(Re)Découverte de la carte micro:bit

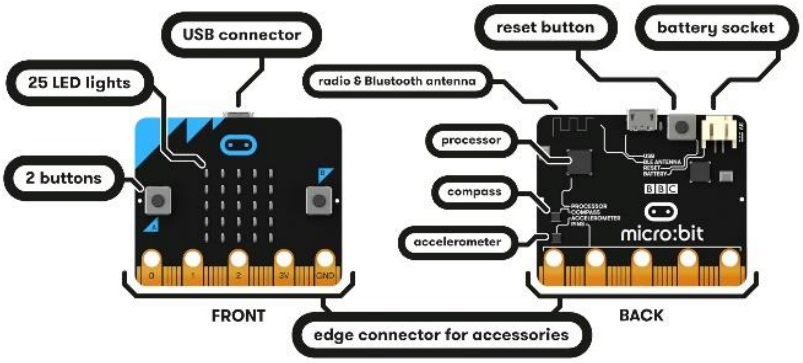
1. La carte micro:bit

La carte micro:bit est un micro-ordinateur de poche programmable. L'un des langages qu'elle comprend est le langage Python, la version utilisée est appelée MicroPython.

1.1. Les constituants

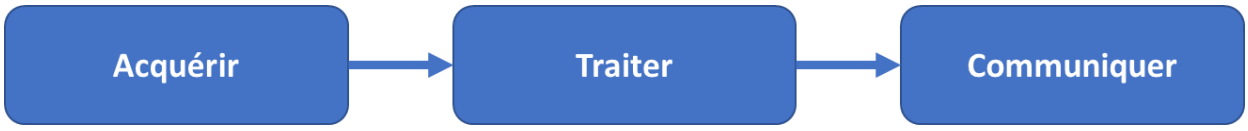
La carte micro:bit est constitué de nombreux éléments :

- un processeur ARM ;
- 25 leds programmables individuellement ;
- 2 boutons programmables ;
- broches de connexion ;
- capteurs de lumière et de température ;
- capteurs de mouvements (accéléromètre et boussole) ;
- communication sans fil (radio et bluetooth) ;
- interface USB.



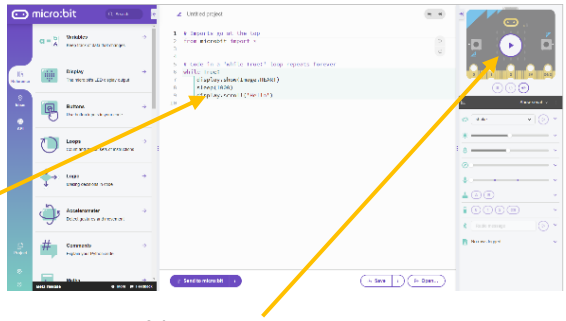
1.2. Chaîne d'information

✍ À partir des constituants de la carte micro:bit, **identifier** les éléments qui constitue les blocs Acquérir, Traiter et Communiquer de la chaîne d'information et **compléter** le schéma ci-dessous.



2. Hello

✍ Ouvrir le logiciel en ligne Python Editor : https://python.microbit.org/v/beta.



2.1. Simulation du programme

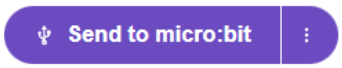
✍ Pour vérifier le fonctionnement du programme en simulation, cliquez sur la flèche **Play**.

✍ **Décrire** ce qui se passe :

2.2. Test du programme

✍ **Connecter** la carte micro:bit à l'ordinateur à l'aide du câble USB.

✍ **Implanter** le programme dans la carte micro:bit en cliquant sur **Send to micro:bit** puis suivre les instructions à l'écran.



✍ **Observer** la carte micro:bit. Obtenez-vous la même chose que pour la simulation ?

Explication du code

```
1  # Imports go at the top
2  from microbit import *
3
4
5  # Code in a 'while True:' loop repeats forever
6  while True:
7      display.show(Image.HEART)
8      sleep(1000)
9      display.scroll('Hello')
10
```

Import de la bibliothèque **`microbit`** nécessaire au fonctionnement de la carte.

Boucle infinie.

Affichage (**`show`**) de l'image d'un cœur (HEART).

Temporisation de 1 000 millisecondes.

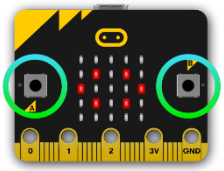
Défilement (**`scroll`**) de la chaîne de caractères **`'Hello'`**.

✍ **Compléter** le programme ci-dessous afin d'**afficher** l'image HAPPY pendant 3 secondes puis de faire **défiler** le message **`'Bienvenue en classe de SIN'`**.

```
from microbit import *

while True:
```

✍ **Simuler** votre programme puis l'**implanter** dans la carte micro:bit.

3. Boutons



La carte micro:bit contient deux boutons nommés A et B (**button_a** et **button_b**).
La méthode **is_pressed** retourne **True** si l'objet associé (**button_a** et **button_b**) associé a été appuyé. Sinon elle retourne **False**.

✍ **Simuler** le programme ci-dessous puis l'**implanter** dans la carte micro:bit.

```
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

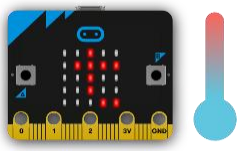
✍ **Implémenter** le programme qui fait **défiler** votre prénom si le **bouton B** est appuyé. Sinon l'image d'un carré est affichée (**Image.SQUARE**).

