Initiation à la programmation de la carte BBC Micro:bit

Objectif :

1 Conventions

Consignes sur le travail à effectuer Données Ressources

2 Prérequis

- Programmation de base en python : variables, opérateurs,fonctions, boucles,

3 Nouvelles compétences à acquérir

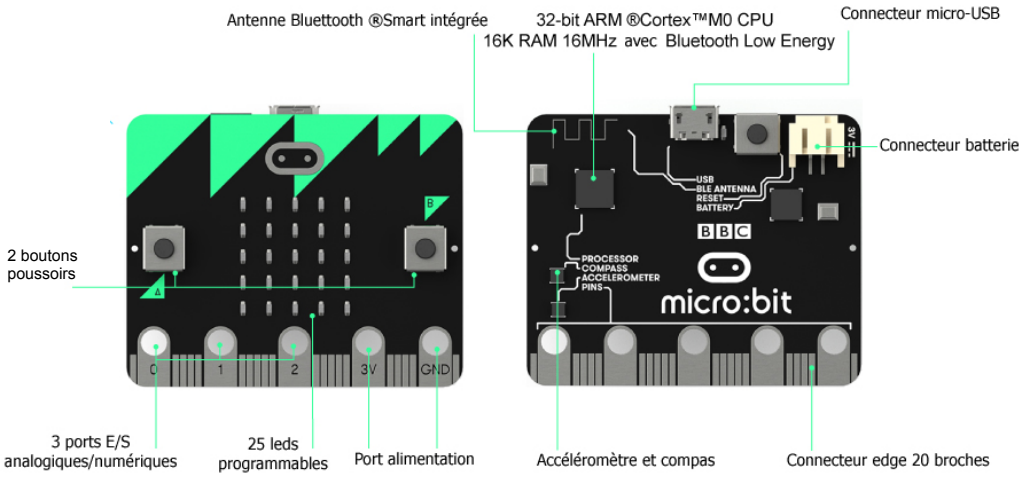
- Utilisation du logiciel mu-editor
- Utilisation basique de la carte micro-bit associée à la carte d'extension (shield) Grove

4 Présentation de la carte micro-bit

La carte micro:bit est une carte électronique programmable, ayant des capteurs et actionneurs intégrés.

La carte peut fonctionner de manière autonome ou elle peut être connectée en USB à un ordinateur. Elle peut alimenter des capteurs en 3,3V.

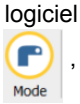
La carte peut-être programmée dans un langage dérivé de Python, mais très proche : le Micropython.

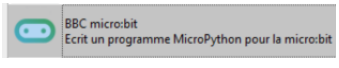


5 Utilisation de la carte

1. Premier programme

- Connecter **l**a carte sur unport USB de l'ordinateur
- Lancer le logiciel mu-editor en cliquant sur 

qui se trouve sur le bureau
- Lorsque le logiciel est ouvert,
cliquer sur 

, puis selectionner 

et valider par OK

- Saisir le programme en cliquant sur 

puis écrire le code suivant dans la zone de saisie

```
from microbit import *
display.show("Bonjour")
```

- Avec 

enregistrer ce programme sous le nom **prog1.py**
- Transférer le programme dans la carte en cliquant sur 

- Pendant le transfert, la led jaune au dos de la carte clignote. Le programme s'exécute automatiquement dès la fin du transfert.

2. **Utilisation des boutons A et B**

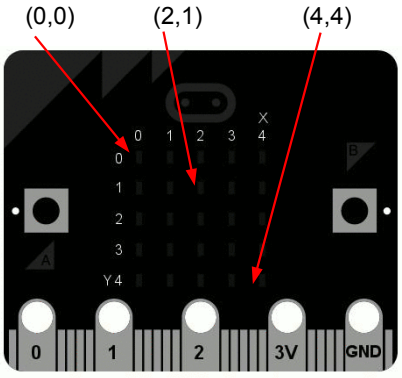
Pour interagir avec la carte, on dispose (entre autres) de 2 boutons A et B.
Pour que le programme détecte l'appui sur les boutons, il faut tester/surveiller l'état des boutons en permanence.
Les fonctions python qui retournent l'état des boutons **doivent donc être dans une boucle While** comme ci-dessous.

```
from microbit import *
# On cree une boucle infinie
while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

- Tester ce programme
- Modifier ce programme pour afficher :
  "A" si on appuie sur A
  "B" si on appuie sur B
  et "-" si on appuie sur aucun bouton

3. **La matrice à leds**

- **Les leds**
  chaque led est repérée par sa position (x,y)



La commande des leds se fait avec "**`display.set_pixel`**(*x, y, value*)" où
"x" (entier entre 0 et 4) indique le numéro de la colonne
"y" (entier entre 0 et 4) indique le numéro de la ligne
"value" (entier entre 0 et 9) fixe la luminosité

- Tester le programme suivant et indiquer ci-dessous les leds qui s'allument en faisant un cercle + ou – gros en fonction de la luminosité

