4 — La bibliothèque microbit

4.1 — Découverte de l'environnement Micro: bit

4.1.1 — Découverte de la carte Micro: Dit



Tu as a ta disposition une carte Micro:bit. Répondre aux questions suivantes concernant la carte.

1. Observer la carte et énumérer les différents constituants de la

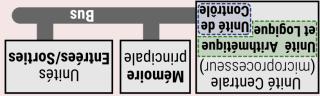
carte Micro:bit.

2. **Indiquer** le rôle de chacun.

3. D'après vous, à quoi sert le composant Processor? que contient-il?

Comme tout modèle d'ordinateurs, la structure de la carte Micro:bit est conforme à un schéma inventé en 1945 et qui a peu évolué depuis :

l'architecture de Von Neumann.



Cerveau de la machine, l'unité centrale (ou microprocesseur) est constituée d'une unité arithmétique et logique (pour effectuer les calculs) et d'une unité de contrôle (pour pilote les échanges à travers le Bus. La mémoire contient à la fois les données et les programmes. Les périphériques d'entrées/sorties permettent d'envoyer ou de recevoir des données avec l'extérieur (clavier, souris, écran, imprimante, etc.).



## **4.1.2 — Premier programme avec la carte Micro: bit**



D'après toi, que fait le programme ci-dessous?

```
from microbit import *
e0 = button_a.is_pressed()
if e0:
    display.show("1")
else:
    display.show("0")
```

Ouvrir l'éditeur Mu-editor, recopier ce programme et le téléverser sur la carte (bouton Flasher ...).

2. **Observe** le fonctionnement de la carte et **propose** une ou plusieurs interprétations (pour confirmer/infirmer tes observations, tu peux manipuler les boutons présents sur la carte : A / B / Reset).

Présenter tes réponses au professeur.

- 3. **Indique** le nombre de fois qu'est exécutée l'instruction conditionnelle.
- 4. **D'après toi**, combien de fois doit être exécutée l'instruction conditionnelle pour que la carte reste à l'écoute des entrées et réagisse chaque fois que l'état du bouton A change?
- 5. **Améliore** le programme en tenant compte de tes observations et de tes réponses aux questions précédentes.

Présente la carte programmée au professeur.



#### 4.2.11 - Nuages de points avec Mu-editor

Il est possible de tracer un nuage de points avec l'éditeur Mu-editor.

Pour cela, il faut

- utiliser l'outil Graphique de Mu-editor,
- faire afficher par le programme en cours d'exécution un tuple de nombres.

## Exemple

Le programme ci-dessous affiche dans le terminal (ou dans le **Graphique** si l'outil de l'IDE est cliqué) deux nombres aléatoires appartenant à - 100..100.

```
from microbit import *
from random import randint
drapeau = True
while True:
    sleep(50)
    if button_a.was_pressed():
        drapeau = not drapeau
    if drapeau:
        nb1 = randint(-100,100)
        nb2 = randint(-100,100)
        print( (nb1,nb2) )
```

La fonction  $\text{button\_a.is\_pressed}()$  peut être avantageusement remplacée par un appel à la fonction  $\text{button\_a.was\_pressed}()$ . Son appel renvoie une valeur booléenne True (vrai) ou False (faux) et permet de savoir si le bouton A a été pressé depuis la dernière fois que permet de savoir si le bouton A a été pressé depuis la dernière fois que

Son appel renvoie une valeur booléenne  $T^{rue}$  (vrai) ou False (faux) et permet de savoir si le bouton A a été pressé depuis la dernière fois que cette fonction a été appelée. Cette fonction renvoie  $T^{rue}$  s'il a été pressé entre deux appels et False sinon.

6. **Modifie** le programme pour que l'affichage 0/1 change uniquement lorsque l'on presse le bouton A (et pas lorsqu'on le relâche).

## 4.2 — Commandes de base pour programmer avec Micro:bit

Dans la suite, tu utiliseras l'éditeur mu-editor pour programmer la carte Micro:bit. Si tu le souhaites tu pourras, **sur ton temps personnel**, essayer de programmer la carte avec VSCodium...

### 3.2.1 — La bibliothèque microbit

En Python, les entrées/sorties de la carte Micro:bit ne sont pas nativement accessibles. Afin de pouvoir utiliser les fonctions prévues à cet effet, il faut utiliser la bibliothèque microbit.

