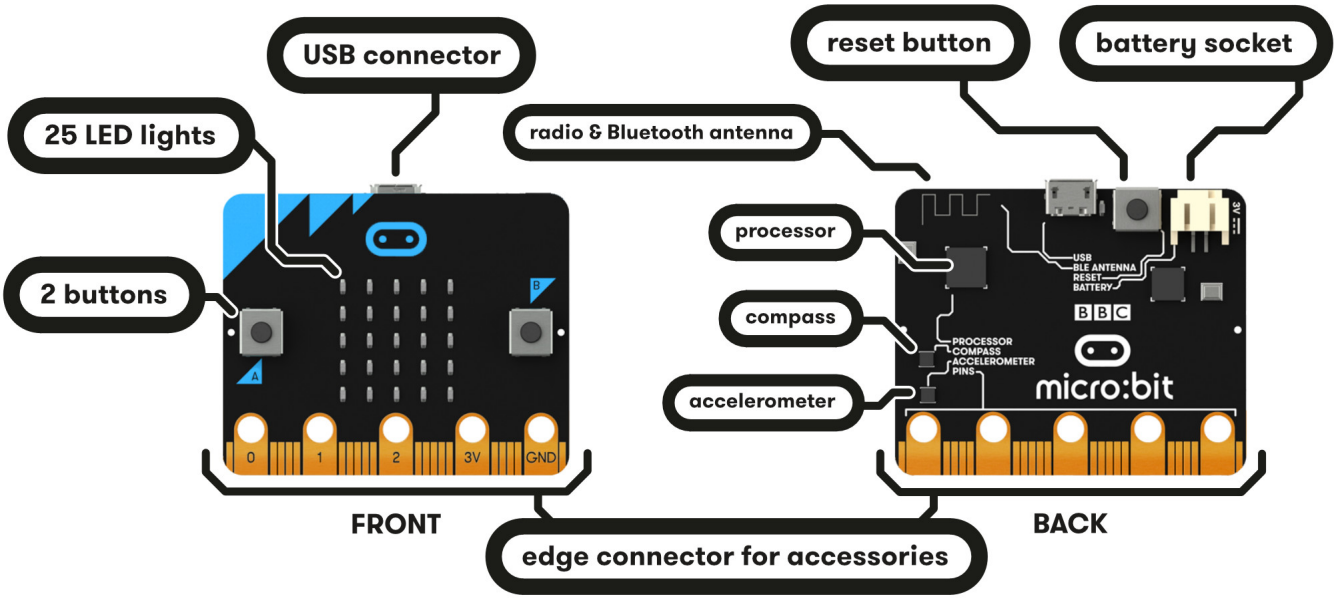
Ce document est inspiré d'une ressource présentée pendant la formation SNT 2019 des enseignants de l'Académie de Lyon et placée sous licence CC BY-NC-SA 4.0 et du tutoriel en ligne https://microbit.org/fr/guide/python/.

1 Présentation de la carte micro:bit

Définition 1

Un **système informatique embarqué** collecte des informations du monde réel à l'aide de **capteurs**, les traite dans un **micorprocesseur** puis agit sur le monde réel par le biais d' **actionneurs**. Le traitement des informations est contrôlé par un programme qui peut interagir avec l'homme à travers une **Interface Homme Machine**.

La carte micro:bit éditée par la BBC, est un **nano-ordinateur** qui peut équiper un **système informatique embarqué**. Elle est munie d'un processeur ARM et de plusieurs capteurs et interfaces de connexion. Le guide de présentation en ligne est disponible sur https://microbit.org/fr/guide/.