```
from microbit import *

display.set_pixel(0, 0, 1)
display.set_pixel(1, 1, 3)
display.set_pixel(2, 2, 5)
display.set_pixel(1, 3, 7)
display.set_pixel(0, 4, 9)
```

*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*

Pour simplifier les programmes, il est fréquent de commander les leds en utilisant les boucles 'for'
Par exemple pour allumer une ligne de leds, on fait :

```
from microbit import *

# allumer les leds de la 1ere ligne
for x in range(5):
    display.set_pixel(x,0,9)
```

Faire un test après chaque modification !

- Modifier le programme pour allumer la dernière ligne au lieu de la 1ère.
- Modifier le programme pour allumer la colonne du milieu.
- Modifier le programme pour allumer les leds d'une diagonale
- Ajouter une commande "display.set_pixel" pour allumer les 2 diagonales

- **Effet chenillard**
  Pour créer cet effet, on allume 2 leds pendant 1/10 s puis
  - on éteint la première
  - on allume la suivante

  L'exemple suivant crée un chenillard sur la première ligne

```
from microbit import *

while True:
    for x in range(4):
        display.set_pixel(x, 0, 9)
        display.set_pixel(x+1, 0, 9)
        sleep(100)
        display.set_pixel(x, 0, 0)

display.clear()
```

- Tester l'exemple et analyser son fonctionnement
- Ajouter une boucle for pour que la "chenille" poursuive son chemin en descendant la colonne de droite puis tester
- Ajouter 2 autres boucles for pour que la chenille tourne sur l'extérieur de la matrice

- **Les images**
  - Images prédéfinies

    La liste des images disponibles se trouve ici :
    https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html

    Pour afficher une image, on utilise la méthode display.show().

    Par exemple pour afficher une fléche vers le haut, il suffit de faire :

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

    On peut afficher une "suite" d'images, pour créer une animation

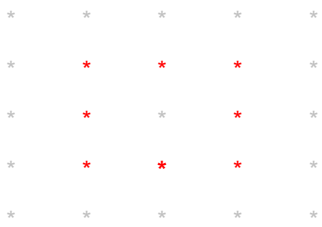
```
from microbit import *
display.show(Image.ALL_ARROWS, loop=True, delay=500)
```

  - Images personnelles

    Pour définir une imgage, il suffit d'indiquer ligne par ligne, le niveau de brillance de chaque led
    Par exemple pour afficher un carré sur les leds périphériques de la matrice

```
from microbit import *
Carre = Image('99999:90009:90009:90009:99999:')
display.show(Carre)
```

- **Tester** le code ci-dessus
- **Modifier** pour obtenir l'image ci-dessous
- **Faire constater** le fonctionnement



4. Les entrées sorties en mode "digital"

Pour connecter des capteurs et des actionneurs, la carte possède 19 broches repérées pin0, pin1, … pin16 et pin19, pin20.

En mode "digital" les broches n'ont que 2 états logiques possibles .

État logique 0 → tension=0v *État logique1 → tension=3,3V*

Pour lire l'état d'une broche utilisée en entrée logique (un bouton poussoir par exemple), on utilise la méthode **read_digital()**.

Exemple :

```
from microbit import *

while True:
    if pin0.read_digital():
        display.show("1")
    else:
        display.show("0")
```

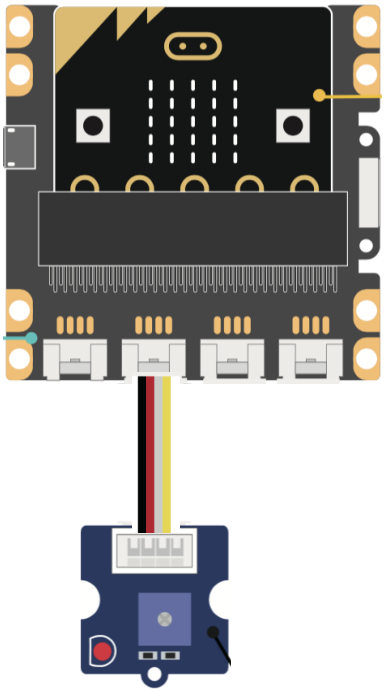
* **Exécuter** ce programme

En utilisant un fil muni de pinces crocodiles,
Indiquer
* la valeur s'affiche quand on relie la broche 0 à la broche 3V →
* la valeur s'affiche quand la broche 0 n'est pas branchée →

Pour commander une broche en sortie logique (allumer/éteindre une led, faire tourner/arrêter un moteur, …) on utilise la méthode **write_digital()**

Expérimentation : Pour brancher des actionneurs sur les broches, il est conseillé d'utiliser une carte d'interface. On utilisera ici le module grove

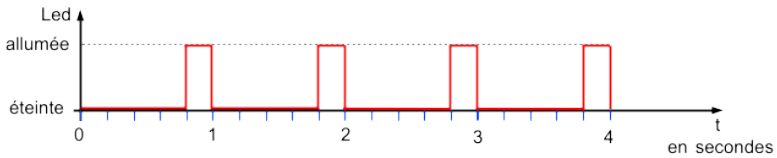
Connecter une led au module comme indiqué ci-dessous



