Séance 2 : Activité 2 Prise en main micro:bit

Sources : Olivier Eloi
Thème : Informatique embarquée et objets connectés
Environnement envisagé :

- PC
- Micro:bit
- logiciel Mu de développement pour micro:bit.

Prérequis :

- Initiation au python

Durée :

- séance 30-40min

Objectifs :

- Découverte de la carte micro:bit
- Réaliser quelques programmes d'informatique embarquée en python.
- Écrire des programmes simples d'acquisition de capteur et de communication avec l'utilisateur : première IHM

Séance 2- Activité 2: La carte micro:bit

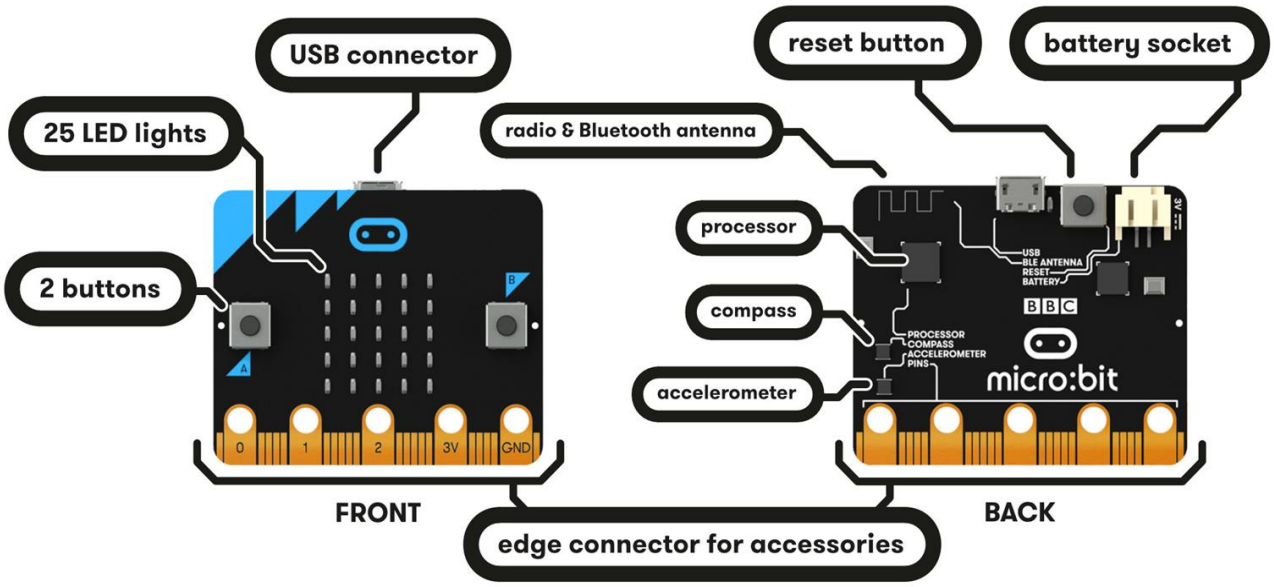
1. Présentation de la carte micro:bit

Les ressources sur la micro:bit :
En français : https://microbit.org/fr/
En anglais : https://microbit-micropython.readthedocs.io/en/latest/index.html

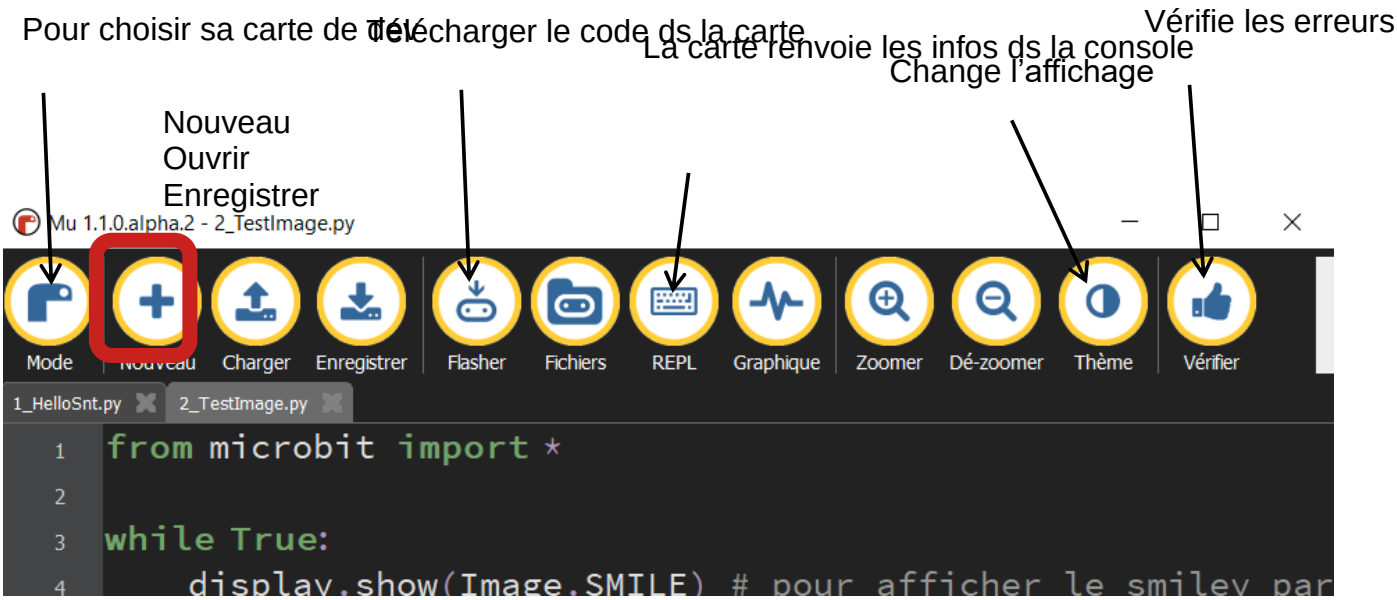
Les ressources pour micropython :
En anglais (la référence): http://micropython.org/