## **Exemple**

Pour importer cette bibliothèque, il faut utiliser la commande :

from microbit import \*

L'utilisation de cette bibliothèque nous permettra d'utiliser différentes fonctionnalités de la carte Micro:bit:

## ВЕМАВО

Avant d'utiliser une fonctionnalité du module compass, il faut obligatoirement calibrer la carte. Sans cela, les valeurs renvoyées sont fausses à cause du bruit magnétique présent dans l'environnement de la carte.

Au moment de calibrer la boussole, l'utilisateur doit bouger la carte dans différentes positions jusqu'à faire passer le point clignotant par toutes les LED de l'écran.

L'outil de calibration se lance automatiquement mais il est possible de programmer son exécution en appelant l'instruction compass.calibrate().

## Exemple

Le programme ci-dessous fait office de boussole et indique le nord magnétique.

Attention, le capteur est sensible aux objets tels que téléphones, ordinateurs ou aux lieux tels que ascenseurs ou salle informatique...

```
from microbit import *
compass.calibrate()
button_b.was_pressed()
while True:
    cap = ((15 - compass.heading()) \\ 30) % 12
    display.show(Image.ALL_CLOCKS[cap])
    if button_b.was_pressed():
        display.clear()
        break
```



- Image pour créer et manipuler les images.
- Button avec deux instances button\_a et button\_b pour connaître l'état des boutons.
- Pin avec différentes instances en fonction du type de broche (par exemple pin0, pin1 et pin2).
- display pour gérer l'écran de LED
- accelerometer pour interroger l'accéléromètre
- **compass** pour manipuler et interroger la boussole
- music pour créer et manipuler de la musique
- **speech** pour faire parler le Micro: bit
- radio pour communiquer entres plusieurs Micro:bit via un protocole simple.

#### 4.2.2 - Micropython dans la carte Micro: bit

Lorsque la carte Micro: bit est flashée par mu-editor avec la bibliothèque microbit, elle contient un noyau *micropython*. Il est alors possible d'écrire du code qui sera interprété par ce noyau micropython de la carte.

Pour accéder à l'interprète Python de la carte, il suffit de cliquer sur l'icône REPL de l'application.

#### **Exemple**

Pour utiliser le terminal de la carte Micro: bit :

- 1. Flasher la carte Micro: bit avec micropython.
- 2. Ouvrir le terminal de la carte en cliquant sur REPL
- 3. Écrire le code ci-dessous :

```
print("Hello, World!")
```



```
button_b.was_pressed()
while True:
    # émetteur
    if button_a.was_pressed():
        radio.send("A")
    if button_b.was_pressed():
        radio.send("B")
    # récepteur
   message = radio.receive()
    if message == "A":
        display.scroll("A")
   if message == "B":
        display.scroll("B")
    # pause pour éviter de saturer la carte
    sleep(20)
    # sortir de la boucle
    if pin0.is_touched():
        display.clear()
        break
```

#### 4.2.10 - La boussole

Pour utiliser la boussole de la carte Micro:bit, il faut utiliser le module compass.

- $compass.get_x()$  (ou  $.get_y()$  ou  $.get_y()$ ) renvoient une des composantes du vecteur champ magnétique.
- compass.heading() renvoie un nombre entier (int) correspondant à l'angle en degré entre l'orientation de la carte Micro:bit et le nord magnétique.

## 4.2.3 - Afficher un texte

Le module  ${\tt display}$  permet de gérer l'écran de LED de Micro: bit. Les fonctionnalités offertes par ce module sont accessible en ajoutant un point '.' après 'display' puis en écrivant le nom de la fonction à appeler :

- display.show(mon\_texte) permet d'afficher la chaîne de caractère
   contenue dans la variable mon\_texte
- display.scroll(mon\_texte) permet de la Micro:bit
- display.clear() permet d'effacer l'écran.

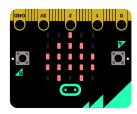


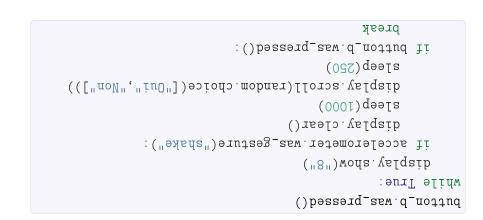
Micro: bit le mot Hello puis afficher la chaîne World!.