```
from microbit import *

while True:
    pin0.write_digital(1)
    sleep(500)
    pin0.write_digital(0)
    sleep(500)
```

Tester le code ci-dessus
Modifier le code pour obtenir le fonctionnement décrit par le chronogramme ci-dessous
Faire constater le fonctionnement



5. **Les entrées sorties en mode "analogique"**

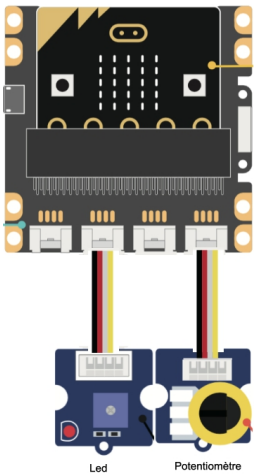
Sur certaines broches, on peut brancher des capteurs (température, luminosité,..) qui délivrent une tension, comprise entre 0V et 3,3V, proportionnelle à la grandeur mesurée.

La méthode **read_ananalog()**, retourne une valeur numérique comprise entre 0 et 1023 et proportionnelle à la tension appliquée sur la broche.

On peut aussi faire varier la tension moyenne sur les broches auquelles on a branché un actionneur (Led, moteur, haut parleur, ..)

Dans ce cas on utilise la méthode **write_analog(x)**. La tension sur la broche sera comprise entre 0 et 3,3V quand varie x entre 0 et 1023.

Ajouter un potentiomètre au module comme indiqué ci-dessous




```
from microbit import *

while True:
    consigne = pin2.read_analog()
    pin0.write_analog(consigne)
```

- **Tester** le code ci-dessus
- I**ndiquer** ce qu'il se passe lorsqu'on tourne le potentiomètre
- **Expliquer** le fonctionnement du code

6. Les capteurs intégrés à la carte

La carte micro-bit comporte :

- Un capteur qui fournit la température de la carte, différente de la température ambiante car placé à côté du micro-processeur
- Un accéléromètre 3 axes qui permet par exemple de mesurer l'inclinaison de la carte
- Un compas (bousssole) qui , après initialisation fournit l'orientation de la carte.

Remarque :

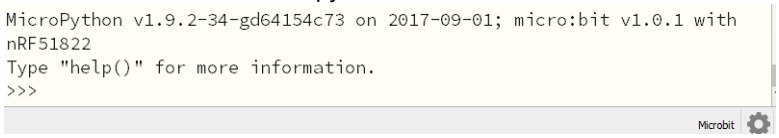
Ces capteurs fournissent une information numérique à plusieurs chiffres difficilement visualisables sur la carte elle-même.
On traitera donc les exemples dans le paragraphe suivant qui traite de la mise en œuvre de la console

7. La console REPL de mu-editor

Lorsqu'on ouvre une console (terminal) REPL en cliquant sur l'icône 

, l'exécution du programme dans la carte microbit est arrêtée
Le système vous "donne la main" pour

- saisir des commandes micro python



Par exemple

```
>>> display.set_pixel(0,0,9)
>>> display.set_pixel(0,0,0)
>>> temperature()
23
>>>
```

Microbit

- Pour visualiser le résultat de la fonction **print()** d'un programme python

```
from microbit import *

while True:
    # on affiche la temperature de la carte
    mesure_T = temperature()
    print("Temperature=", mesure_T)
    # on peut faire aussi directement
    #print("Temperature=", temperature())
    # on affiche l'inclinaison suivant l'axe x
    print("Inclinaison x=", accelerometer.get_x())
    sleep(500)
```

- **Quitter** le mode console en cliquant sur l'icône REPL
- **Saisir** le et **flashe**r le code ci-dessus
- **Rouvrir** la console REPL
- **Relancer** l'exécution du programme en faisant CTRL+D
  En inclinant la carte vers la droite ou la gauche, vous devez observer les variations de l'inclinaison

```
Inclinaison x= 400
Temperature= 23
Inclinaison x= 652
Temperature= 23
Inclinaison x= 640
Temperature= 23
Inclinaison x= 632
Temperature= 23
```

8. Le mode graphique

Ce mode qu'on active en cliquant sur 

permet le tracé d'une variable en utilisant la fonction **print((variable1,)** ou plusieurs en faisant print((variable1,variable2,...variable-nn,))

A noter qu'on peut utiliser simultanément la console REPL et le mode graphique

```
from microbit import *

while True:
    Inclinaison = accelerometer.get_x() * 90 / 1024
    if button_a.is_pressed():
        # on affiche l'inclinaison suivant l'axe x
        print("Inclinaison x=", Inclinaison)
    else:
        # on trace la courbe
        print((Inclinaison,))
    sleep(200)
```

Avec le code ci-dessus, on affiche la valeur de l'inclinaison dans la console REPL si on appuie sur le bouton A, sinon on trace la courbe.