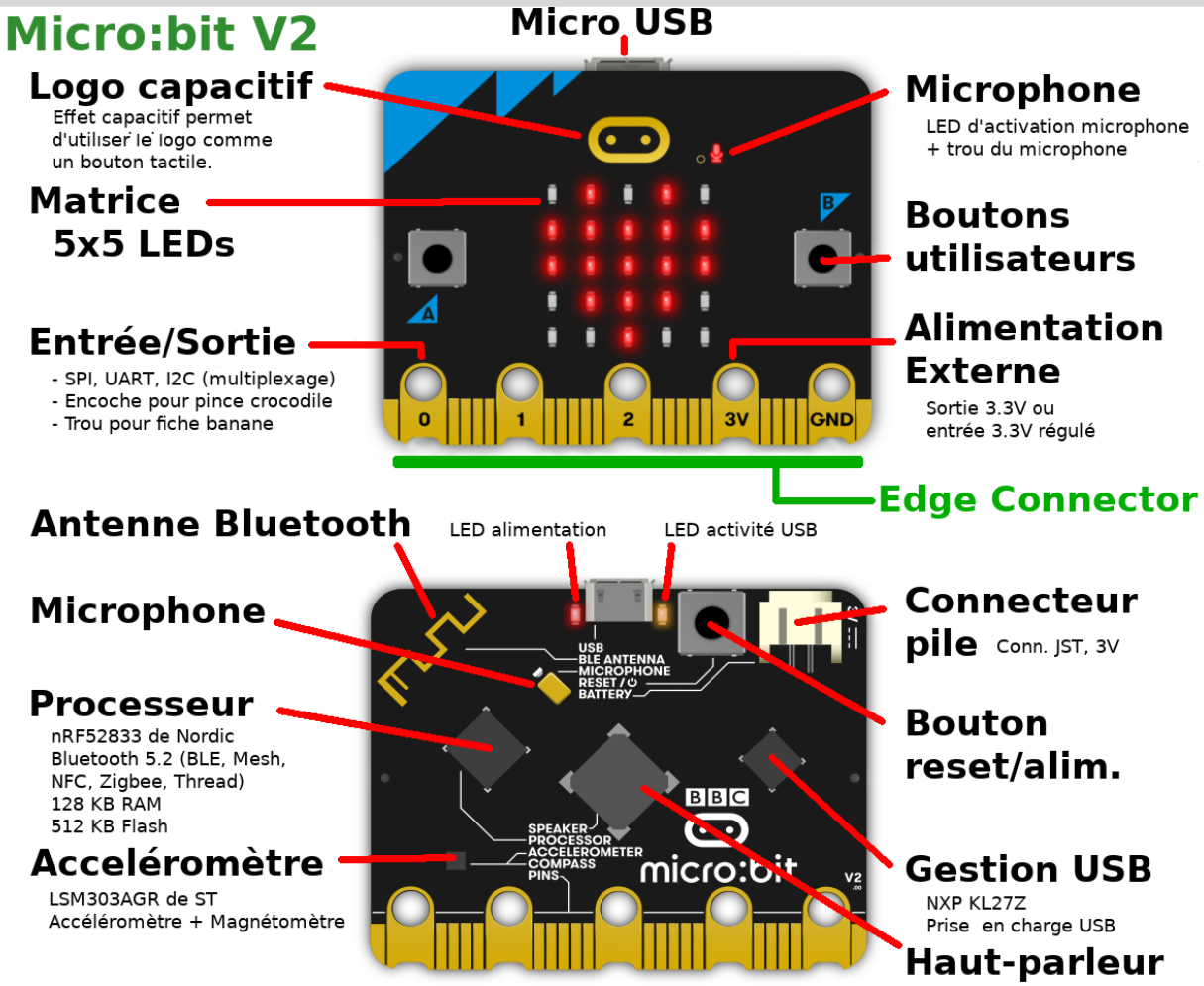
# Découverte carte micro:bit

## 1 Présentation de la carte micro:bit

**Définition 1**

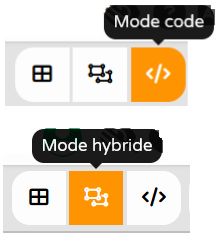
Un **système informatique embarqué** collecte des informations du monde réel à l'aide de **capteurs**, les traite dans un **microprocesseur** puis agit sur le monde réel par le biais d'**actionneurs**. Le traitement des informations est contrôlé par un programme qui peut interagir avec l'homme à travers 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 et de plusieurs capteurs et interfaces de connexion.
Nous utiliserons 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. Avec un explorateur (chrome, edge, …) ouvrir le site vittascience.
Cliquer sur programmer puis choisir l'interface de la carte micro:bit.
Nous programmerons la carte avec le langage Python (mode code) néanmoins il sera possible de programmer par bloc (mode hybride).

