|  |  |  |  |
| --- | --- | --- | --- |
| http://localhost:8087/WEFiles/Image/WEImage/objet-connectes-WEd607547dc3.png | Lycée Joliot Curie à 7 | ***SNT – Séance n°5*** | Classe de Seconde |
| **Informatique embarquée et objets connectés** | | **Classe 2° …**  **Nom : ………………..**  **Nom : ………………..** |

**Que sont les objets connectés ...” Vidéo 1**

Résumé de la vidéo 1 du chapitre ............................................................................................................................................................................................................

............................................................................................................................................................................................................

............................................................................................................................................................................................................

............................................................................................................................................................................................................

............................................................................................................................................................................................................

............................................................................................................................................................................................................

............................................................................................................................................................................................................

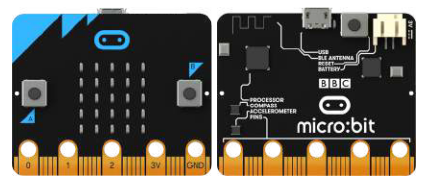
............................................................................................................................................................................................................

............................................................................................................................................................................................................

**Présentation de la carte BBC micro:bit**

**BBC micro:bit** est une carte à [microcontrôleur](https://fr.wikipedia.org/wiki/Microcontr%C3%B4leur) conçue en 2015 au Royaume-Uni pour développer l'apprentissage de l'algorithmique et de la programmation. Pourvu de capteurs et d'actionneurs, ce petit ordinateur possède la dernière technologie qui équipe les appareils modernes : téléphones mobiles, réfrigérateurs, montres intelligentes, alarmes antivol, robots, etc...

Ainsi, il s'apparente à ce que l'on nomme l' Internet des objets : Internet of Things, abrégé IoT .



La carte micro:bit dispose des spécificités techniques (<https://microbit.org/fr/guide/features/>) suivantes :

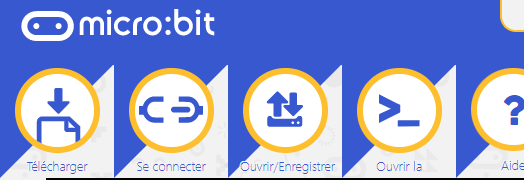
|  |  |
| --- | --- |
| - 25 LEDs programmables individuellement  - 2 boutons programmables  - Broches de connexion  - Capteurs de lumière et de température | - Capteurs de mouvements (accéléromètre et boussole)  - Communication sans fil, via Radio et Bluetooth  - Interface USB |

**Partie A : Découverte des fonctionnalités de la carte BBC micro:bit**

**1-Toute 1ère connection:**

- Brancher la carte sur un port USB.

- Sur Capneuronal chapitre 5, ouvrir l’éditeur "Python - Micro:bit V2"

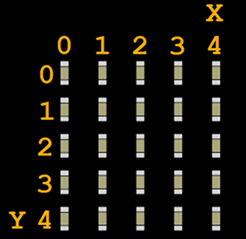


- Cliquez sur “se connecter” et sélectionner la carte

- Cliquez sur “flash”

Le programme écrit en Python se “téléverse” sur la carte et s’exécute.

*A chaque fois que vous changerez le programme, il faudra le téléverser sur la carte.*

**2-** **Commandes de base de l'afficheur, matrice de 5x5 LEDs**

- Copier le programme ci-contre dans l’éditeur

from microbit import \*

while True:

display.scroll('Hello, World!')

display.show(Image.HEART)

sleep(2000)

While : tant que … Matrice 5 x 5

**a- Afficher un texte "défilant" display.scroll(string, delay=400)**

Nous allons commencer par afficher quelques informations sur l'afficheur.

- La première ligne de ce programme importe la bibliothèque de fonctions micro:bit.

from microbit import \*

display.scroll( "SNT" )

- La deuxième ligne fait défiler un message à l’écran. **Cela n'arrive qu'une seule fois.**

Exercice 1: Modifier le programme précédent pour qu'il affiche le texte de ton choix.

from microbit import \*

display.scroll(" ", delay=… )

La vitesse de défilement peut être ralentie ou accélérée à l'aide du paramètre delay . Plus le nombre est grand, plus le défilement est lent.

from microbit import \*

…

display.scroll(" ", delay=… )

Exercice 2 : Modifier votre programme pour que l’affichage ne s’arrète jamais !

