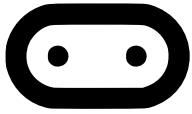


## Activité 3 : Commande d'un bandeau lumineux



Dans cette activité nous allons chercher à piloter un bandeau lumineux à partir d'une carte microprogrammable.

### 1. Présentation du système lumineux

Le système lumineux mis en œuvre lors de cette activité est un bandeau NeoPixel piloté par une carte programmable Microbit

#### Détail du bandeau lumineux

Bandeau Neopixel composé de 30 Led. Chaque led est pilotable en couleur et en luminosité de façon indépendante.  
Les instructions de commande sont transmises au bandeau par l'intermédiaire d'un bus de données série.



Figure 1: Bandeau Néopixel

#### Carte microprogrammée Microbit

Cette carte microprogrammée intègre un environnement de programmation Micropython qui permet, malgré l'absence de système d'exploitation, de programmer la carte en langage Python. Cette carte possède un bus série UART pour piloter différents systèmes électronique dont les bandeaux Neopixels.

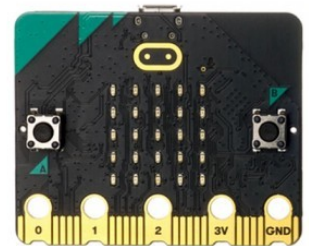
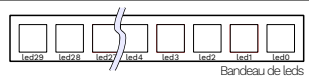
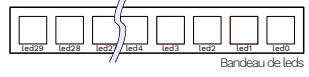
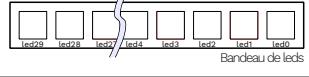



Figure 2: Carte Micro:bit

#### Prise en main de la commande d'éclairage

Le pilotage du bandeau sous Python est simplifié par l'utilisation de la bibliothèque pixeldriver dont les principales commandes sont définies en page 4.

2. A partir de la documentation page 4 et pour chaque instruction du tableau suivant, **calculer** la valeur binaire du masque et **repérer** les leds allumées

Commande	Mask exprimé en binaire	Etat du bandeau
<code>lightOn(0)</code>		
<code>lightOn(1)</code>		
<code>lightOn(536870912)</code>		
<code>lightOn(3)</code>		

La fonction `led_gauche(nb)` suivante permet de faire clignoter la led pilotée par le MSB un nombre de fois défini en paramètre.

```
from neodriver import *

connect(pin0)

def led_gauche(nb) :
    '''fait clignoter la led 29 nb fois
    parametre nb (entier) : nombre de clignotements
    '''
    for i in range(nb) :
        lightOn(536870912, True)
        sleep(500)
        lightOn(0, True)
        sleep(500)
```

3. **Saisir** ce programme puis le **tester** avec le bandeau lumineux.
4. A partir de cette fonction, **créer** une autre fonction `double_clignotement(nb)` qui commande le clignotement alternatif de la led 29 puis la led 24 un nombre de fois passé en paramètre

### Scénario 1

But : programmer un nouveau clignotement qui allume les leds paires puis les impaires.

5. Calculer le masque nécessaire à l'allumage des leds paires puis celui pour les leds impaires.

*Remarque* : Le calcul à la main des masques est fastidieux, il peut plus facilement être réalisé avec un petit programme Python mettant en œuvre une boucle `for`.

6. **Programmer** cette fonction et **tester** ce scénario.



## Scénario 2

But : programmer un chenillard partant de la droite vers la gauche.

7. **Calculer** le masque commandant l'allumage de la led0 puis celui commandant la led 1 et enfin celui commandant la led 2.
8. **En déduire** le coefficient multiplicateur liant le masque commandant une led avec celui commandant la led précédente.
9. **Programmer** ce scénario2 avec une fonction puis la **tester** sur le bandeau lumineux.
10. **Modifier** ce programme afin de programmer le scénario 2 défini au dessus.

### Modification du programme

L'opérateur `>>` permet de décaler vers la droite les bits d'un nombre binaire ; `<<` décale de la même manière vers la gauche. Par exemple `16 >> 1` renvoie la valeur 8.

11. **Calculer** les valeurs renvoyée par les instructions suivantes :

Instruction	Valeur renvoyée	Instruction	Valeur renvoyée
<code>8 &lt;&lt; 1</code>		<code>25 &lt;&lt; 2</code>	
<code>32 &gt;&gt; 2</code>		<code>131 &gt;&gt; 3</code>	

Le scénario 2 effectué précédemment décale l'allumage des leds.

12. **Modifier** le programme du scénario 2 afin de mettre en œuvre les opérateurs de décalage de bits ( `<<` et `>>` ).

