Périphériques d'entrée et de sortie
Interface Homme-Machine (IHM)

REFERENCE AU PROGRAMME

Niveau : 1ère N.S.I.

Contenus	Capacités attendues	Commentaires
Architectures matérielles et systèmes d'exploitation		
Périphériques d'entrée et de sortie Interface HommeMachine (IHM)	Identifier le rôle des capteurs et actionneurs. Réaliser par programmation une IHM répondant à un cahier des charges donné.	Les activités peuvent être développées sur des objets connectés, des systèmes embarqués ou robots.
Langages et programmation		
Constructions élémentaires	Mettre en évidence un corpus de constructions élémentaires.	Séquences, affectation, conditionnelles, boucles bornées, boucles non bornées, appels de fonction.
Utilisation de bibliothèques	Utiliser la documentation d'une bibliothèque.	Aucune connaissance exhaustive d'une bibliothèque particulière n'est exigible.

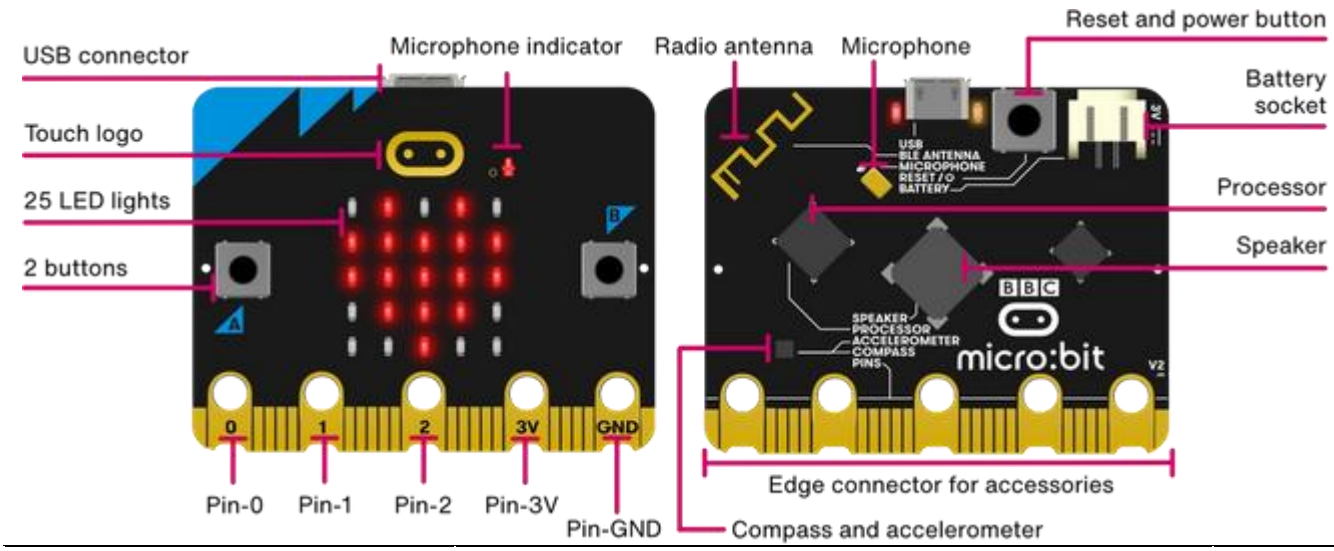
MISE EN SITUATION.

Un **microcontrôleur** (en notation abrégée **µc**, ou **uc** ou encore **MCU** en anglais) est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

Par rapport à des systèmes électroniques à base de microprocesseurs et autres composants séparés, les microcontrôleurs permettent de diminuer la taille, la consommation électrique et le coût des produits. Ils ont ainsi permis de démocratiser l'utilisation de l'informatique dans un grand nombre de produits et de procédés.

Source wikipedia

Voici les différentes fonctionnalités natives de la carte ***micro:bit*** en votre possession,.



TRAVAIL DEMANDE

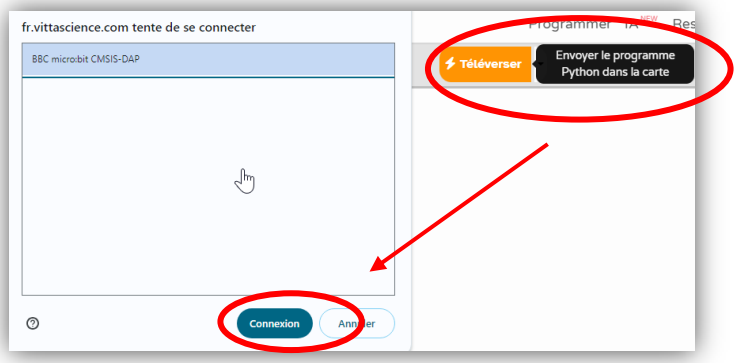
1. PREMIERE CONNEXION ET RECOMMANDATIONS

Ouvrir, en fonction de votre choix pédagogique, un logiciel de programmation possédant l'interpréteur `MicroPython` (`BBC micro:bit`). Le site Vittascience est utilisé dans ce document.

