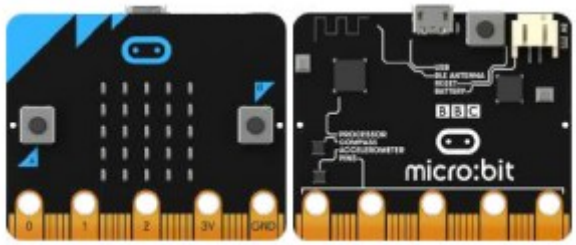
Informatique embarquée et objets connectés Partie1

Activité élève BBC micro:bit et python ,Proposé par :

thomas.basso@ac-polynesie.pf

1. 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 :

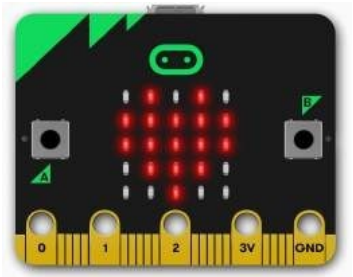
25 LEDs programmables individuellement

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

2. Découverte des fonctionnalités

21 Commandes de base de l'afficheur, matrice de 5x5 LEDs

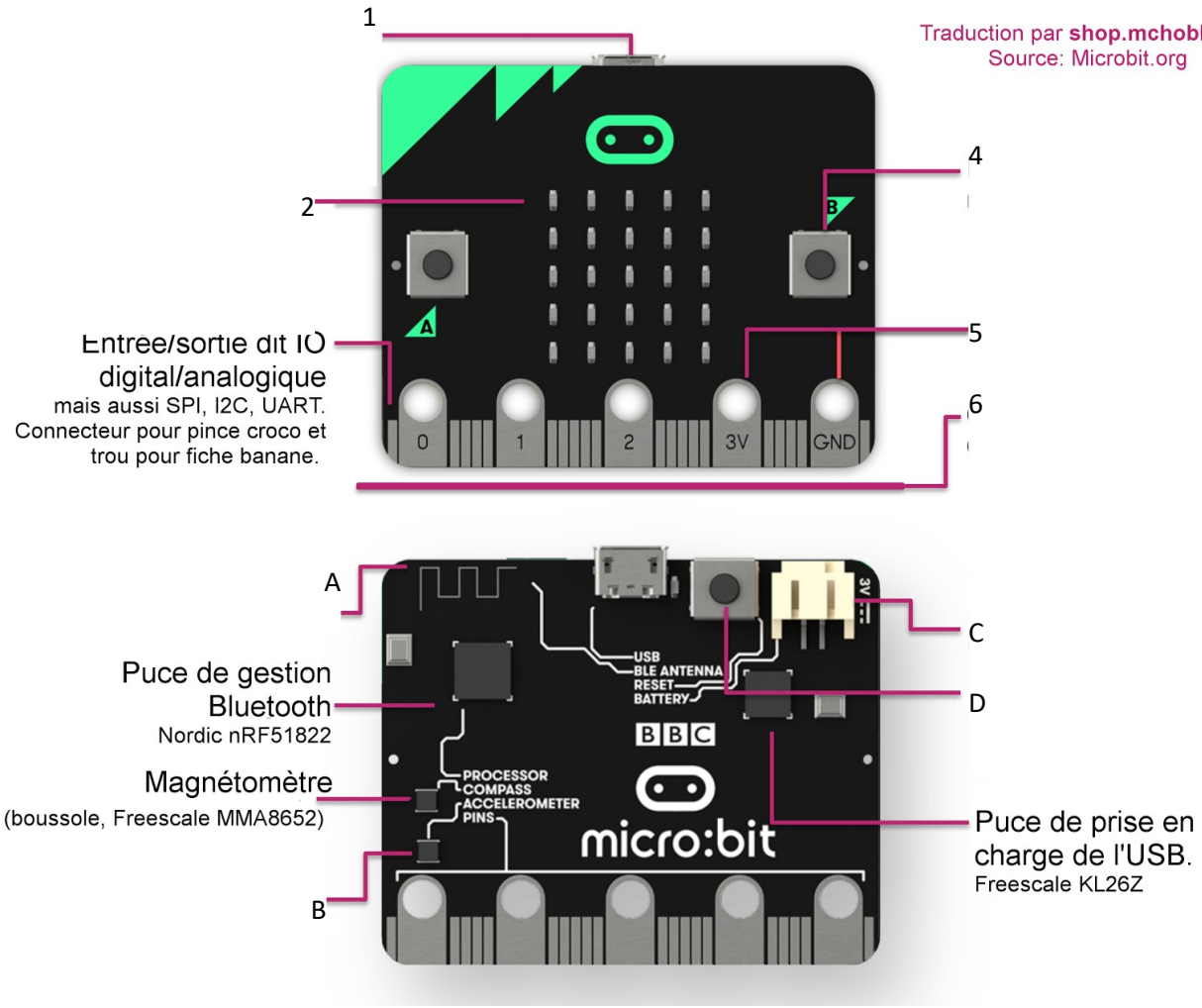
Voir : vidéo explicative (en



anglais)(https://youtu.be/qqBmvHD5bCw)

LED signifie Light Emitting Diode, Diode électroluminescente. La carte micro:bit en dispose de 25, toutes programmables individuellement, ce qui permet d'afficher du texte, des nombres et des images.

