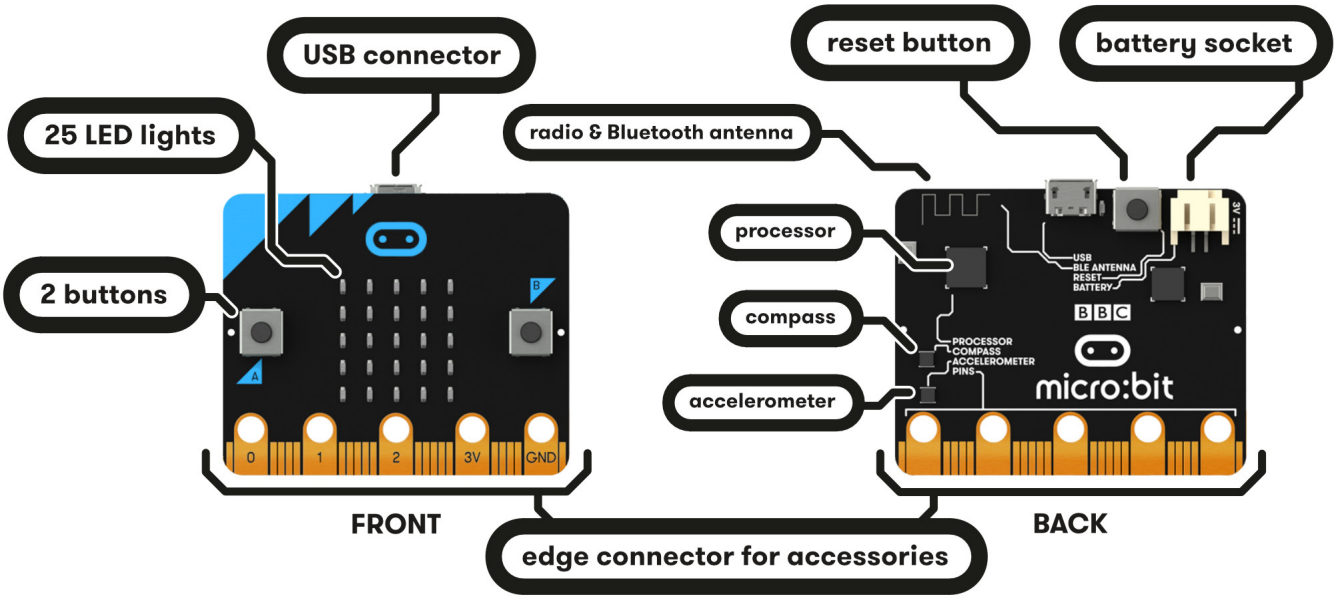
Ce document est inspiré d'une ressource présentée pendant la formation SNT 2019 des enseignants de l'Académie de Lyon et placée sous licence CC BY-NC-SA 4.0 et du tutoriel en ligne https://microbit.org/fr/guide/python/.

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 **microprocesseur** 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/.



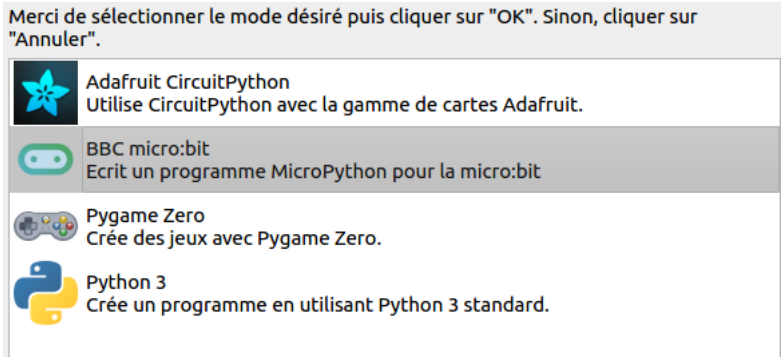
Source : https://microbit.org/fr/guide/features/

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.

