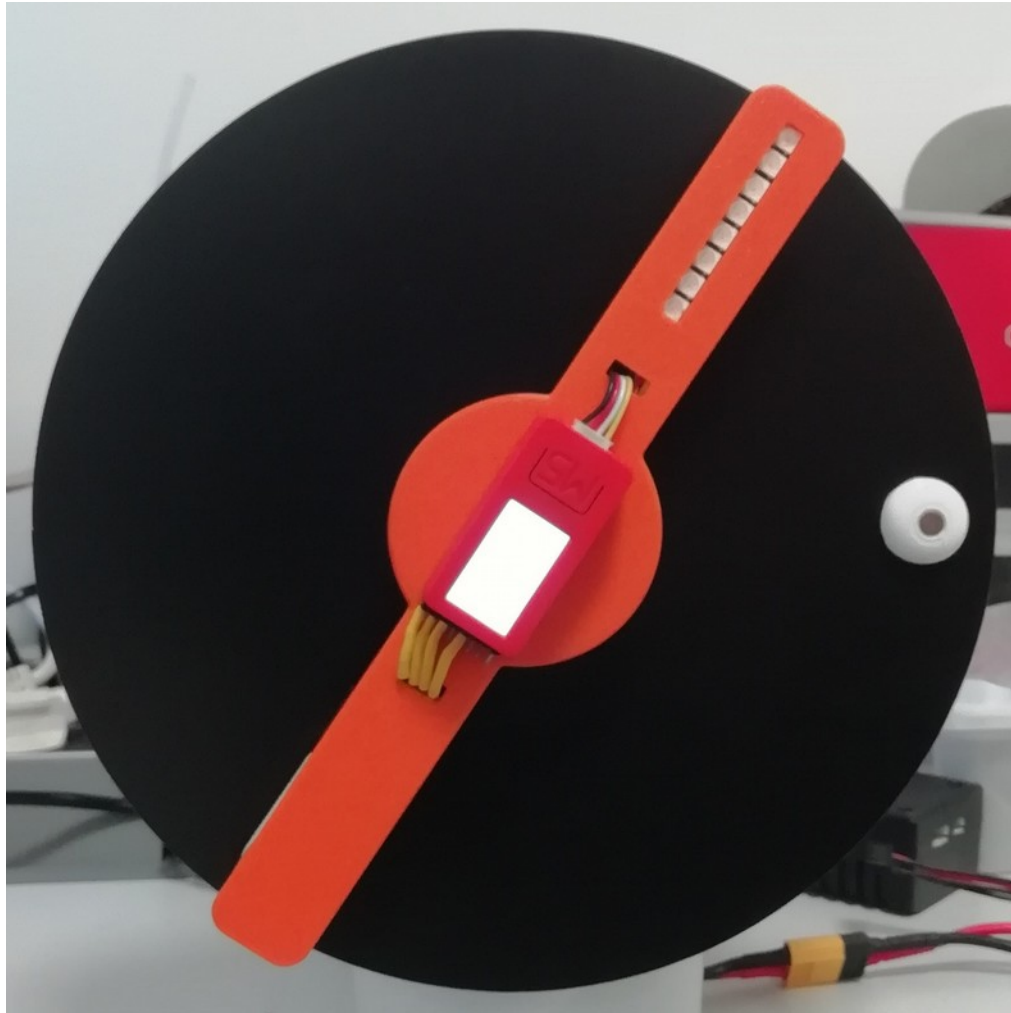
On va utiliser mon application préférée « [Bluetooth Electronics](#) » pour ce faire.



Le code qui génère l'interface graphique ci-dessus.

```
void sendBT() {
  //////////// Build panel in app
  SerialBT.println("* .kwl");
  SerialBT.println("clear_panel()");
  SerialBT.println("set_grid_size(14,7)");
  SerialBT.println("add_text(8,1,xlarge,L,Rouge,255,0,0,E)");
  SerialBT.println("add_text(8,2,xlarge,L,Vert,0,255,0,F)");
  SerialBT.println("add_text(8,3,xlarge,L,Bleu,0,0,255,G)");
  SerialBT.println("add_slider(0,4,7,10,1000,300,D,,0)"); // Vitesse Scroll
  SerialBT.println("add_text(8,4,xlarge,L,V-Scroll,245,240,245,H)");
  SerialBT.println("add_led(11,2,2,L,0,0,0)"); // led color
  SerialBT.println("add_switch(13,1,3,E,e,0,1)");
  SerialBT.println("add_slider(0,1,7,0,255,0,A,,0)"); // R
  SerialBT.println("add_slider(0,2,7,0,255,255,B,,0)"); // G
  SerialBT.println("add_slider(0,3,7,0,255,0,C,,0)"); // B
  SerialBT.println("add_button(0,5,4,M,m)"); // -
  SerialBT.println("add_text(2,5,xlarge,R," + String(maxletter) + ",245,240,245,J)"); // Nombre de caractères à afficher
  SerialBT.println("add_button(4,5,5,N,n)"); // +
  SerialBT.println("add_text(5,5,xlarge,L,Nombre de char a afficher,245,240,245,)");
  SerialBT.println("add_text(3,6,xlarge,R," + String(dt / 60) + " RPM,245,240,245,I)"); // Texte RPM
  SerialBT.println("add_send_box(5,6,8,Ob.i LAB,#,&)"); // Envoi texte
  SerialBT.println("set_panel_notes(-,,)");
  SerialBT.println("run()");
  SerialBT.println("*");
}
```

VERSION ACTUELLE...



Télécharger le code de ces exemples sur Sourceforge :
<https://sourceforge.net/projects/openhardware-eu/files/ws2812b/>

CONCLUSION

Nous espérons vous avoir intéressé dans ces exemples.

Présentation réalisée pour le FABLAB SAINT GELY-DU-FESC PIC SAINT LOUP

Le 7 juillet 2022

François & Maxime