Pour aller plus loin :

- x **Créer** une fonction `chenillard_droite()` qui crée un chenillard de gauche à droite
- x **Créer** une fonction `chenillard(n, type)` qui commande un nombre de fois `n` passé en paramètre, l'allumage d'un chenillard faisant un aller (de droite à gauche) ou un retour (gauche à droite) ou un aller/retour en fonction d'un paramètre `type` passé en paramètre (0 : aller ; 1 : retour ; 2 : aller/retour)



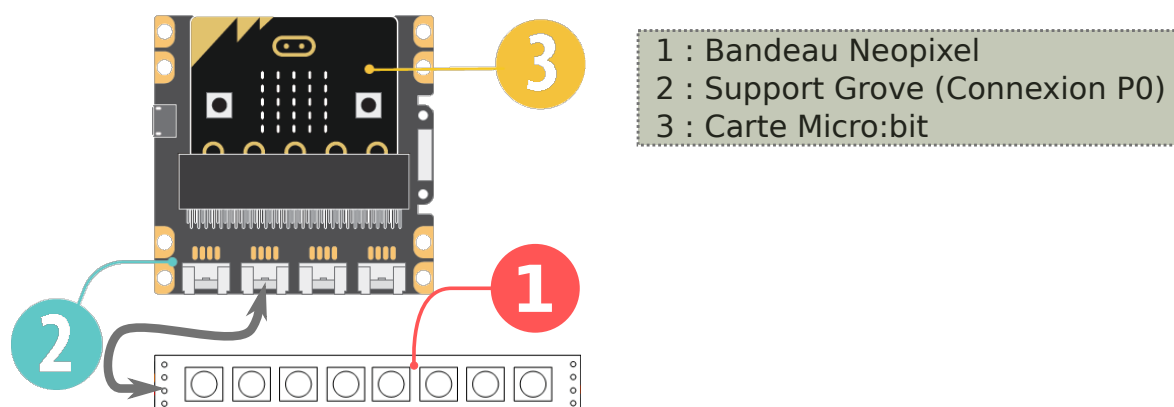
# Document ressource : Description de la platine

## 1. Présentation générale

La platine est composée d'une carte microprogrammée Micro:bit et d'un bandeau lumineux Néopixel communiquant entre eux par l'intermédiaire d'un bus de communication série.

Remarque : Pour en savoir plus sur la carte Micro:bit une description complète est disponible sur Moodle dans l'onglet 'Ressources pour la NSI'

## 2. Câblage de la platine



## 3. Programmation de la carte Micro:bit

La programmation de la carte s'effectue avec l'IDE Mu editor disponible sur le bureau de l'ordinateur. La bibliothèque neodriver installée dans la carte met à disposition des commande simplifiées de pilotage du bandeau Néopixel. L'utilisation de cette bibliothèque nécessite de l'importer en début de programme :

```
from neodriver import *
```

Les fonctions disponibles dans cette bibliothèque sont les suivantes :

### **connect**

```
connect(pin)
```

La fonction connect permet d'initialiser la communication entre la carte Micro:bit et le bandeau lumineux. Le paramètre pin renseigne le numéro de patte sur laquelle est connecté le bandeau lumineux.

La communication avec le bandeau lumineux connecté sur la patte P0 (cas décrit sur le schéma de câblage ci-dessus) s'effectue avec la commande :

```
connect(pin0)
```



## lightOn

`lightOn(mask, verbose)`

La fonction `lightOn` permet de commander l'allumage de certaines leds en fonction de la valeur de `mask`. Chaque led est commandée suivant l'état d'un bit de `mask`. La led 0 est commandée par le LSB de `mask`, la led 1 est commandée par le deuxième bit en partant de la droite ainsi de suite jusqu'à la led 29 qui sera commandée par le bit 29 (MSB). La mise à 1 du bit du masque a pour conséquence d'allumer la led correspondante.

Exemple :

```
lightOn(10)
```

allume les leds 1 et 3 du bandeau lumineux

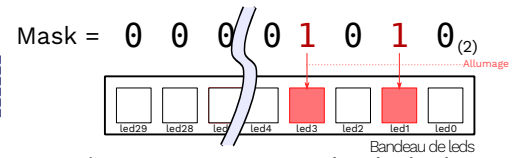


Figure 3: Commande de led  
(mask = 10)

Le paramètre `verbose` permet lorsqu'il est `True` de renvoyer sur la console à chaque appel de la fonction, la valeur en binaire de `mask`. Ce paramètre est optionnel (`False` par défaut).

Exemple :

Pour allumer les leds 1 et 3 et renvoyer la valeur binaire de `mask` sur la console, il faut écrire :

```
lightOn(10, True)
```

## color

`color(rgb)`

La fonction `color` permet de choisir la couleur des leds à l'aide de triplet de valeurs `rgb`. Chaque valeur est codée sur un octet et peut donc prendre des valeurs comprises entre 0 et 255.

- ✗ La première valeur du triplet règle la nuance de rouge (0 : 0 % / 255 : 100 %)
- ✗ La deuxième valeur règle la nuance de vert
- ✗ La troisième valeur règle la nuance de bleu

L'affichage de la couleur choisie s'effectue après l'exécution de la commande `lightOn` suivante.

Exemple :

L'allumage rouge de la led0 est commandée par les instructions :

```
color((255,0,0))  
lightOn(1)
```

L'allumage violet des led0 et led1 est commandée par les instructions :

```
color((255,0,255))  
lightOn(3)
```

