Eleman Delorme 28 août 2025

TP Neopixel

La carte **Kitronik ZIP Halo HD** pour micro:bit contient 60 LEDs qui sont tous adressable en format RGB. Le circuit contient également un circuit RTC (real time clock) pour créer des horloges.

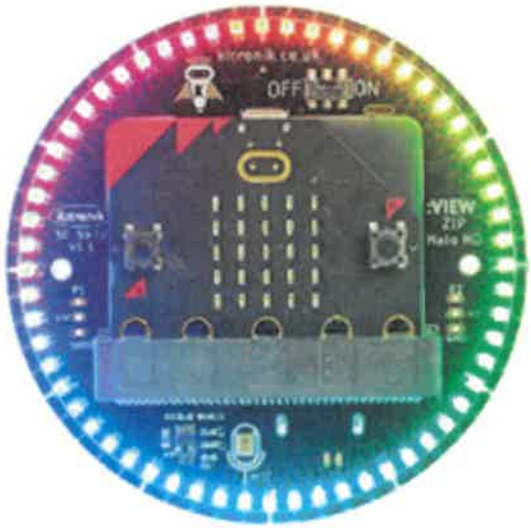


Figure 3 La carte Kitronik ZIP Halo HD.

Préparation

- Téléchargez le fichier **`ZIP_halo_hd.py`**
- Lancez le navigateur Google Chrome (Safari et FireFox ne marchent pas avec WebUSB)
- Ouvrez le projet dans l'éditeur
- Connectez votre microbit
- Téléchargez le programme
- Observez le code est le fonctionnement des programmes 0..9
- Expliquez les lignes de code
- Ecrivez les programmes demandés (

Importation

Expliquez chaque ligne de code.

`from microbit import *`	importe du module microbit toutes (*) les fonctions et variables
`import neopixel`	importe un module neopixel pour pouvoir contrôler le neopixel avec Python
`import music`	importe un module de musique
`import random`	importe un module pour créer des choses aléatoires

Les couleurs

Expliquez chaque groupe de code.

`red = (10, 0, 0)`
`green = (0, 10, 0)`
`blue = (0, 0, 10)`
	définit les tuples (R, G, B) des 3 couleurs de base, red, green, blue, avec une intensité de 10 sur 255

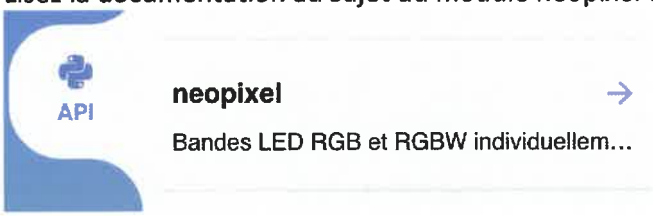
`yellow = (10, 10, 0)`
`magenta = (10, 0, 10)`
`cyan = (0, 10, 10)`
`orange = (10, 5, 0)`
	définit les tuples (Y, M, C) qui sont les couleurs primaires et orange qui est une couleur secondaire avec une intensité de 10 à 255

`black = (0, 0, 0)`
`gray = (5, 5, 5)`
`white = (10, 10, 10)`
	définit les couleurs achromatiques sur une intensité allant de 0 à 255

`colors = (red, orange, yellow, green, cyan, blue, magenta)`	nomme toute les couleurs définies auparavant

Le module neopixel

Lisez la documentation au sujet du module neopixel dans API et répondez aux questions.



Expliquer les termes et fonctions.

API	
`np.clear()`	efface tout les pixels
`np.show()`	afficher les pixels
`np.fill(RGB)`	colorer tout les pixels d'une valeur RGB/RGBW donnée

Initialisation

Expliquez chaque groupe de code.

`np = neopixel.NeoPixel(pin8, 60)`	np = neopixel.NeoPixel ↳ mentionne le NeoPixel pin8 → la broche 8 contrôle la bande 60 → nbre de pixels sur la bande

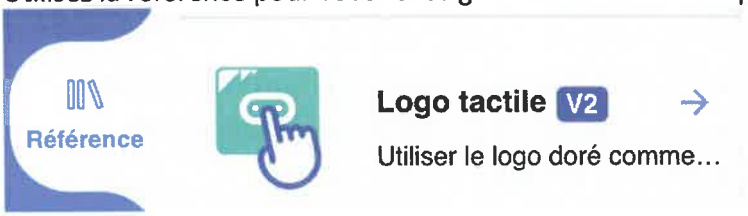
Variables globales

Expliquez chaque variable.

`prog = 0`	indique le programme de base
`i = 0`	indique l'indice de la led
`j = 0`	indique l'indice de couleur

Choix du programme

Utilisez la référence pour vous renseigner sur les fonctions que vous ne connaissez pas.