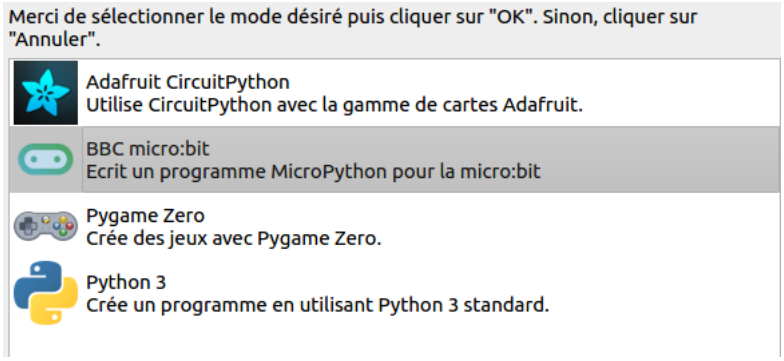
Source : https://microbit.org/fr/guide/features/

Nous utiliserons uniquement la carte en la connectant à un ordinateur avec le câble USB fourni qui assure la liaison de communication et l'alimentation. Si on veut intégrer la carte dans un système embarqué, il est possible de la connecter à une alimentation externe par piles.

Lorsque la communication entre l'ordinateur et la carte échoue, on peut essayer de la redémarrer avec le bouton reset situé au verso.

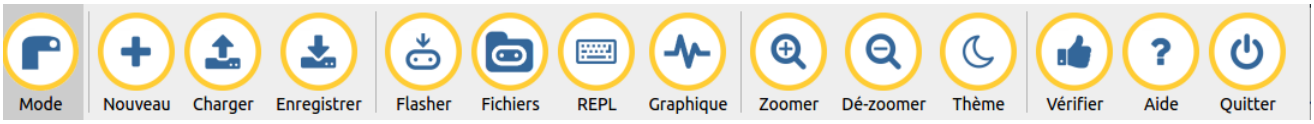
Exercice 1 *Premiers programmes*

1. Lancer l'environnement de programmation Mu depuis le bureau et sélectionner le mode BBC micro:bit.



Nous programmerons la carte avec le langage Python et son module microbit.

2. **a.** Sélectionner Nouveau dans la barre de menu pour créer un nouveau programme puis enregistrer le fichier sous le nom programme1.py dans un dossier pertinent de son espace personnel sur le réseau pédagogique.



b. Saisir dans l'éditeur de texte le code ci-dessous en respectant bien l'indentation c'est-à-dire l'espacement par rapport à la marge de gauche. Enregistrer le programme avec la combinaison de touches CTRL + S.

Programme 1

```
1 from microbit import *
2
3 display.scroll("Hello World")
```

c. Pour transférer le programme sur la carte, cliquer sur Flasher. Lors de chaque téléversement la mémoire Flash contenant le programme exécuté par la carte est réinitialisée.

d. Décrire l'effet du programme sur la carte. Une interaction est-elle possible ?

.....

Préciser le rôle de chaque instruction.

.....

.....

3. **a.** Créer un autre programme programme2.py enregistré dans le même dossier que le précédent avec le code source ci-dessous.

Programme 2

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

b. Transférer ce programme sur la carte.
Décrire l'effet du programme sur la carte. Une interaction est-elle possible ?

.....

Préciser le rôle de chaque instruction.

.....

.....

.....

D'autres images sont disponibles, voir la liste en ligne.

2 Boucle et capture d'événements

Méthode

Un algorithme de contrôle fréquent sur un système informatique embarqué consiste en une boucle infinie où s'enchaînent capture d'événements par les émetteurs, traitement puis action par les actionneurs.

```
Initialiser les actionneurs à leur position de départ
Tant que Vrai
    Lire les informations des capteurs
    Traiter ces informations
    Calculer des informations sur les actionneurs
    Transmettre ces informations aux actionneurs
```

Exercice 2 *Première boucle*

1. Créer un programme `programme3.py` enregistré dans le même dossier que les précédents avec le code source ci-dessous en respectant bien l'indentation.