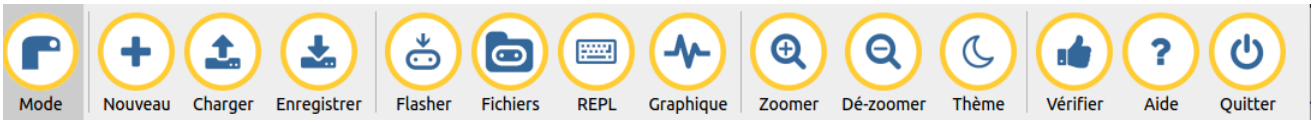
Exercice 1 *Premiers programmes*

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

```
from microbit import *

#boucle infinie
while True:
    #Structure conditionnelle avec 3 choix
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    elif  button_b.is_pressed():
        display.show(Image.ANGRY)
    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. .....

.....

.....

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.

.....

.....

.....

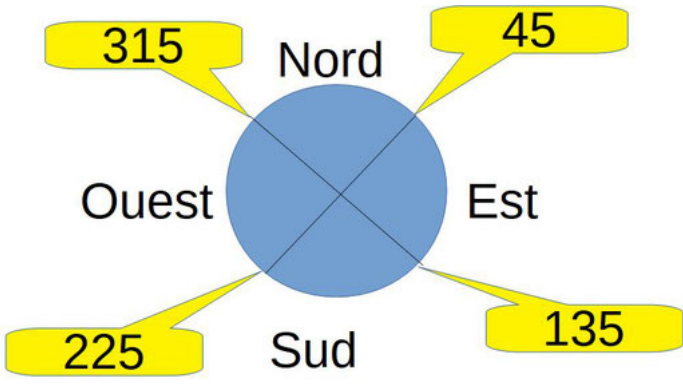
.....

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.

.....

.....

3 Manipulation d'images

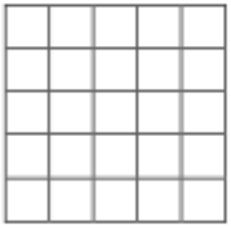
Définition 2
Une Interface Homme Machine ou IHM est un ensemble de dispositifs physique (boutons, curseurs) et logiciels (interface graphique) permettant d'échanger des informations avec une machine.

Exercice 6 *Créez votre propre image!*

Chacun des 25 pixels du micro :bit peut prendre une valeur entre 0 et 9 : 0 pour éteint, 9 pour allumé avec la puissance maximale, les valeurs comprises entre 1 et 8 sont des valeurs de luminosité intermédiaires. Par exemple, saisir le programme ci-dessous dans l'éditeur Mu Python puis le transférer sur la carte micro :bit. Devinez ce que peut représenter l'image affichée.

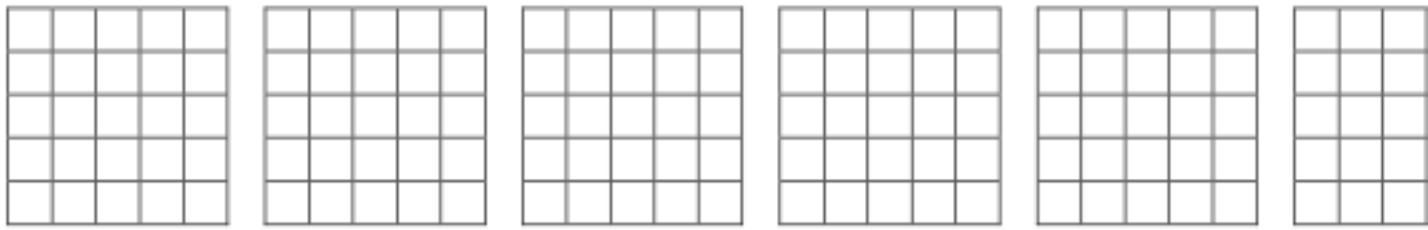
```
from microbit import *
boat = Image("05050:"
"05050:"
"05050:"
"99999:"
"09990")
display.show(boat)
```

Dessinez à l'aide de la grille ci-dessous votre propre image et testez-là!