```
from microbit import *

while True:
    if
```

✍ **Simuler** votre programme puis l'**implanter** dans la carte micro:bit.

4. Température

La carte micro:bit peut donner la température en degrés Celsius.



La carte micro:bit n'a pas un capteur de température dédié. Au lieu de cela, la température fournie est en fait la température de la puce de silicium du processeur principal. Comme le processeur chauffe peu en fonctionnement (c'est un processeur ARM à grande efficacité), sa température est une bonne approximation de la température ambiante.

✍ **Simuler** le programme ci-dessous puis l'**implanter** dans la carte micro:bit.

```
from microbit import *

while True:
    val = temperature()
    display.scroll(val)
    sleep(5000)
```

✍ **Compléter** le programme afin de faire défiler la valeur de la température lorsque le **bouton A** est actionné.

```
from microbit import *

while True:
    val = temperature()
    if
```

✍ **Simuler** votre programme puis l'**implanter** dans la carte micro:bit.

5. Image

✍ **Simuler** le programme ci-dessous puis l'**implanter** dans la carte micro:bit.

```
from microbit import *

boat = Image("05050:"
             "05050:"
             "05050:"
             "99999:"
             "09990")

display.show(boat)
```

Le programme permet d'afficher un bateau, mais il peut aussi être écrit comme ci-dessous.

```
boat = Image("05050:05050:05050:99999:09990")
```

✍ **Implémenter** un programme afin de créer une flèche de signalisation (signe >).

```
from microbit import *
```

✍ **Simuler** votre programme puis l'**implanter** dans la carte micro:bit.

6. Animation

Pour créer une animation, il suffit de créer une liste contenant les images de l'animation.

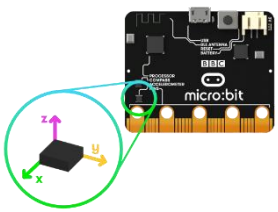
✍ **Simuler** le programme ci-dessous puis l'**implanter** dans la carte micro:bit.

```
from microbit import *

bateau1 = Image("05050:05050:05050:99999:09990")
bateau2 = Image("00000:05050:05050:05050:99999")
bateau3 = Image("00000:00000:05050:05050:05050")
bateau4 = Image("00000:00000:00000:05050:05050")
bateau5 = Image("00000:00000:00000:00000:05050")
bateau6 = Image("00000:00000:00000:00000:00000")

tous_les_bateaux = [bateau1,bateau2,bateau3,bateau4,bateau5,bateau6]
display.show(tous_les_bateaux, loop=True, delay=200)
```

Ici **`display.show`** va afficher toutes les images de la liste **`tous_les_bateaux`** avec un temps de 200 millisecondes entre chaque image (**`delay=200`**). L'animation est réalisée en boucle grâce à l'instruction **`loop=True`** (par défaut **`loop=False`**).

✍ **Implémenter** une animation qui fait bouger la flèche de signalisation.

7. Accéléromètre



L'accéléromètre mesure l'accélération de la carte micro:bit, ce composant détecte quand le micro:bit est en mouvement. Il peut aussi détecter d'autres actions, par exemple quand il est secoué, incliné ou qu'il tombe.
L'accéléromètre mesure les mouvements suivant trois axes :

- X : inclinaison gauche droite
- Y : inclinaison devant derrière
- Z : monté descente

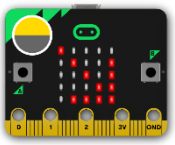
Les méthodes **`get_x`**, **`get_y`**, and **`get_z`** retournent, pour chaque axe, une valeur positive ou négative en fonction de l'accélération.
L'accéléromètre reconnait aussi les mouvements suivants : **`up, down, left, right, face up, face down, freefall, 3g, 6g, 8g, shake`**. La méthode **`current_gesture()`** retourne le mouvement.
La méthode **`was_gesture(name)`** retourne **`True`** ou **`False`** pour indiquer si le dernier mouvement cité (**`name`**) a été détecté depuis le dernier appel de la méthode. Les mouvements doivent être des chaînes de caractère (**`'up'`** par exemple).

✍ **Simuler** le programme ci-dessous puis l'**implanter** dans la carte micro:bit.

```
from microbit import *

while True:
    reading = accelerometer.get_x()
    if reading > 20:
        display.show("R")
    elif reading < -20:
        display.show("L")
    else:
        display.show("-")
```

✍ **Implémenter** un programme qui affiche 'Ne me secoue pas' lorsque la carte est secouée. Le mouvement de la carte secoué est reconnu par l'instruction **shake**.

8. Lumière



La carte utilise les LED de l'écran en mode de polarisation inverse pour détecter la quantité de lumière tombant sur l'écran. Un entier entre 0 et 255 est renvoyé représentant le niveau de lumière. Une valeur forte montre une forte quantité de lumière.

✍ **Simuler** le programme ci-dessous puis l'**implanter** dans la carte micro:bit.

```
from microbit import *

while True:
    val = display.read_light_level()
    display.scroll(val)
    sleep(5000)
```

✍ **Implémenter** un programme qui détecte la nuit (inférieur à 5), le jour et l'ensoleillement (supérieur à 100). Le mot correspondant à la situation lumineuse est affiché.

Source : https://microbit-micropython.readthedocs.io/fr/latest/