Programme 3

```
from microbit import *

#début de la boucle
while running_time() < 5000:
    display.show(Image.ASLEEP)
#fin de la boucle
display.show(Image.SURPRISED)
sleep(5000) #attente de 5000 millisecondes
display.clear()
```

2. Transférer ce programme sur la carte.

Décrire l'effet du programme sur la carte. Une interaction est-elle possible ?

.....

Préciser le rôle de chaque instruction. Identifier capteur et actionneur.

.....

.....

.....

.....

.....

.....

Exercice 3 *Boucle avec structure conditionnelle*

1. **a.** Créer un programme `programme4.py` enregistré dans le même dossier que les précédents avec le code source ci-dessous en respectant bien l'indentation.

Programme 4

```
from microbit import *

#boucle infinie
while True:
    #Structure conditionnelle avec 2 choix
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

b. Transférer ce programme sur la carte.
Décrire l'effet du programme sur la carte. Une interaction est-elle possible ?

.....

Préciser le rôle de chaque instruction. Identifier capteur et actionneur.

.....

.....

.....

.....

.....

.....

2. **a.** Créer un programme `programme5.py` enregistré dans le même dossier que les précédents avec le code source ci-dessous en respectant bien l'indentation.

Programme 5

```
1  from microbit import *
2
3  #boucle infinie
4  while True:
5      #Structure conditionnelle avec 3 choix
6      if button_a.is_pressed():
7          display.show(Image.HAPPY)
8      elif  button_b.is_pressed():
9          display.show(Image.ANGRY)
10     else:
11         display.show(Image.SAD)
```

b. Transférer ce programme sur la carte.
Décrire l'effet du programme sur la carte. Une interaction est-elle possible ?

.....

Préciser le rôle de chaque instruction. Identifier capteur et actionneur. .....

.....

.....

Exercice 4 *Boucle avec test*

1. Créer un programme `programme6.py` enregistré dans le même dossier que les précédents avec le code source ci-dessous en respectant bien l'indentation.

Programme 6

```
1 from microbit import *
2
3 #boucle avec test
4 while not button_a.is_pressed():
5     display.show(Image.SAD)
6 #fin de boucle
7 display.show(Image.HAPPY)
```

2. Transférer ce programme sur la carte.

Décrire l'effet du programme sur la carte. Une interaction est-elle possible ?

.....

Préciser le rôle de chaque instruction. Identifier capteur et actionneur.

.....

.....

.....

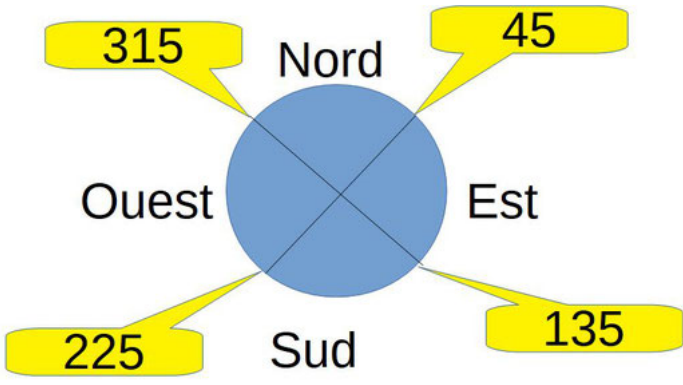
.....

Exercice 5 *Boussole*

La carte `micro:bit` dispose d'un capteur de champ électromagnétique accessible par l'objet `compass`.

1. Avant toute utilisation du `compass`, il faut le calibrer avec `compass.calibrate()` : un message défilant sur l'écran nous invite à incliner la carte jusqu'à ce que les 25 diodes soient allumées.

`compass.heading()` retourne l'angle entre le vecteur du champ magnétique mesuré et le Nord magnétique comme 0. La mesure en degrés, est un entier compris entre 0 et 360. Par défaut, le champ magnétique terrestre est mesuré.
Pour simuler une boussole indiquant les quatre point cardinaux avec la carte `micro:bit`, on découpe l'intervalle [0 ; 360] en 4 intervalles d'amplitude 90 degrés.



https://courstechnocollege.jimdo.com

2. Créer un programme `programme7.py` enregistré dans le même dossier que les précédents avec le code source ci-dessous en respectant bien l'indentation.

Programme 7