Expliquez chaque ligne de code.

code	explication
`while True:`	début d'une boucle infinie
`if button_a.was_pressed():`	début d'un bloc conditionnel qui sera exécuté si le bouton A est appuyé
`prog = (prog + 1) % 10`	la variable prog est incrémenté, mais limité à la plage 0..9 avec l'opérateur modulo (%)
`display.show(prog)`	l'écran montre le programme
`np.clear()`	tout les pixels du NeoPixel s'effacent.
`music.pitch(440, 20)`	Joue un La pendant 20 ms

Choix de couleur

Expliquez chaque ligne de code.

code	explication
`if pin_logo.is_touched():`	début d'un bloc conditionnel qui sera exécuté si le pin logo est touché.

`j = (j + 1) % len(colors)`	A chaque fois que le logo est touché, la couleur change d'un indice
`music.pitch(540, 20)`	Joue un son à 540 Hz pendant 20 ms
`sleep(100)`	le programme s'arrête pendant 100 ms

Passage par 0

Expliquez d'abord ce que fait le programme et ensuite expliquez chaque ligne de code.

code	explication
`# explication`	A chaque passage par 0 de l'indice i, ce bloc efface les LEDs et fait jouer un petit bip.
`if i == 0:`	si l'indice est égal à 0
`np.clear()`	les leds se réinitialisent
`music.pitch(880, 20)`	Joue un son à 880 Hz pendant 20 ms

Programme 0

Expliquez d'abord ce que fait le programme et ensuite expliquez chaque ligne de code.

code	explication
`# explication`	Le programme 0 remplit les 60 LEDs avec une couleur, dans le sens de l'horloge, avec un intervalle de 20ms. Au passage par 0 il y a un bip et les LEDS sont effacés.
`if prog == 0:`	Si le programme est égal à 0
`np[i] = colors[j]`	affiche la couleur en fct de l'intensité
`np.show()`	affiche le résultat
`sleep(20)`	s'arrête pendant 20 ms

`i = (i + 1) % 60`	augmente l'intensité lumineuse de 1 à chaque fois

Programme 1

Le programme 1 allume une seule LED et la fait avancer dans le sens de l'horloge.
Au passage par 0 il y a un bip.
Trouvez le code selon les explications.

code	explication
np.clear()	effacer la LED précédente
np[i] = colors[j]	allumer la LED avec l'indice i avec la couleur j
np.show()	montrer cette nouvelle LED
sleep(20)	attendre 20 ms
i = i+1	aller à la LED suivante

Programme 2

Expliquez d'abord ce que fait le programme et ensuite expliquez chaque ligne de code.

code	explication
`# explication`	Affiche les couleurs de la variable sur les 60 pixels
`j = i % len(colors)`	définit les couleurs à afficher
`np[i] = colors[j]`	allume la LED avec l'indice i avec la couleur j
`i = (i + 1) % 60`	indique aux LED de faire un tour complet

Programme 3

Expliquez d'abord ce que fait le programme et ensuite expliquez chaque ligne de code.

`# explication`	Affiche une led qui fait la moitié des pixels à chaque seconde. Si on appui sur le bouton B, la LED fait demi-tour

`if button_b.is_pressed():` `  i = i - 1`	Si l'on appui sur le bouton b, la led fait demi-tour
`else:` `  i = i + 1`	Sinon la led augmente d'un à chaque fois
`i = i % 30`	la led ne parcourt que 30 leds

Programme 4

Le programme 4 allume les LED 0 à 59 dans l'ordre de l'horloge, mais seulement quand le bouton B est appuyé. revient. Au passage par 0 il y a un bip et les LEDS sont effacés.
Trouvez le code selon les explications.

if button_b.is_pressed()	**début d'un bloc conditionnel qui sera exécuté seulement si le bouton B est appuyé.**
i = i+1 i = i % 60	**la variable i est incrémentée et limitée à la plage 0..59 à l'aide de l'opérateur modulo.**
np[i] = color[j]	**allumer la LED avec l'indice i avec la couleur j**
np.show	**montrer cette nouvelle LED**
sleep(20)	**attendre 20 ms**

Programme 5

Le programme 5 allume les LED 0 à 29 avec une intensité croissante de la couleur magenta.
A 30 il y a un bip et les LEDS sont effacés et ça recommence.
Trouvez le code selon les explications.