Passer en mode code : 

1.1. Connecter votre carte à l'ordinateur *via* le câble USB
1.2. Tester la connexion entre la carte et l'ordinateur

Téléversement d'un programme :



Pour visualiser les éventuels messages d'erreur ou les données transmises par la carte, ouvrir la console `micropython` :

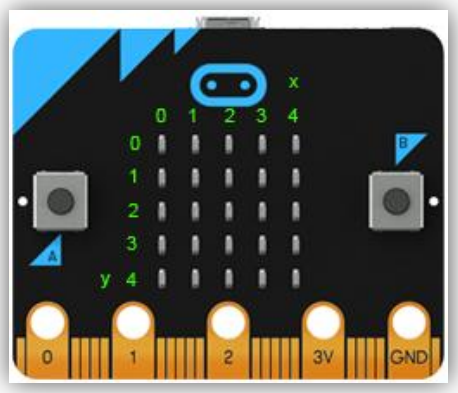
Pour afficher un texte dans la console (exemple la température), il suffit d'utiliser la fonction `print()` comme en Python.

2. PROGRAMMATION D'INTERFACE DE SORTIE

Nous allons commencer par programmer les leds présentes sur la carte. Il existe plusieurs méthodes pour programmer la matrice des leds.

Allumage led par led :

Il est possible d'éteindre toutes les leds à l'aide de l'instruction : `display.clear()`



Les 25 leds sont disposées en matrice 5 x 5 et peuvent être adressées de manière individuelle à l'aide d'un système de coordonnées (x, y) avec l'intensité `n`, un entier compris entre `0` et `9` (0 = éteint et 9 = luminosité maximale).
L'instruction est :

```
display.set_pixel(x, y, n)
```

Allumage de la matrice :

- ✓ Pour allumer des leds spécifiques, il faut créer une variable afin d'enregistrer une instanciation de l'objet `Image` puis appeler la méthode `show()` de l'objet `display` avec cette variable en paramètre.
- ✓ L'objet `Image` prend en paramètre une chaine de type :
  `'99999:00900:00900:00900:99900'`.
- ✓ Chaque élément séparé par "`:`" correspond physiquement à une ligne de leds.
- ✓ L'intensité lumineuse de chacune des leds est un entier compris entre `0` et `9` (0 = éteint et 9 = luminosité maximale).

Exemple :

```
from microbit import *

while True:
  mon_image  =Image('99999:00900:00900:00900:99900')
  display.show(mon_image)
```

Allumage d'un texte déroulant :

- Pour écrire un texte à l'aide des leds, appeler la méthode `show()` de l'objet `display` avec le texte à afficher en paramètre.

```
display.show('Bonjour')
```

2.1. Algorithme du programme à implémenter et à téléverser dans la carte :

```
Afficher le message "ON"
Afficher un carré
Effacer l'affichage
Afficher la led centrale (x = 2, y = 2)
```

On constate que le fonctionnement n'est pas celui attendu.

Le passage d'une instruction à une autre est trop rapide pour être visible à l'œil humain. Il faut donc introduire des pauses entre chaque affichage.

L'instruction pour réaliser une pause est : `sleep(ms)`. Le paramètre `ms` est un entier permettant de régler le temps de pause en millisecondes.

2.2. Modifier votre programme pour implémenter l'algorithme précédent et le visualiser sur la carte

2.3. Écrire un programme permettant d'augmenter (de 0 à 9) l'intensité lumineuse de la led centrale, puis de la réduire.

Faites valider votre programmation par le professeur.

3. PROGRAMMATION D'INTERFACE D'ENTREE

Nous allons maintenant programmer une des interfaces d'entrée : les boutons poussoirs.

La méthode `is_pressed()` appliquée à l'instanciation de l'objet `button` renvoie un booléen qui vaut `True` si le bouton poussoir est appuyé, `False` sinon.

