DBSF - Préformation

Micro:bit

Bastien Gorissen & Thomas Stassin

Disclaimer

Disclaimer

Les slides qui suivent ont pour but de vous enseigner comment utiliser le BBC Micro:bit. Ceci ne remplace pas la lecture de la documentation de celui-ci.
Donc, en lectures supplémentaire, on vous conseille:

https://microbit.org/

https://microbit.org/get-started/user-guide/python/

(IM)MATÉRIEL

"What's in the box?!"

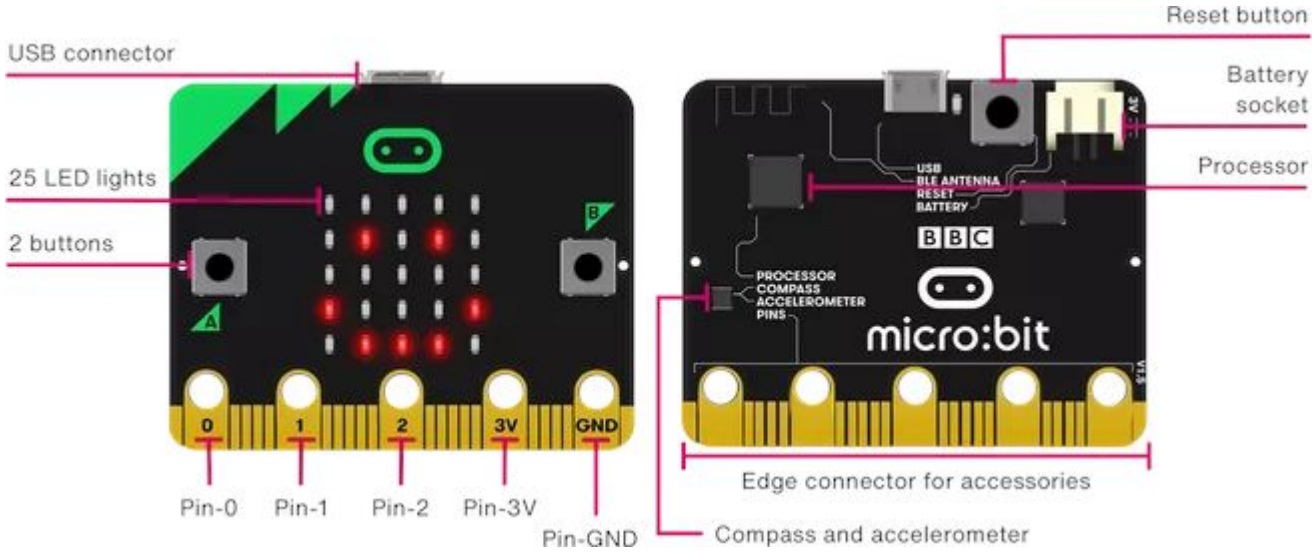
Liste de courses

Pour l'atelier, nous allons utiliser un nombre limité d'outils :

- Python
- La star du show : Un mini-ordinateur, le micro:bit
- "mu", un éditeur de code que maintenant vous connaissez bien.

Le BBC micro:bit

Le micro:bit est un mini-ordinateur destiné à l'éducation, permettant aux plus jeunes de découvrir la programmation de façon ludique.



mu (pour ceux qui auraient oublié)

mu est un éditeur de code pour Python conçu spécialement pour les débutants. Il vient avec juste ce qu'il faut d'options pour travailler avec Python, et un mode spécifique dédié au micro:bit !

On peut télécharger l'installeur en passant par le site :

https://codewith.mu/

(On peut aussi trouver des instructions ici, y compris si le programme doit être installé automatiquement avant les ateliers : https://codewith.mu/en/howto/1.0/install_windows)

Hello micro:bit!