`np[i] = (i, 0, i)`	**la LED i est allumée avec une nuance de magenta dont l'intensité est proportionnelle à i (0..29)**
i = (i+1) % 30	**la variable i est incrémentée, mais limitée à la plage 0..29 à l'aide de l'opérateur modulo.**
np.show()	**montrer cette nouvelle LED**
sleep(20)	**attendre 20 ms**

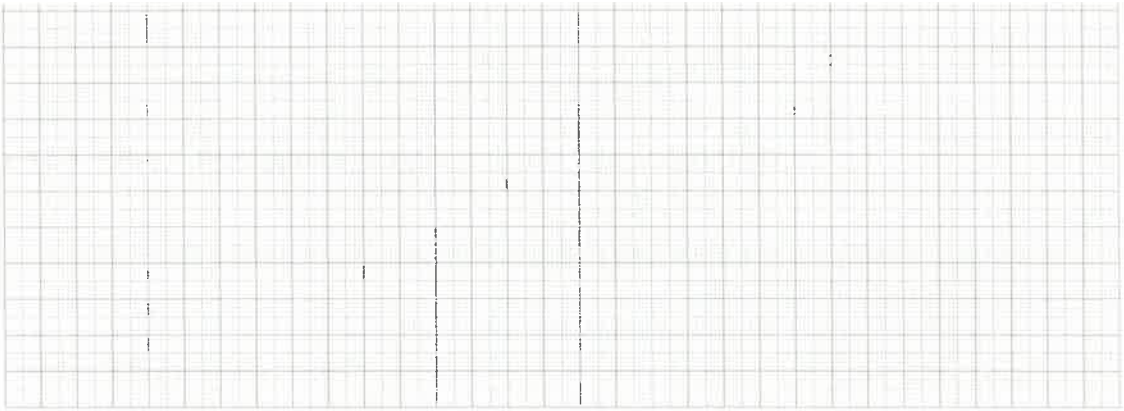
Programme 6

Expliquez d'abord ce que fait le programme et ensuite expliquez chaque ligne de code.

`# explication`	Fait apparaître les couleurs de l'arc-en-ciel toute les 29 leds en boucle.

`i = (i+1) % 20`	la variable i est incrémentée et limitée à la plage 0...19
`np[i] = (20-i, i, 0)`	affiche les pixels depuis le début avec des couleurs variables
`np[i+20] = (0, 20-i, i)`	affiche les pixels depuis le 20ème avec des couleurs variables
`np[i+40] = (i, 0, 20-i)`	affiche les pixels depuis le 40ème avec les couleurs variables.

Faits un graphe pour i = 0..59. Affiche les 3 graphes pour R, G, B, avec une plage de 0..20. Utilisez dans votre graphe un stylo avec les 3 couleurs rouge, vert et bleu. Indiquez les 7 couleurs de l'arc-en-ciel.



Programme 7

Le programme 7 allume de façon aléatoire une seule LED parmi 0..59 toutes les 50 ms. Quand la LED0 est allumée, un bip est joué.
Trouvez le code selon les explications.

`np.clear()`	**effacer les LEDs**
`i = random.randint(0, 59)`	**la variable i est assignée avec un entier aléatoire entre 0 et 59**
`np[i] = [10*i for i in colors[j]]`	**la LED désignée par l'indice i est allumée avec la couleur indiquée par la variable j, avec 10 fois l'intensité.**
`np.show()`	**afficher toutes les LEDs**
`sleep(50)`	**attendre 50 millisecondes**

Programme 8

Le programme 8 affiche avec une LED rouge les 1/60 de seconde et avec une LED verte les secondes.
Trouvez la signification de la fonction **`running_time()`** dans la référence.

`t = running_time()`	ajoute une horloge dans le programme pour avoir les secondes

`i = (t % 1000) // 17`	L'opération modulo réduit la valeur de t à la plage 0...999. La division entière // par 17 réduit la valeur à la plage 0..58
`i2 = (t//1000) % 60`	La division réduit la valeur de t et le module réduit la valeur de la plage à 59
`np[i] = red`	indique que la couleur du pixel est rouge
`np[i2] = green`	indique que la couleur du pixel nord est vert.

Programme 9

Le programme 8 affiche une boussole avec 2 LED lumineuses. La LED rouge pointe vers le nord, lad LED bleu pointe vers le sud.

Trouvez la signification de la fonction **`compass_heading()`** dans la référence.

`if button_b.is_pressed():` `  compass_on = True`	Si le bouton B est appuyé, la variable booléenne compass_on devient True
`if compass_on:`	Si la boussole est activée
`compass.heading()`	calibre la boussole
`compass.heading() // 6`	
`i = 59 - c.heading() // 6`	
`np[i] = red`	le premier pixel sera rouge
`np[(i+30)%60] = blue`	le 2ème pixel, 30 plus loin que le premier sera bleu