Exemple :
`button_a.is_pressed()` renvoie `True` si le bouton est appuyé `False` sinon.
Même chose pour le bouton b : `button_b.is_pressed()`

Algorithme du programme à implémenter et à téléverser dans la carte :

```
Si le bouton "a" est appuyé :
   • Afficher le message "ON" sur les leds (utiliser la méthode show() de
     l'objet display
   • Afficher en permanence les trois leds centrales (utiliser la méthode
     show() de l'objet display

Si le bouton b est appuyé :
   • Afficher le message "OFF" sur les leds
   • Eteindre toutes les leds (display.clear())
```

3.1. Écrire le programme, le tester sur la carte `micro:bit`.

Faites valider votre programmation par le professeur.

4. TEMPERATURE

Autre périphérique d'entrée intégré, le thermomètre.

Pour relever la température il suffit d'appeler la fonction `temperature()`.

4.1. Écrire un programme permettant d'afficher la température ambiante sur la matrice à leds.

5. ACCELEROMETRE

Nous souhaitons maintenant obtenir les valeurs d'accélérations sur les trois axes x, y et z.

Pour obtenir la valeur de l'accélération sur l'axe x, utiliser la méthode `get_x()` sur l'objet `accelerometer`.

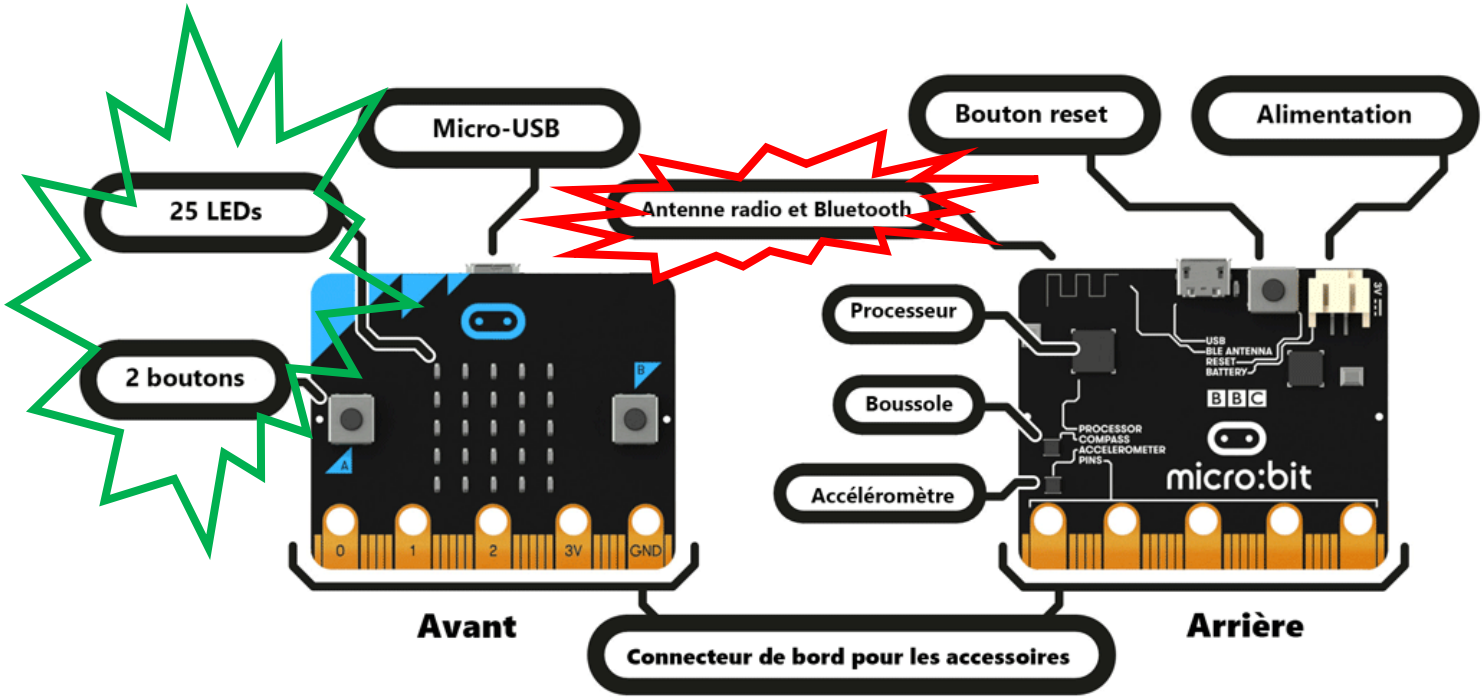
Exemple : `ax = accelerometer.get_x()` renvoie la valeur de l'accélération sur l'axe x.