Exercice 7 *Créez votre propre animation*

1. Pour cela, comme un dessin animé, créez une liste d'images qui vont s'afficher successivement. Il faut en créer au moins 2.



Par exemple, si vous avez créé 6 bateaux nommés `boat1, boat2, boat3, boat4, boat5, boat6`, vous pouvez alors afficher l'animation à l'aide des instructions suivantes. La liste ordonnée d'images est créée et les images défilent toutes les 200 millisecondes.

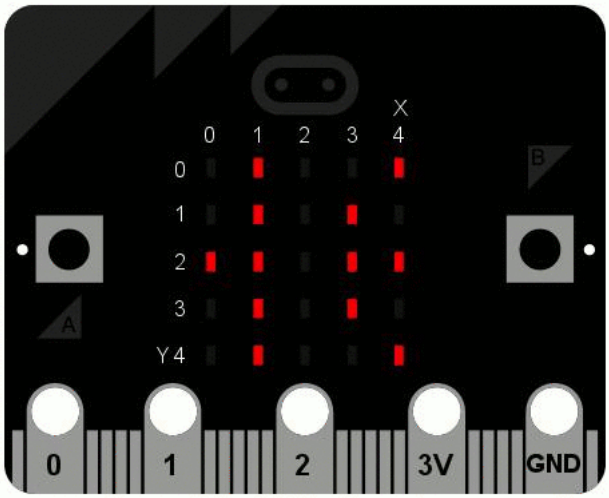
```
all_boats = [boat1, boat2, boat3, boat4, boat5, boat6]
display.show(all_boats, delay=200)
```

2. Modifiez le programme pour qu'il affiche l'animation quand on appuie sur le bouton A et l'animation « inverse » quand on appuie sur le bouton B.

Exercice 8 *Manipulation de diodes et boucle `for`*

L'écran de la carte `micro:bit` est une grille ou matrice constituée de 25 diodes électroluminescentes ou LED. Chaque diode est repérée par ses coordonnées `x` et `y` variant entre 0 et 4, comme ci-dessous. De plus une diode peut être émettre un signal lumineux d'intensité variable entre 0, diode éteinte, et 9, intensité maximale.

En `Python`, l'instruction `display.set_pixel(x, y, v)` permet d'allumer le diode de coordonnées `x` et `y` avec l'intensité `v`.



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

1. Ouvrir le logiciel `Mu` en mode `micro:bit`, saisir le programme ci-dessous dans l'éditeur avant de l'enregistrer dans un dossier pertinent sous le nom `exo1-microbit.py` puis de le transférer sur la carte.

Programme 8