"Such a small thing…"

Faire fonctionner le micro:bit

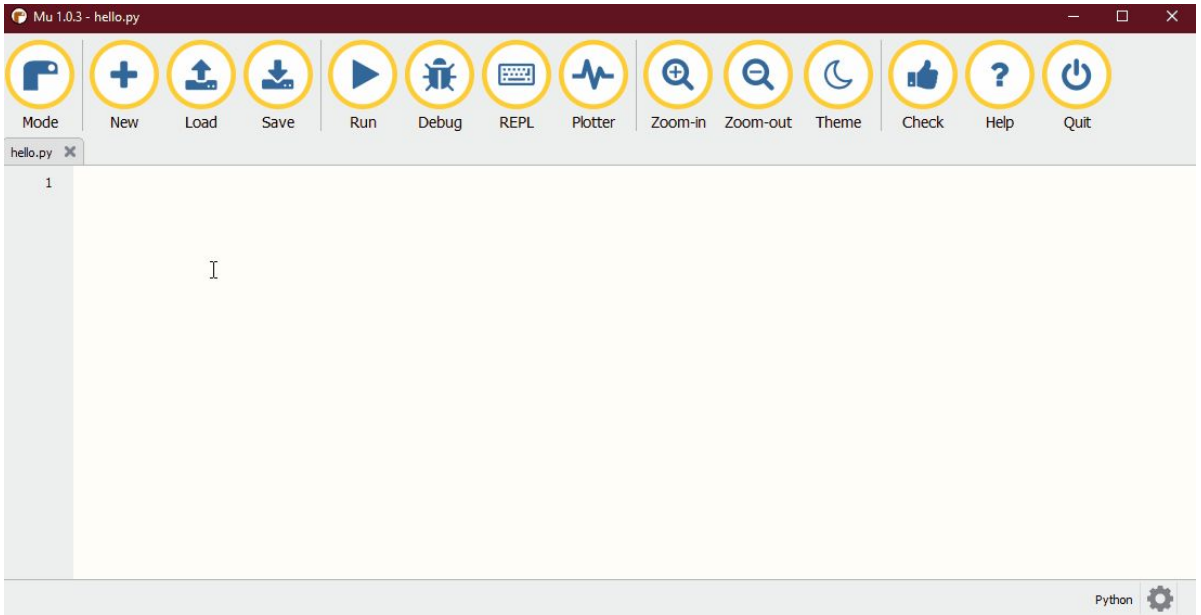
Pour commencer, voyons simplement quel est le processus pour mettre du code sur le micro:bit.

Le micro:bit peut être alimenté soit via des piles, soit en étant connecté par USB au PC.

Connectez-le donc avec le câble du kit, et passons mu en mode "micro:bit". (Il va normalement reconnaître le micro:bit directement.)

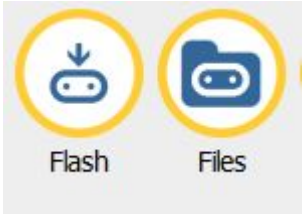
Changer de mode manuellement

Vous pouvez aussi choisir le "mode" de mu manuellement avec le bouton en haut à gauche de la fenêtre.



Des nouveaux boutons !

La différence majeure entre le mode micro:bit et le mode normal est le remplacement du bouton "Run" par un bouton "Flash" qui envoie le code sur le micro:bit via USB. Le bouton Files permet de voir les fichiers présents sur l'appareil.



L'utilisation du micro:bit (et du robot) demande donc d'écrire le code sur le PC, et puis de l'envoyer sur le contrôleur, qui va l'éxécuter.

Premier programme micro:bit !

Pour notre premier programme, nous allons donner un petit visage à notre micro:bit. Dans un nouveau fichier Python, écrivez le code suivant :

```
from microbit import *

display.show(Image.HAPPY)
```

Ensuite, cliquez sur "Flash", et attendez quelques secondes…

La boucle Infinie

"Pour que cela ne s’arrête jamais"

Une boucle infinie, pourquoi faire?

Précédemment, nous avons qu'une boucle infinie est quelque chose qu'on préférait éviter.

Dans le cadre d'un programme classique, il y aura toujours un début et une fin, donc la boucle infinie est peu recommandée.

Dans le cadre de la programmation sur le Micro:bit, dans la majorité des cas, vous serez intéressés par faire tourner le programme tout le temps, ou du moins tant que vous ne l'aurez pas éteint. Dans ce cas une boucle infinie est interressante.

Une boucle infinie, pourquoi faire?