5.1. Écrire un programme qui permet d'afficher les accélérations sur les trois axes.
5.2. Visualiser les accélérations en mode graphique.

Rappel : pour réaliser des affichage dans la console ou en mode graphique, il faut utiliser la fonction `print()` comme en python

6. COMMUNICATION RADIO

Maintenant que vous savez utiliser les boutons poussoirs et les Leds, nous allons apprendre à faire communiquer deux cartes entre elles.



Un premier message par les ondes

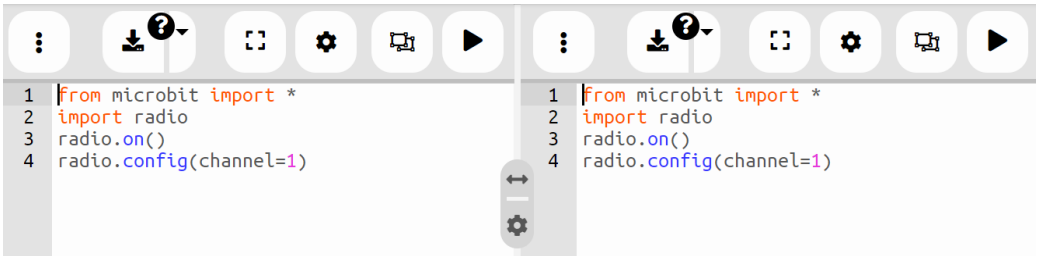
Pour cela, il faut en premier lieu importer et activer le module radio. L'instruction est la suivante :

```
from microbit import *
import radio
radio.on()
```

Préciser ensuite le canal de communication (1 à 100) pour éviter les interférences avec d'autres cartes :

```
radio.config(channel = 1)
```

Finalement, le début du programme pour les deux cartes doit être le même :



Pour envoyer un message sous forme :

- ✓ d'une chaîne de caractères, utiliser sur la carte émettrice : `radio.send("message")`
- ✓ d'un nombre utiliser l'instruction : `radio.send(nombre)`

Pour recevoir le message, utiliser la procédure suivante sur la carte réceptrice : `radio.receive()`

6.1. Transcrire ces deux algorithmes en langage PYTHON et programmer les cartes pour les faire communiquer entre elles.

```
# Carte 1 (émetteur)
Tant que Vrai:
    Si "Appui sur bouton A", alors:
            Envoyer un message (de votre choix)
```

```
# Carte 2 (récepteur)
Tant que Vrai:
   Enregistrer le message reçu dans une variable (par exemple "incoming").
   Si la variable est active (if incoming:), alors:
            Afficher le message sur la matrice à Led
```

7. GRANDS JEUX

7.1. Le plus rapide

Le premier qui appuie sur un bouton affiche le message `NO` sur la carte de son adversaire et `YES` sur la sienne.

7.1.1. Écrire un programme qui permet de savoir qui appuie en premier.

Algorithmes :

```
Carte 1 (émetteur) et 2 (récepteur)
Tant que Vrai:
  Eteindre la matrice à LEDs
  Enregistrer le message reçu dans une variable (par exemple
"opponent").
  Si "Appui sur bouton A", alors:
    Envoyer un message (1)
    Afficher l'image YES sur la matrice à Led
    Faire une pause d'une seconde
  Si la variable est égale à 1, alors:
    Afficher l'image NO sur la matrice à Led
    Faire une pause d'une seconde
```

Variante : Le premier qui secoue trois fois sa carte affiche le message `NO` sur la carte de son adversaire et `YES` sur la sienne.

7.1.2. Modifier le programme pour qu'après trois secousses, les images s'affichent.

```
nbSecousses = 0
while True:
    if accelerometer.was_gesture("shake"):
        nbSecousses +=1
        display.show(str(nbSecousses))
    if nbSecousses == 3:
```

7.2. Le plus chanceux, Chifoumi !

7.2.1. Écrire un programme : Pierre, feuille, ciseaux en suivant les indications suivantes :

Reprendre le programme précédent.
Programmer les images comme indiqué ci-contre.
Envoyer aléatoirement l'une des trois images.
Vérifier qui a gagné !

```
from microbit import *
from random import randint

pierre = Image("00900:"
               "09990:"
               "99599:"
               "09990:"
               "00900")
feuille = Image("99900:"
                "90090:"
                "90009:"
                "90009:"
                "99999")
ciseaux = Image("96009:"
                "69090:"
                "00900:"
                "69090:"
                "96009")
```

```
if nbSecousses == 3:
    choix = randint(1,3)
```