**b- Afficher une "image" display.show(image)**

Exercice 3 : On constate que la carte est un peu triste. Modifier le programme précédent pour lui redonner de la joie.

from microbit import \*

display.show(Image.SAD)

Aide: ci-dessous la liste des images intégrées:

Image.HEART Image.HEART\_SMALL Image.HAPPY Image.SMILE Image.SAD Image.CONFUSED

Image.ANGRY Image.ASLEEP Image.SURPRISED Image.SILLY Image.FABULOUS Image.MEH Image.YES

Image.NO Image.CLOCK12, Image.CLOCK11, Image.CLOCK10, Image.CLOCK9, Image.CLOCK8,

Image.CLOCK7, Image.CLOCK6, Image.CLOCK5, Image.CLOCK4, Image.CLOCK3, Image.CLOCK2,

Image.CLOCK1 Image.ARROW\_N, Image.ARROW\_NE, Image.ARROW\_E, Image.ARROW\_SE,

Image.ARROW\_S, Image.ARROW\_SW, Image.ARROW\_W, Image.ARROW\_NW Image.TRIANGLE

Image.TRIANGLE\_LEFT Image.CHESSBOARD Image.DIAMOND Image.DIAMOND\_SMALL Image.SQUARE

Image.SQUARE\_SMALL Image.RABBIT Image.COW Image.MUSIC\_CROTCHET Image.MUSIC\_QUAVER

Image.MUSIC\_QUAVERS Image.PITCHFORK Image.XMAS Image.PACMAN Image.TARGET Image.TSHIRT

Image.ROLLERSKATE Image.DUCK Image.HOUSE Image.TORTOISE Image.BUTTERFLY

Image.STICKFIGURE Image.GHOST Image.SWORD Image.GIRAFFE Image.SKULL Image.UMBRELLA

Image.SNAKE

Essayer plusieurs images intégrées.

from microbit import \*

display.show(Image. )

**c- Créer sa propre image**

Chaque pixel LED sur l’affichage physique peut prendre une parmi dix valeurs.

from microbit import \*

bateau = Image( "05050:"

"05050:"

"05050:"

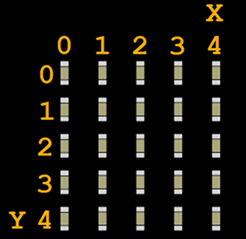
"99999:"

"09990" )

display.show(bateau)

Si un pixel prend la valeur 0 c’est qu’il est éteint. Littéralement, il a une luminosité de zéro.

En revanche, s’il prend la valeur 9 il est à la luminosité maximale. Les valeurs de 1 à 8 représentent des niveaux de luminosité entre éteint (0) et « au maximum » (9).

Comment dessiner une image?

**Chaque ligne** de l’affichage physique est représentée par une ligne de nombres se terminant par : et entourée de guillemets doubles " .

**Chaque nombre** indique une luminosité. Il y a cinq lignes de cinq nombres donc il est possible de spécifier la luminosité individuelle de chacune des cinq LED sur chacune des cinq lignes sur l’affichage physique. C’est ainsi que l'on crée une image.

Exercice 4 : Le programme précédent crée et affiche l'image d'un bateau à deux mâts.

Le modifier (en le recopiant) ci-dessous pour obtenir un bateau à un **seul mât**.

from microbit import \*

bateau2 = Image( " :"

" :"

" :"

" :"

" " )

display.show(bateau2)

Exercice 5 : Rédiger ci-dessous le programme qui affiche l'image suivante:

from microbit import \*

lettreH = Image( " :"

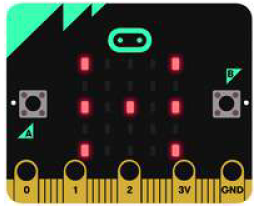
" :"

" :"

" :"

" " )

display.show( )



Remarque : on peut aussi écrire les images en une seule ligne:

from microbit import \*

bateau = Image( "05050:05050:05050:99999:09990" )

display.show(bateau)

**d- Les pixels ( display.set\_pixel(x, y, val) )**

Vous pouvez régler la luminosité des pixels de l'affichage individuellement de 0 (désactivé) à 9 (luminosité

maximale).