Présentation de la carte microbit



Activité : *En observant la carte et en recherchant les informations sur internet, Compléter les tableaux suivants en plaçant les numéros au bon endroit :*

Schéma du haut :

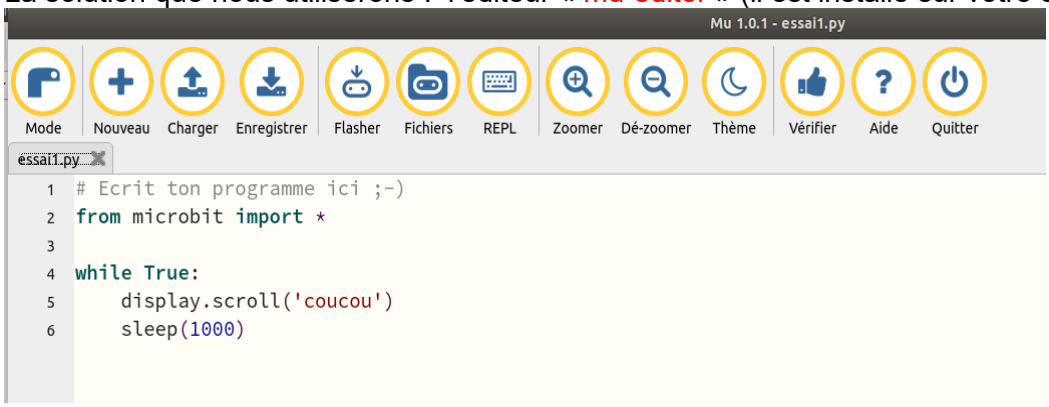
Connecteur microUSB	Boutons utilisateur	Connecteur	Alimentation externe	Matrice d'affichage 5x5 led

Schéma du bas

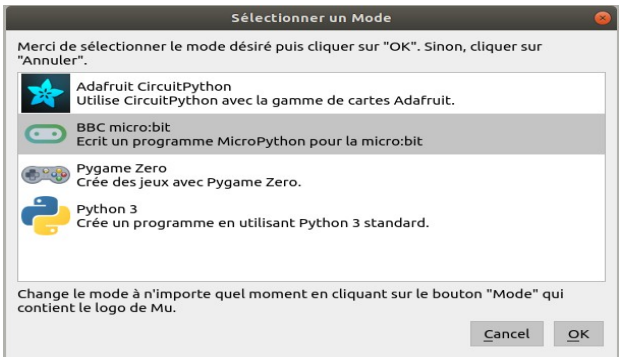
Antenne bluetooth	Bouton reset	Accéléromètre	Connecteur de pile

Logiciel

La version python utilisée pour programmer la micro:bit, appelée microPython, est proche de python3. Elle permet d'utiliser les objets et instructions usuels : entiers, réels, chaînes de caractères, booléens, listes, instructions conditionnelles, boucles itératives et conditionnelles, fonctions, ...
Certaines bibliothèques sont cependant absentes, ces dernières nécessitant notamment une place mémoire trop importante ou l'utilisation de composants (hardware) absents du microcontrôleur.
Des modules font également leur apparition, ils permettent d'utiliser les différents capteurs, actionneurs et la matrice de LEDs.
Comment programmer la carte micro:bit en Python?
La solution que nous utiliserons : l'éditeur « mu-editor » (il est installé sur votre ordinateur).



Avant toute chose il faut choisir l'environnement correspondant à notre carte Micro:bit : cliquez sur « Mode »



Et vous choisissez BBC micro:bit

Le logiciel permet d'écrire son programme (et propose « intelligemment » des fin d'instruction
Le logiciel permet d'envoyer son programme dans la carte (Flasher),
de vérifier la syntaxe de votre programme (Vérifier)
Il possède aussi un mode de visualisation en direct (un peu équivalent au moniteur série d'Arduino) : REPL

Activité : *Recopier le programme ci dessus , le téléverser dans la carte et vérifier que le fonctionnement est correct .*

2.1.1 Afficher un texte "défilant" display.scroll(string, delay=400)

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

Saisir et exécuter le programme suivant :

```
from microbit import *
display.scroll("SNT")
```

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

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.

Saisir et exécuter le programme suivant :

```
from microbit import *
display.scroll("IL ETAIT UNE FOIS...", delay=20)
```

Exercice 2: Le défilement de la phrase du programme ci-dessus est trop rapide pour pouvoir la lire correctement. Modifie la valeur du paramètre delay pour qu'on puisse la lire facilement.