Télécharger Mu : Privilégier les versions alpha pour les ESP https://codewith.mu/en/download

La carte micro:bit



Le logiciel Mu



2. Travail demandé

Des symboles prédéfinis peuvent être affichés sur la matrice de LEDs.
Le nom des symboles sont
Image.HEART, Image.HEART_SMALL, Image.HAPPY, Image.SMILE, Image.SAD, Image.CONFUSED, Image.ANGRY, Image.ASLEEP, Image.SURPRISED, Image.SILLY, Image.FABULOUS, Image.MEH, Image.YES, Image.NO, Image.CLOCK12, Image.CLOCK11, Image.CLOCK10, Image.CLOCK9, Image.CLOCK8, Image.CLOCK7, Image.CLOCK6, Image.CLOCK5, Image.CLOCK4, Image.CLOCK3, Image.CLOCK2, Image.CLOCK1, Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E, Image.ARROW_SE, Image.ARROW_S, Image.ARROW_SW, Image.ARROW_W, Image.ARROW_NW, Image.TRIANGLE, Image.TRIANGLE_LEFT, Image.CHESSBOARD, Image.DIAMOND, Image.DIAMOND_SMALL, Image.SQUARE, Image.SQUARE_SMALL, Image.RABBIT, Image.COW, Image.MUSIC_CROTCHET, Image.MUSIC_QUAVER, Image.MUSIC_QUAVERS, Image.PITCHFORK, Image.XMAS, Image.PACMAN, Image.TARGET, Image.TSHIRT, Image.ROLLERSKATE, Image.DUCK, Image.HOUSE, Image.TORTOISE, Image.BUTTERFLY, Image.STICKFIGURE.

Travail 1 :

La ligne de code permettant d'afficher un symbole est :

```
display.show(Image.SMILE) # pour afficher le smiley par exemple
```

La syntaxe du code et l'alignement de celui-ci (que l'on nomme indentation) sont
extrêmement importants. Il est donc impératif de respecter :
• les MAJUSCULES et les minuscules,
• la position des lignes de codes les unes par rapport aux autres.

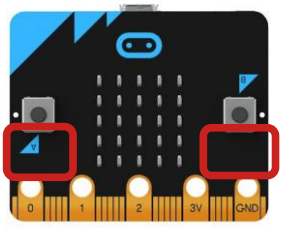
Un exemple de code pour afficher 3 symboles pourrait être :

```
from microbit import *
while True :
        display.show(Image.HEART)
        sleep(500)
        display.show(Image.HEART_SMALL)
        sleep(500)
        display.show(Image.BUTTERFLY)
        sleep(500)
```

Taper le code permettant d'afficher le symbole PACMAN, suivi de ROLLERSKATE, puis enfin HOUSE. Un temps de repos sleep(500) devra être inséré entre chaque affichage afin d'observer chacune des figures.

Travail 2 :

2 boutons sont présents sur la carte afin de rajouter de l'interactivité dans les programmes.



Les fonctions associées sont :
button_a.is_pressed() renvoie True si le bouton est effectivement appuyé
button_a.was_pressed() renvoie True si le bouton a été appuyé
button_a.get_pressed() renvoie le nombre de fois que le bouton a été appuyé.

Les mêmes fonctions existent pour le bouton B en remplaçant la lettre ‘’a’’ dans la fonction par ‘’b’’.

Voici un exemple de code utilisant la première fonction :

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

Tester ce bout de code.

Travail 3 :

On veut réaliser un compte à rebours à partir de 10.
L'algorithme est le suivant :

```
Debut compte_a_rebour :
        Faire infiniment :
                Pour i de 10 à 0 par pas de -1 :
                        afficher(i, pendant 1s)
Fin compte_a_rebours
```

Remarques

• La fonction d'affichage possède des paramètres qui permettent de régler le temps d'affichage, et l'extinction des LEDs entre 2 affichages, la syntaxe est la suivante :

display.show(str(i) , delay = 1000, loop= False, clear = True)

• Une boucle **POUR** décomptant de 5 en 5 en partant de 100 en python s'écrit :

```
for i in range(100, 0, -5) :
        display.show(str(i) , delay = 1000, loop= False, clear = True)
```

Travail 4 : Réaliser un dé électronique à 6 faces.

L'appui sur le bouton A provoque l'affichage d'un nombre choisi au hasard entre 1 et 6 par la microbit. On utilise la fonction **random.randint(a,b)** pour tirer un nombre au hasard, il est nécessaire au préalable d'importer la bibliothèque random, voici un exemple de code :

```
from microbit import *
import random
tirage= random.randint(1,100) # tirage contient un nombre entier au hasard entre 1 et 100
```

Bonus : simuler le roulage du dé, en faisant une petite animation avant d'afficher le nombre.