Pour des informations sur les coordonnées de l'affichage, voir le guide pour matrice à LED

(https://microbit.org/guide/hardware/leds/) .

from microbit import \*

display.set\_pixel( 1 , 4 , 9 )

Exécuter le programme suivant:

**Exercice 6:** Recopier le programme précédent ci-dessous et le modifier pour allumer la LED du centre de la

matrice.

**3- Boucle while**

Le programme suivant utilise une boucle while pour faire clignoter le pixel central de manière répétée sur

l’écran.

La boucle while se répète tant que la condition spécifiée est vraie ( True ).

Dans ce cas, nous avons dit que la condition est vraie. Cela crée une boucle infinie.

Le code qui doit être répété est en retrait (c'est une "indentation" du texte).

L'instruction de veille sleep() provoque la pause du micro:bit pendant un nombre défini de millisecondes choisi entre parenthèses.

from microbit import \*

while True :

display.set\_pixel( 2 , 2 , 9 )

sleep( 500 )

display.clear()

sleep( 500 )

L'instruction display.clear() éteint l'affichage.

Exécuter le programme ci-contre:

Dans le programme suivant que vous exécuterez, on importe le module **random** de MicroPython et on l'utilise pour afficher un pixel au hasard sur la matrice.

from microbit import \*

import random

n=random.randint( 0 , 4 )

p=random.randint( 0 , 4 )

display.set\_pixel(n, p, 9 )

Exécuter le programme ci-contre:

*Tester le programme précédent plusieurs fois de suite. Pour cela, redémarrer la micro:bit en appuyant sur le bouton RESET situé à l'arrière de la carte.*

Exercice 7: Ecrire un programme ci-dessous qui allume successivement et indéfiniment des pixels au hasard à l'écran (aide: utiliser une boucle infinie).

from microbit import \*

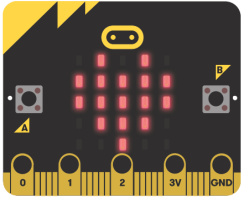
import random

Exercice 8:

from microbit import \*

import random

Ecrire ci-dessous un programme qui fait clignoter un cœur indéfiniment (voir illustration ci-dessous).



from microbit import \*

while True :

for i in range ( 5 ):

display.set\_pixel(i, 0 , 9 )

sleep( 200 )

display.clear()

**4- Boucle for**

Le programme suivant utilise une boucle for pour faire défiler un pixel sur une ligne. Exécutez-le.

Exercice 10 (la double boucle) : S'inspirer du programme précédent pour réaliser un programme ci-dessous

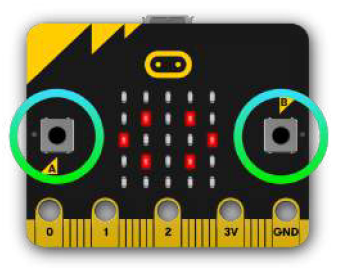
qui fait défiler un pixel sur tout l'écran.

from microbit import \*

while True :

**5- Les entrées boutons A, B et A+B – programmation**

Bouton B



Bouton A

Il y a deux boutons sur la face avant du micro:bit (étiquetés A et B). On peut détecter quand ces boutons sont pressés, ce qui permet de déclencher des instructions sur l'appareil.

Exemples avec le boutton A:

button\_**a**.**is**\_pressed() : renvoie True si le bouton spécifié est actuellement enfoncé et False sinon.

button\_**a**.**was**\_pressed() : renvoie True ou False pour indiquer si le bouton a été appuyé depuis le démarrage de l'appareil ou la dernière fois que cette méthode a été appelée.

Exemple : Essayer le programme suivant qui fait défiler le texte "SNT" indéfiniment. On introduit l'instruction conditionnelle *if* qui va tester si le bouton A a été pressé (pendant le défilement du texte ou pendant la pause), auquel cas le programme s'arrête en exécutant la commande break .

from microbit import \*

while True :

display.scroll( "SNT" )

sleep( 200 )

if button\_a.was\_pressed():

break

display.show(Image.SAD)

La touche *Reset* permet de relancer le programme.