```
from microbit import *

#Calibrage du compas
compass.calibrate()

#boucle
while True:
    angle = compass.heading()
    if  315 <= angle or angle <= 45:
        display.show('N')
    elif 45 < angle and angle <= 135:
        display.show('E')
    #attente d'une seconde
    sleep(1000)
```

3. Préciser le rôle de chaque instruction. Identifier capteur et actionneur.

.....

.....

.....

.....

4. Compléter le programme pour qu'il affiche aussi les directions Sud et Ouest et qu'un appui sur le bouton A provoque une sortie de la boucle suivie d'un effacement de l'écran.

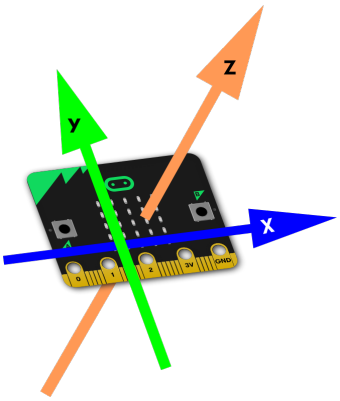
5. Imaginer un système embarqué utilisant le capteur boussole.

.....

.....

Exercice 6 *Accéléromètre et température*

L'accéléromètre est un capteur mesurant l'accélération de la carte `micro:bit`. Il détecte ses mouvements et son inclinaison. La mesure fournie comporte trois composantes suivant les axes d'un repère de l'espace, chacune prend des valeurs entières entre -2000 et 2000 g où g est l'unité d'accélération représentant approximativement la pesanteur terrestre.



Source : https://microbit.org/fr/guide/features/

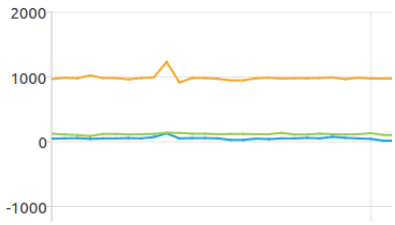
- `accelerometer.get_x()`, `accelerometer.get_y()`, et `accelerometer.get_z()` donnent les valeurs des composantes de l'accélération suivant chaque axe.
- `accelerometer.get_values()` retourne le triplet de composantes.

1. **a.** Créer un programme `programme8.py` enregistré dans le même dossier que les précédents avec le code source ci-dessous en respectant bien l'indentation, puis transférer ce programme sur la carte.

Programme 8

```
1 from microbit import *
2
3 while True:
4     sleep(20)
5     print(accelerometer.get_values())
```

b. Sélectionner `Graphique` dans la barre de menu de `Mu`. Les valeurs des composantes sont représentées en orange pour `z`, en bleu pour `x` et en vert pour `z`. Incliner la carte pour faire varier les valeurs des composantes. Déterminer les inclinaisons de la carte qui permettent d'obtenir les triplets `(x,y,z)=(0,0,2000)`, `(x,y,z)=(0,0,-2000`, `(x,y,z)=(0,2000,0)` et `(x,y,z)=(2000,0,0)`.



2. Créer un programme `programme9.py` enregistré dans le même dossier que les précédents.

Le programme doit répondre aux spécifications suivantes :

- Dans une boucle :
  - capturer la valeur absolue de la composante en `x` de l'accélération dans une variable avec `gx = accelerometer.get_x();`
  - faire défiler la température capturée avec `display.scroll(temperature())` si `gx > 100` ou sinon effacer l'écran avec `display.clear()`.
- Sortir de la boucle si le bouton A est pressé puis effacer l'écran.

La température capturée par `temperature()` est celle du processeur mais comme il chauffe peu c'est une bonne approximation de celle de l'environnement.