Voici comment la structure de base de la plupart des codes que vous réaliserez sur le Micro:bit

```
from microbit import *

#code d'initialisation

while True:

    # code devant se répéter à l'infini.
```

Une boucle infinie, pourquoi faire?

Avant la boucle infinie `while True` se trouvera le code devant être exécuté avant d'entrer dans la boucle, comme par exemple la déclaration des variables qui seront utilisées dans la boucle.
Le reste du code sera lui "contrôlé" par la boucle while et sera exécuté en permanence.

Une boucle infinie, pourquoi faire?

Dans cet exemple, si la variable `counter` n'avait pas été déclarée en dehors de la boucle, le résultat aurait été quelque peu décevant.

```
from microbit import *

counter = 0

while True:
    display.scroll(counter)
    counter += 1
```

Display

"Avec un écran de 5x5, une résolution de fou!!!"

25 pixels de bonheur

Le Micro:bit possède sur sa face avant un écran LED de 25 pixels (5x5) ou chacune des LED peut avoir une intensité lumineuse de 0 à 9.

Bien qu'on est déjà vu quelques mécanismes lié à `Display`, dans cette section, on va un peu plus s'étendre sur le sujet.

Scroll

`scroll` est une méthode de `display` qui affiche ce que vous lui passez en paramètre en le faisant scroller sur l'écran LED.

```
while True:
    display.scroll("Python")
```

Ce code affichera “Python” en boucle sur votre Micro:bit

Scroll - delay

Un des paramètres possible pour `scroll` est `delay`. Il régule la vitesse.
Plus le délai est grand plus la vitesse est faible et inversement. La vitesse par défaut est de 150.

```
while True:

    display.scroll("Python", delay=75)
```

Show

`show` est une autre méthode de `display` qui affiche ce que vous lui passez en paramètre.
A la différence de scroll, il n'y aura pas de scrolling et si le message contient plusieurs lettres (ou chiffres) ils seront affichés un par un.

```
while True:
    display.show("Python")
```

Ce code affichera “P”, “y”, “t”, “h”, “o”, “n” en boucle.

Image

`show` permet aussi d'afficher une "image".

Les images sont des matrices 5x5 qui expriment ce qu'il faut éclairer sur l'écran. Nous y reviendrons dans un instant.

Nous avons déjà utilisé une image dans notre "Hello World" mais il en existe plusieurs de pré-enregistrées.

```
while True:

    display.show(Image.DUCK)
```

Image

On peut construire son image soi-même assez facilement. L'écran étant une matrice 5x5, on doit décrire son image en donnant une intensité entre 0 et 9 pour chacune des LED de l'écran.

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

Image

Après il ne reste plus qu’à l’afficher:

```
while True:

    display.show(boat)
```

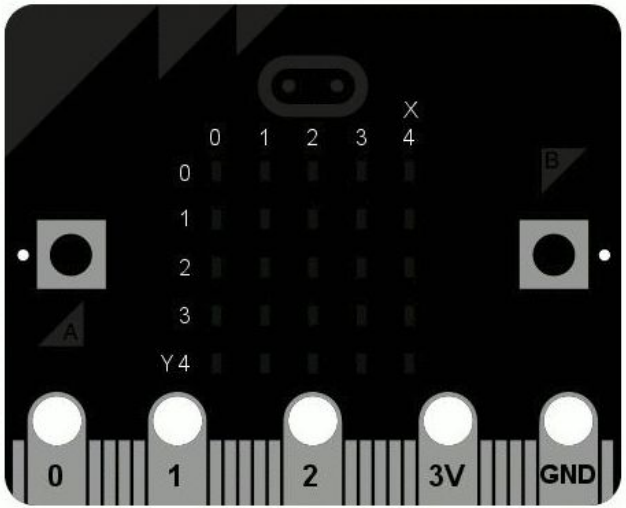
La chaîne de caractère qui représente l’image aurait pu s’écrire en une ligne ("05050:05050:05050:99999:09990") mais pour plus de clarté on préfère l’écrire en 5.