```
1 from microbit import *
2
3 for x in range(5):
4     display.set_pixel(x,x,9)
```

2. Décrire l'effet du programme sur la carte.

.....

Préciser le rôle de chaque instruction.

.....

.....

.....

3. Modifier le programme pour qu'il allume tous les pixels de coordonnées `(x,0)` avec $0 \leqslant x \leqslant 4$.

4. Modifier le programme pour que les diagonales de la matrice de diodes s'allument en forme de croix.

4 Simuler le hasard

Exercice 9

1. La fonction `randint(a, b)` du module `random` de `Python` permet de simuler le choix d'un entier aléatoire compris entre deux entiers `a <= b`. On commence par l'importer avec l'instruction `from random import randint`.

   Compléter le programme ci-dessous dans l'éditeur Mu Python puis le transférer sur la carte micro :bit pour que la simulation d'un lancer de dé à 6 faces soit affichée pendant deux secondes sur l'écran dès qu'on appuie sur le bouton A.

```
from microbit import *
from random import randint

while True:
    #à  compléter
    display.clear()
```

2. Adapter le programme précédent pour que le résultat de chaque lancer soit affiché jusqu'à l'obtention du premier 6 puis affiche l'image `HAPPY` pendant deux secondes avant que la partie se termine.

```
from microbit import *
from random import randint

#à  compléter
display.show(Image.HAPPY)
sleep(2000)
```

3. Modifier le dernier programme pour que le nombre de lancers avant l'obtention du premier 6 soit affiché à la fin pendant deux secondes (après l'image `HAPPY`).

Exercice 10

1. Saisir le programme ci-dessous dans l'éditeur Mu Python puis le transférer sur la carte micro :bit. On note A un appui sur le bouton A, B un appui sur le bouton B et AB un appui simultané sur A et B.

   - Décrire la succession d'affichages obtenus pour la séquence d'appuis A - A - B - A - AB
   - Décrire la succession d'affichages obtenus pour la séquence d'appuis A - B - B - B - AB
   - Combien de séquences de six appuis (sans compter l'appui simultané AB) permettent d'obtenir l'affichage -2.

```
from microbit import *
from random import randint

validation = False
choix = 0
while not validation:
    boutonA = button_a.was_pressed()
    boutonB = button_b.was_pressed()
    if boutonA and boutonB:
        validation = True
    elif boutonA:
        choix = choix + 1
    elif boutonB:
        choix = choix - 1
    display.show(choix)
    sleep(1000)
display.show(Image.HAPPY)
```

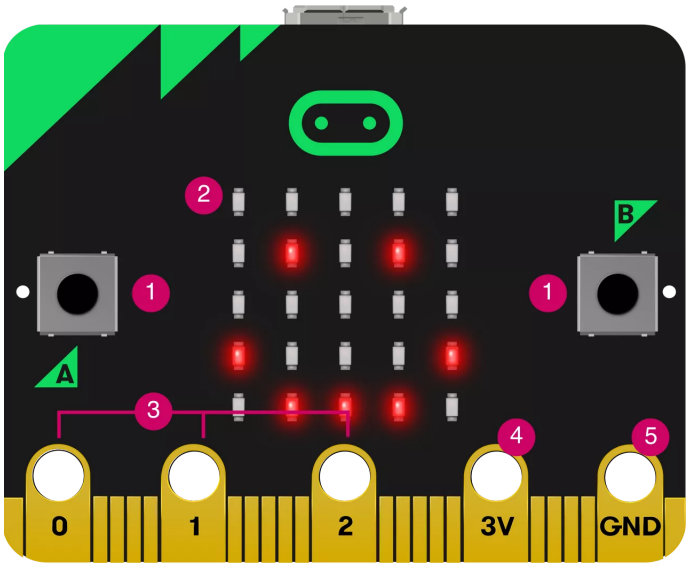
La variable `validation` est de type *booléen*, elle peut prendre deux valeurs `True` ou `False`. Les valeurs *booléennes* sont utilisées dans les opérations logiques comme la **conjonction** (ET) et la **disjonction** (OU).

Exercice 11

À l'aide des programmes précédents, écrire un programme qui simule le jeu de devinette décrit ci-dessous :

- l'ordinateur choisit un nombre au hasard entre 1 et 9 ;
- tant que le joueur n'a pas deviné ce nombre, il peut proposer un nombre à l'aides boutons A et B : un appui sur A permet d'incrémenter de 1 la proposition précédente (0 avant que la partie commence) et un appui sur B de la décrémenter de 1, un double appui sur A et B valide la proposition ;
- le jeu s'arrête lorsque le joueur a deviné le nombre secret, l'image `HAPPY` est affichée ainsi que le nombre de coups pour deviner le nombre secret.

Quel nombre minimal de coups nous permet de deviner le secret à coup sûr ? Comparer cette méthode avec une recherche ordonnée si on doit deviner un nombre entre 0 et 100000.


2
B
1
1
A
3
4
5
0
1
2
3V
GND