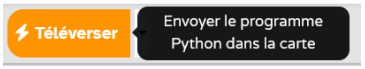



2. Taper ce programme en respectant l'indentation c'est-à-dire l'espacement par rapport à la marge de gauche.

**Programme 1**

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

3. Pour transférer le programme sur la carte, cliquer sur Téléverser.




Lors de chaque téléversement la mémoire Flash contenant le programme exécuté par la carte est réinitialisée.

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

.....

5. Remplacer le programme précédent par celui-ci :

**Programme 2**

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

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

..... .....

7. Facultatif : Il est possible d'afficher d'autres images déjà programmées.

# 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.** Remplacer le programme précédent par celui-ci en respectant l'indentation c'est-à-dire l'espacement par rapport à la marge de gauche.

**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.** Téléverser le programme puis décrire l'effet du programme sur la carte.
Une interaction est-elle possible ?

.....

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

.....

.....

.....

.....

.....

.....

## Exercice 3 *Boucle avec structure conditionnelle*

**1.** Remplacer le programme précédent par celui-ci en respectant l'indentation c'est-à-dire l'espacement par rapport à la marge de gauche.

**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)
```

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

.....

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

.....

.....

.....

.....

.....

.....

**4.** Remplacer le programme précédent par celui-ci en respectant l'indentation c'est-à-dire l'espacement par rapport à la marge de gauche.

**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)
```

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

.....

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

.....

.....

.....

.....

.....

.....

### Exercice 4 *Boucle avec test*

1. Remplacer le programme précédent par celui-ci en respectant l'indentation c'est-à-dire l'espacement par rapport à la marge de gauche.

**Programme 6**

```
from microbit import *

#boucle avec test
while not button_a.is_pressed():
    display.show(Image.SAD)
#fin de boucle
display.show(Image.HAPPY)
```

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

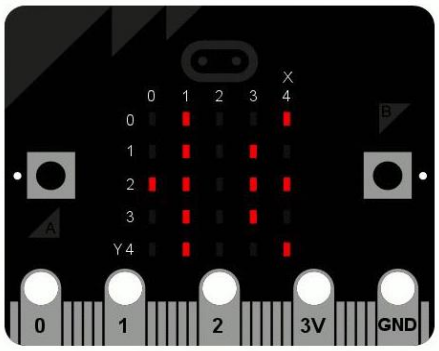
3. Préciser le rôle de chaque instruction. Identifier capteur et actionneur.
.....
.....
.....
.....

**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.

## 3 Manipuler des images

### Exercice 5 *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 la diode de coordonnées x et y avec l'intensité v.



1. Taper ce programme en respectant l'indentation c'est-à-dire l'espacement par rapport à la marge de gauche.

**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. Une interaction est-elle possible ?

.....

3. Préciser le rôle de chaque instruction.

.....

.....

.....

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

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

# 4 Simuler le hasard

## Exercice 6

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

   Compléter le programme ci-dessous dans l'éditeur de vittascience puis le transférer sur la carte micro:bit pour que la simulation d'un résultat 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 7

1. Saisir le programme ci-dessous dans l'éditeur vittascience 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 logo un appui sur le logo.

```
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 pin_logo.is_touched():
        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).

- Décrire la succession d'affichages obtenus pour la séquence d'appuis A - A - B - A – logo

- Décrire la succession d'affichages obtenus pour la séquence d'appuis A - B - B - B – logo

## Exercice 8

A 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 des 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 permet de la décrémenter de 1, on appuie sur le logo pour valider la proposition. Si sa proposition est fausse l'image SAD est affichée.
- 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.