Chaque passage à la ligne sur l’écran doit être indiqué par un “:”

Set and Get Pixel

Il y a aussi un moyen d’agir directement sur les pixels de l’écran.

La valeur en x et y des LED de l’écran sont répartie comme suit:



set and Get Pixel

la méthode `set_pixel` prend 3 paramètres: x, y et l'intensité de la LED.

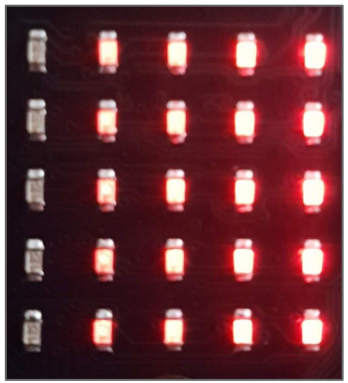
`display.set_pixel(2, 2, 5)` ajoutera un pixel au milieu de l'écran d'intensité moyenne.

La méthode `get_pixel` prend elle 2 paramètres: x et y.
`display.get_pixel(2, 2)` renverra 5, par exemple.

Exercice

Ecrivez un code qui, en utilisant `set_pixel`, donnera à chaque pixel de votre écran un intensité égale à sa valeur en x. De cette manière vous devriez obtenir un dégradé.

Comme ceci:



Clear

`display.clear()` est un fonction qui efface l'écran, remettant tous les pixels à 0.

Ca ne parait pas très utile, sauf si on a besoin d'un écran vierge. Mais c'est mieux que d'utiliser une boucle avec `set_pixel` pour désactiver tout les pixels de l'écran.

Sleep

"zzz..."

Sleep

sleep est une fonction qui endort le programme pour un temps défini en paramètre.

`sleep(1000)` met en pause l'exécution du programme pour 1000 millisecondes, c’est à dire 1 seconde.

Sleep

```
x = 0
while True:
    display.set_pixel(x, 0, 9)
    x += 1
    x %= 5
```

Change beaucoup si on ajoute un sleep à la fin.

```
x = 0
while True:
    display.set_pixel(x, 0, 9)
    x += 1
    x %= 5
    sleep(1000)
```

Temperature

"chaud devant..."

Thermometre

Le Micro:bit est équipé de plusieurs senseurs, l’un des plus facile à appréhender est le thermomètre.

Pour récupérer la température (en celsius) autour du Micro:bit il suffit d'utiliser la fonction `temperature()`, celle-ci retourne un entier correspondant à la température.

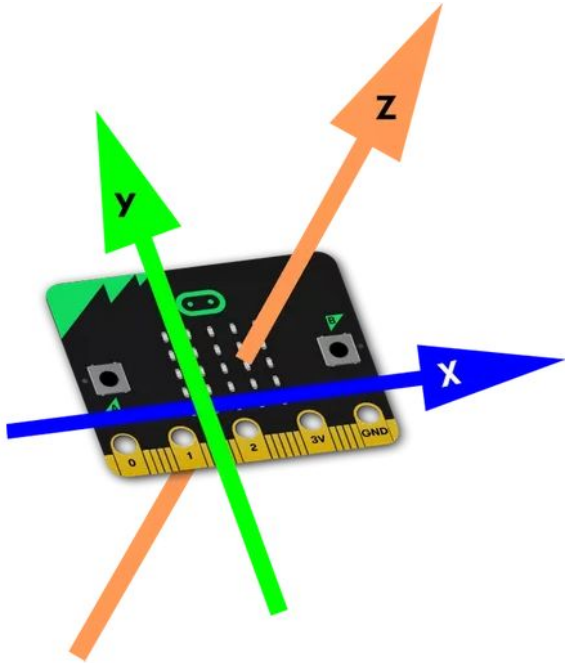
Fort de cette information vous pouvez maintenant faire votre propre thermomètre qui affichera en permanence la température.

Accelerometre

"Là où il est question de gesture"

Accelerometre

Le Micro:bit est équipé d’un accéléromètre qui permet donc de connaitre l’accélération que subit le micro:bit sur 3 axes: x, y et z



Accelerometre

Juste pour voir ce que ça donne, écrivez le code suivant:

```
while True:
    print(accelerometer.get_values())
    sleep(50)
```

Restez connectés au PC après avoir flashé le code et aller voir dans plotter 

ce qui se passe.