2.1.2 Afficher une "image" display.show(image)

Saisir et exécuter le programme suivant :

```
from microbit import *
display.show(Image.SAD)
```

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

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

Prolongement: essayer plusieurs images intégrées.

Créer sa propre image

Chaque pixel LED sur l'affichage physique peut prendre une parmi dix valeurs. Si un pixel prend la valeur 0 c'est qu'il est éteint. Litéralement, 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).

Saisir et exécuter le programme suivant :

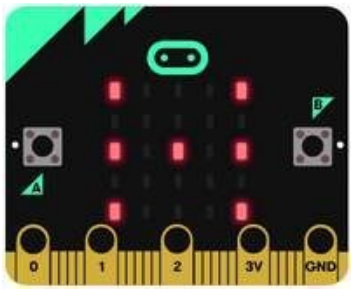
```
from microbit import *

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

display.show(bateau)
```

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.

Copiez votre programme ici :

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



Copiez votre programme ici :

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

```
from microbit import *

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

display.show(bateau)
```

2.1.3 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/).

Exécuter le programme suivant :

```
from microbit import *
display.set_pixel(1, 4, 9)
```

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

Copiez votre programme ici : ✈

22 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.

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

Exécuter le programme ci-dessous:

```
from microbit import *
while True:
    display.set_pixel(2, 2, 9)
    sleep(500)
    display.clear()
    sleep(500)
```

Avec un peu d'aléatoire (voir documentation sur le hasard (https://microbitmicropython.readthedocs.io/fr/latest/tutorials/random.html))

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.

Exécuter le programme ci-dessous:

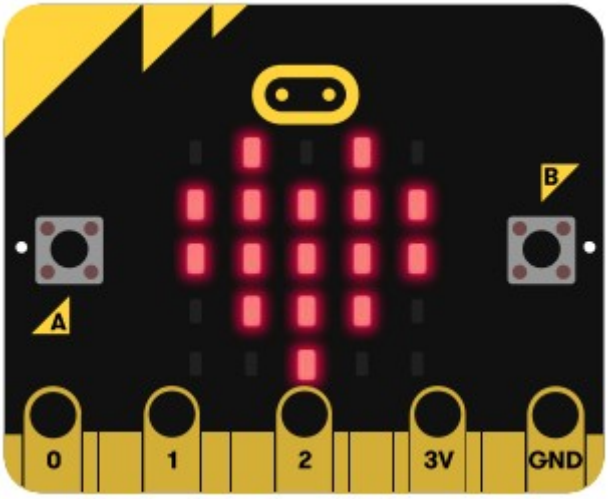
```
from microbit import *
import random
n=random.randint(0,4)
p=random.randint(0,4)
display.set_pixel(n, p, 9)
```

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).

Copiez votre programme ici :

Exercice 8: Ecrire ci-dessous un programme qui fait clignoter un coeur indéfiniment (voir illustration ci-dessous).



Copiez votre programme ici :

Créer une animation

En affichant plusieurs images successives, on peut réaliser une animation.

Exercice 9: Ecrire un programme qui réalise l'animation suivante:

Illustration du rendu recherché : https://makecode.microbit.org/---run?id=_4xi7Ct2DzWXK

Copiez votre programme ici :

23 Boucle for

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

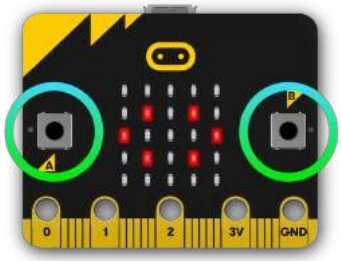
```
from microbit import *
while True:
    for i in range(5):
        display.set_pixel(i,0,9)
        sleep(200)
        display.clear()
```

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.

Copiez votre programme ici :

24 Les entrées boutons A, B et A+B - programmation événementielle

vidéo explicative :(https://youtu.be/t_Qujjd_38o)



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
```

Instructions conditionnelles `if` , `elif` , else

Voici comment se structure une instruction conditionnelle. Selon la situation, il n'est pas forcément nécessaire d'utiliser `elif` ou else .

```
if quelque chose est vrai (``True``):
    # fais un truc
elif autre chose est vrai (``True``):
    # fais un autre truc
else:
    # fais encore autre chose.
```

Exercice 11 : *Pierre feuille ciseaux!* 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.

Voici pour exemple une illustration du résultat recherché: dans le simulateur suivant, clique sur le bouton A, le bouton B et le bouton qui simule la pression simultanée des boutons A et B.
https://makecode.microbit.org/---run?id=_LYvH1C4LK2CT'

Complétez le programme ci dessous:

```
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(500)
    elif button_a.is_pressed():
        display.show(pierre)
        sleep(500)
    elif button_b.is_pressed():
        display.show(feuille)
        sleep(500)
    display.clear()
    sleep(100)
```