Exercice 11: Pierre feuille ciseaux!

from microbit import \*

pierre = …………………….

feuille = …………………….

ciseaux = Image( "99009:"

"99090:"

"00900:"

"99090:"

"99009:" )

while True:

if button\_a.is\_pressed() and button\_b.is\_pressed():

display.show(ciseaux)

sleep( … )

elif button\_a.is\_pressed():

display.show(………….)

sleep( … )

elif button\_... .is\_pressed():

display.show(…………)

sleep( … )

……………………..

sleep( 100 )

Compléter le programme suivant dans lequel

- une pression simultanée sur les boutons A et B affichera une image de ciseaux.

- Sinon si, une pression sur le bouton A affichera une image de pierre.

- Sinon si, une pression sur le bouton B affichera une image de feuille.

Il faudra créer vous-même l'image de la pierre et de la feuille avec un temps d'affichage de 0,5 seconde.

**6- Capteur de lumière**

En inversant les LEDs d'un écran pour devenir un point d'entrée, l'écran LED devient un capteur de lumière basique, permettant de détecter la luminosité ambiante.

La commande display.read\_light\_level() retourne un entier compris entre 0 et 255 représentant le niveau de lumière.

Plus l’entier est élevé et l’intensité lumineuse est élevée.

from microbit import \*

soleil = Image( "90909:09990:99999:09990:90909:" )

lune = Image( "00999:09990:09900:09990:00999:" )

while True :

if display.read\_light\_level() > … :

display.show(soleil)

else :

display.show(………..)

sleep( 500 )

Exercice 12: Compléter le programme ci-contre qui affiche une image de lune si on baisse la luminosité (en recouvrant la carte avec sa main par exemple) et un soleil sinon.

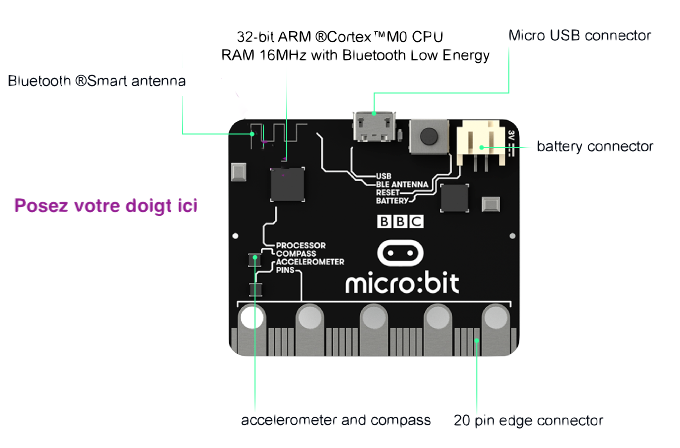
Exercice 13: créer un programme qui affiche le niveau de luminosité et le tester avec la LED d'un téléphone portable ou une lampe-torche par exemple. Plus la luminosité sera élevée, plus il y aura de LEDs affichées sur

la matrice.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  8  9  10  11  12  13  14 |  |  |  | 15  16  18  19  20 |  |  |  |

**6- Capteur en température**

Le micro:bit n’a pas un capteur de température dédié. Au lieu de cela, la température fournie est en fait la température de la puce de silicium du processeur principal. Comme le processeur chauffe peu en fonctionnement, sa température est une bonne approximation de la température ambiante. L'instruction ***temperature()*** renvoie la température de la carte micro:bit en degrés Celsius.

Exercice 13: Ecrire un programme qui affiche la température (aide: on pourra utiliser l'instruction

display.scroll() ;

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5 |  |  |  |

**Les défis !**

Le Shield (bouclier) est parmi nous !

|  |  |  |
| --- | --- | --- |
|  |  |  |
| A quoi sert le Shield Grove pour microbit BBC ?  C'est une interface qui permet de connecter des capteurs et actionneurs grove à la carte Microbit | Les capteurs et actionneurs disponibles se connectent sur la broche P0/P14  - Ruban led - Afficheur digital  - servomoteur – Haut parleur | Les cordons GROVE sont toujours constitués de 4 fils |

**Défi 1 : le ruban à Leds**

from microbit import \*

import neopixel

from random import randint

np = neopixel.NeoPixel(pin0, 8)

# 30 LEDS connectées sur la pin0

while True:

# Pour chaque NeoPixel du stick, de l'adresse\_pixel = 0 à 7

for adresse\_pixel in range(0, 8):

# intensité aléatoire entre 0-60 par canaux RVB

# une intensité aléatoire entre 0-255 est trop forte

r = randint(0, 60)

v = randint(0, 60)

b = randint(0, 60)

# Affecte le tuple (r, v, b) au NeoPixel

np[adresse\_pixel] = (r, v, b)

# Envoi des données sur le stick Neopixel

np.show()

sleep(100)

Après avoir connecté votre Micro :bit V2 sur un Shield Grove, brancher le ruban de 30 LEDs en P0/P14.