Harry Plotter

Plotter est un outil de Mu qui permet de visualiser les `print` qui contiennent des tuples en graphique (la ou les valeurs iront sur l'axe des y et l'axe représentera le temps passé.

Donc si vous avez besoin d'analyser une donnée à l'aide d'un graphique, il suffit d'en faire un tuple.

protip: pour transformer un donnée en tuple:

```
a = 3
print((a,))
```

`(a,)`c'est comme ça qu'on écrit un `tuple` de 1 élément

Gesture

En dehors du faite de pouvoir capter un l'accélération, le Micro:bit reconnait les "gestes" (gestures in English) que l'on fait avec lui.
Ils existent plusieurs gestures qui peuvent être captées.

Par exemple: "shake" qui représente le faite de secouer le Micro:bit.

La méthode `accelorometer.was_gesture` prend une *gesture* en paramètre et renvoie `True` si cette gesture a été exécutée depuis le dernier appel de la méthode.

Gesture

```
while True:
    if accelerometer.was_gesture("shake"):
        display.show(Image.SAD)
        sleep(2000)
    else:
        display.show(Image.HAPPY)
```

Ce code fait en sorte que le Micro:bit fasse une tête triste quand il est secoué, sinon, il fait une tête contente.

Gesture - Exercice

Faites en sorte que lorsque que le Micro:bit est retourné face vers l’avant (la gesture “face up”), il affiche un chiffre entre 1 et 6.

Compass

"Pour ne jamais perdre le nord"

Calibration

Pour utiliser la boussole, il faut commencer par la calibrer, pour cela, rien de plus facile.

```
compass.calibrate()
```

Vous verrez lors de l'exécution que le micro:bit vous demandera de faire quelques manipulations.

Après ça, la direction du compas vous sera donnée par `compass.heading()`

Compass

```
compass.calibrate()

while True:
    direction = compass.heading()
    display.scrolling(direction)
```

Le code suivant donne un angle de 0 à 359 (0 étant le nord et 180 le sud).

Compass

```
compass.calibrate()

while True:

    direction = compass.heading()

    direction = ((15 - direction) // 30) % 12

    display.show(Image.ALL_CLOCKS[direction])
```

Ce code vous affichera une aiguille pour la boussole.
Le calcul `((15 - direction) // 30) % 12` donne un chiffre de 0 à 11. `ALL_CLOCKS` contenant la position des aiguille d'une horloge, indexées de 0 à 11.

Buttons

"C’est un truc de malade"

Buttons

Vous l'aurez sans doute remarqué, le Micro:bit possède deux boutons, A et B (oui, comme la NES).

Il y a une méthode pour détecter qu'ils ont été pressés: `button_a.was_pressed()` et `button_b.was_pressed()`

Ex:

```
value = 0

while True:
    display.show(value)
    if button_a.was_pressed():
        value += 1
```

Is or Was

Une autre méthode possible est is_pressed().

was_pressed checke si le bouton a été appuyé depuis la dernière fois que la méthode a été appelée.

is_pressed checke si le bouton est en train d'être appuyé au moment où la méthode est appelée.

Faisons une expérience pour mieux illuster.

Is or Was

```
value = 0

while True:
    for n in range(value % 26):
        x = n % 5
        y = n // 5
        display.set_pixel(x, y, 9)
    if button_a.was_pressed():
        value += 1
```

Remplacez le `button_a.was_pressed()` par un `button_a.is_pressed()` et constater le changement.

Lire la Lumière

Dieu a dit : "Que la lumière soit !"
Chuck Norris lui a répondu : "On dit s’il vous plaît !"

Fiat Lux

Aussi surprenant que cela puisse être, la méthode pour lire le niveau de lumière vient de `display` (c'est à dire des 25 LED composant l'écran).
La méthode se nomme `read_light_level` et renvoient une valeur entre 0 et 255 correspondant au niveau de lumière captée.

Vous pouvez tester avec le code suivant:

```
while True:
    value = display.read_light_level()
    display.scroll(value)
```