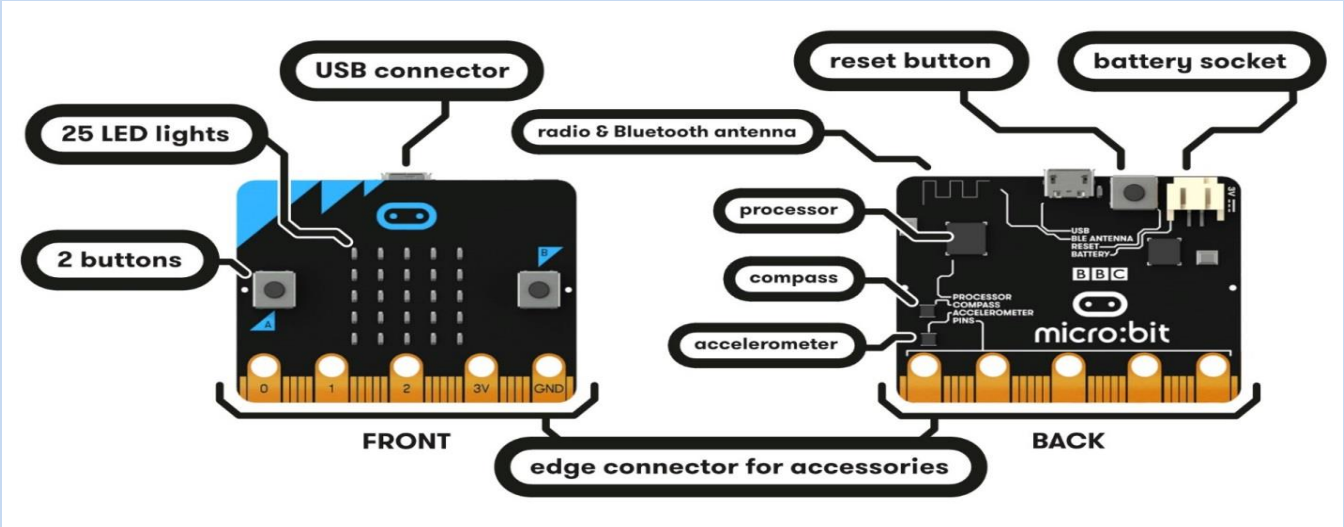
Nom : Classe:

TP maison Micro:Bit 3H

1. Introduction

Les schémas ci-dessous résument de façon visuelle l'ensemble des ressources matérielles disponibles sur la carte BBC Micro:Bit :



On va utiliser un simulateur de la carte…. Vu que… …..

https://create.withcode.uk/

Écrire le code MicroPython puis cliquer sur

pour l'exécuter.

Micro:Bit

Activité 1. Mon 1er programme

- Toujours commencer un programme par la ligne ***from microbit import ****
- Ecrire le programme suivant **(Attention à respecter les minuscules/majuscules et les espaces)**

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

mycode.py

```
1  from microbit import *
2  display.show(Image.HAPPY)
3
4
```

Que se passe-t-il ?

- On peut changer l'image : https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html

On veut rendre la micro :bit triste, écrire le nouveau programme.

Il y a une matrice de 25 LED (diodes électroluminescentes) sur la face avant de la carte.
Plusieurs commandes d'affichage sont disponibles en microPython :

- `display.show()` : affiche le caractère ou l'image choisi(e) entre parenthèses
- `display.scroll(string, delay=400)` : affiche une chaîne de caractères (*string*, du texte) en défilement avec une certaine vitesse (*delay*, plus le délai est grand, moins le texte défilera rapidement)
- `display.set_pixel(x, y, val)` : allume le pixel de coordonnées *x* et *y* (de 0 à 4 en abscisse et ordonnée) d'une intensité *val* (entre 0 et 9)
- `sleep()` : provoque la pause de la carte pendant un nombre défini de millisecondes choisi entre parenthèses
- `display.clear()` : efface l'affichage en cours