- Faire fonctionner le programme ci-contre.

- Modifier le programme pour allumer toutes les Leds du ruban.

Défis :

- Modifier le programme afin d’allumer les 2 Leds puis faites les se rejoindre au centre …

- A vous de donner libre court à votre imagination.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  8  9  10  11  12  13  14 |  |  |  | 15  16  18  19  20 |  |  |  |

**Défi 1 : Haut parleur**

**from microbit import \***

**import music**

**while True:**

**if button\_a.is\_pressed():**

**tune = ["g4:4", "g4:4", "g4:4","d#4:3",…**

**music.play(tune)**

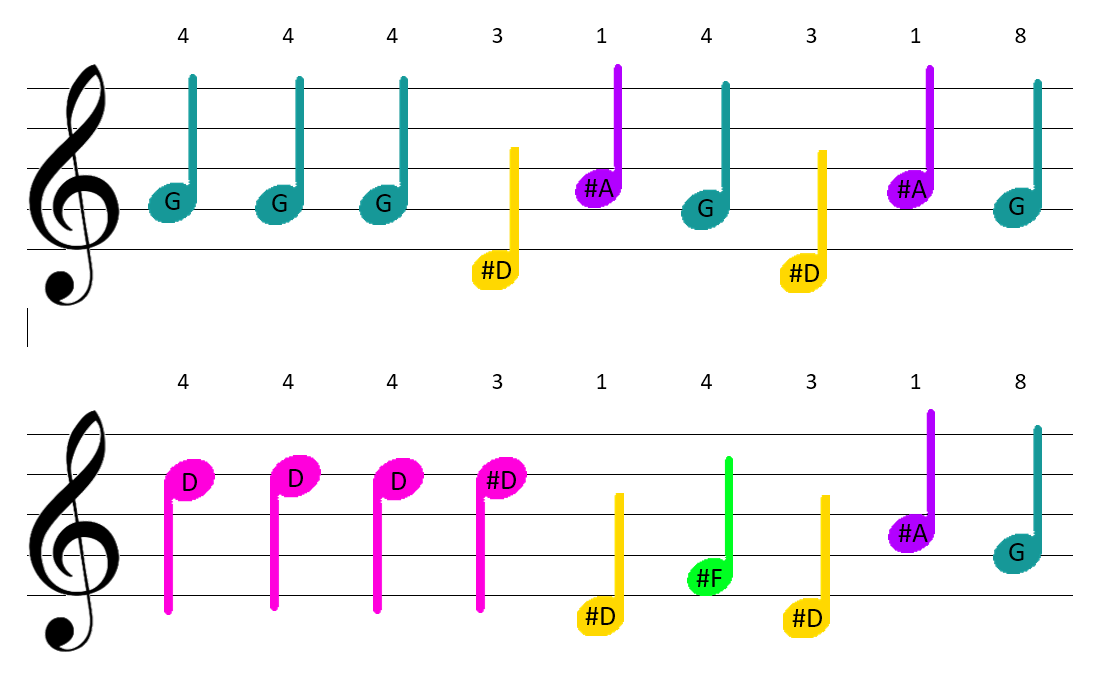
Après avoir connecté votre Micro :bit V2 sur un Shield Grove, brancher le haut parleur

Une**note de musique** est décrite par sa hauteur et sa durée. Nous utiliserons le système anglo-saxon pour décrire la hauteur d’une note. La description de la hauteur contient un caractère entre A et G, et un numéro d’octave.



- Modifier le programme pour jouer la musique de star War

Les premières notes de Star Wars



Défis :

- Jouer « Au claire de lune »

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  8  9  10  11  12  13  14 |  |  |  | 15  16  18  19  20 |  |  |  |