- On peut faire défiler du texte avec ***display.show("Hello")***
- On aimerait attendre 2 secondes avant d'afficher le Smiley . On va utiliser l'instruction ***sleep(2000)***

Ecrire le programme dans https://create.withcode.uk/

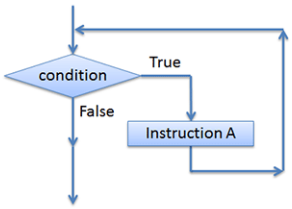
- On va utiliser une boucle infinie ***(Attention à bien décaler ce qui doit être dans la boucle)***
- https://www.youtube.com/watch?v=aAdF7crqKnk

```
while True:
    display.show("Bonjour")
```

Boucle `while`

Syntaxe :

```
while condition:
    Instruction A
```



Exemple de programme :

- Expliquer en quelques mots la boucle while.

Activité 2 : COMPTE À REBOURS

- https://www.youtube.com/watch?v=x4cVoamH6EE

```
from microbit import *
for i in range(5):
  display.scroll (i)
```

- Faîtes-le fonctionner sur le simulateur en ligne: https://create.withcode.uk/
- Combien de fois se répète la boucle for?

- Modifiez le programme ci-dessus afin qu'il compte en boucle de 0 jusqu'à 9

Copiez le code de votre programme ci-dessous

- Expliquer en quelques mots la boucle for.

Activité 3 : **Les entrées boutons A, B et A+B – programmation événementielle**

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 un code 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.
- button_a.get_presses() : renvoie le nombre de fois où le bouton a été appuyé depuis le démarrage ou la dernière fois que la méthode a été appelée et réinitialise ce total à zéro.

Instructions conditionnelles if , elif , else

https://www.youtube.com/watch?v=5K2lwktygt4

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.
```

Étudiez le code suivant :

```
from microbit import *
compteur = 0
while True:
        if button_a.was_pressed():
            display.show(str(compteur))
             sleep(1000)
            display.clear()
```

Que fait le code?

- Vérifiez votre réponse en le testant sur la carte ou le simulateur en ligne.
  https://create.withcode.uk/

Modifiez le programme ci-dessus pour

- que le nombre augmente de 1 à chaque appui sur le bouton A.
- que le nombre revienne à 0 lorsqu'on appuie sur le bouton B.
- que le nombre reste affiché en permanence

Tester le. https://create.withcode.uk/

Activité 3 : 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 lum = display.read_light_level() retourne un entier compris entre 0 et 255 sur la variable lum, représentant le niveau de lumière.
Compléter un programme ci-dessous qui affiche une lune si on baisse la luminosité (en recouvrant la carte avec sa main par exemple) et un soleil sinon.

```
from microbit import *
soleil = Image("90909:"
"09990:"
"99999:"
"09990:"
"90909:")
lune = Image("00999:"
"09990:"
"09900:"
"09990:"
"00999:")
while True:
```

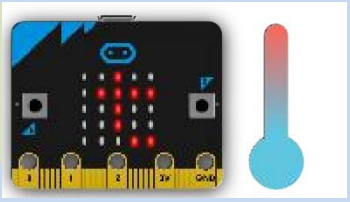
Tester le. https://create.withcode.uk/

- Expliquer en quelques mots l'instruction if.

Activité 4 : Capteur de 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 (c'est un processeur ARM à grande efficacité), sa température est une bonne approximation de la température ambiante.
L'instruction `temperature()` renvoie la température de la carte micro:bit en degrés Celsius.

```
from microbit import *
while True:
     sleep(3000)
     display.scroll(temperature())
```



Tester le. https://create.withcode.uk/

Réaliser un thermomètre de réfrigérateur.

- Il indique la température toutes les 5 secondes.
- Si la température est comprise entre 0°c et 6°c , on affiche un smiley pendant 1 seconde.
- Si la température est supérieur à 6°c , on affiche smiley angry.

Tester le. https://create.withcode.uk/