# CORRECTION ( display.show("World!") ( display.show("World!")

#### 4.2.4 — Des images

La classe  ${\rm Image}$  contient comme attributs de nombreuses images préprogrammées. Ces constantes sont accessibles grâce à un point ' . ' placé après '  ${\rm Image}$ '. Par exemple l'image du coeur est accessible via l'instruction  ${\rm Image}$  :





#### 4.2.9 — La radio

Les cartes Micro: bit peuvent communiquer entres elles au moyen du module radio.

- radio.send(texte) permet d'envoyer par radio la chaîne de caractère texte.
- radio.receive() renvoie les données reçues par radio converties en une chaîne de caractère. Si rien n'a été reçu, la chaîne vaut None.

## Exemple

Le programme ci dessous est à téléverser sur 2 cartes Micro:bit. Il contient à la fois une partie émetteur et une partie récepteur. Un appui sur les boutons A ou B de l'une ou l'autre des cartes Micro:bit affiche les texte "A" ou "B" sur l'autre. Un appui sur la broche pim0 arrête le programme.

```
* troqmi tidort
tron_a.was_pressed()
```



Comme pour les textes, les images s'affichent grâce à l'instruction display.show().

## **Exemple**

Voici par exemple comment afficher successivement des images : une image est affichée pendant une seconde puis ensuite une autre image s'affiche. Une seconde plus tard, l'écran s'efface.

```
from microbit import *
display.show(Image.HAPPY)
sleep(1000)
display.show(Image.ANGRY)
sleep(1000)
display.clear()
```

Il faut mettre le Micro: bit en pause. Sinon l'utilisateur n'aura pas le temps de tout voir. La commande sleep(duree) arrête donc la carte pendant une durée duree exprimée en milliseconde par un nombre entier (int).

#### 4.2.5 - Des boutons

Pour connaître les états des boutons de la carte Micro: bit on utilise les classes button\_a (pour le bouton A de la carte) et button\_b (pour le bouton B).

Comme d'habitude, les fonctionnalités offertes par ces classes sont accessibles via un point ' . ' placé après le nom de la classe :

- get\_presses() renvoie le nombre de fois que le bouton a été pressé depuis le dernier appel de cette fonction
- .is\_pressed() renvoie une valeur booléenne (vrai True ou fausse False) qui indique si le bouton est actuellement pressé



#### 4.2.8 - Les gestes

Le module accelerometer peut aussi détecter des mouvements ou des positions pré-programmés: les gestes ('up', 'down', 'left', 'right', 'face up', 'face down', 'freefall', '3g', '6g', '8g', 'shake').

## **REMARQUE**

L'utilisation des gestes est très lente.

#### Pour cela, il existe:

- accelerometer . get\_gestures () qui renvoie un ensemble appelé tuple contenant l'historique des gestes. Le dernier élément du tuple est le geste le plus récent. Le tuple est réinitialisé à chaque appel de cette fonction.
- accelerometer.is\_gesture(nom\_du\_geste) qui renvoie une valeur booléenne indiquant si le geste en cours est nom\_du\_geste
- accelerometer.was\_pressed(nom\_du\_geste) qui renvoie une valeur booléenne indiquant si le geste nom\_du\_geste a été pressé depuis le dernier appel de cette fonction.

#### Exemple

Le programme ci-dessous affiche le nombre "8".

Lorsque la carte Micro: bit est secoué, il s'affiche une seconde plus tard de manière aléatoire et équiprobable soit "Oui", soit "Non".

Après une petite pause, le jeu recommence sauf si on appui sur le bouton B ce qui fait sortir de la boucle.

from microbit import \*
from random import choice

– .was\_pressed() renvoie une valeur booléenne qui indique si le bouton a été pressé depuis le dernier appel de cette méthode.



carte.

🛂 ГАИСАСЕ

Ecris un code permettant d'afficher le nombre de fois que le bouton B a été pressé dans les 10 secondes qui ont suivies l'alimentation de la

(bon à savoir) : la fonction str(nb) permet de transformer le nombre en entier nb en une chaîne de caractère qu'on peut  $\mathrm{Microsbit}$ .

## CORRECTION []: from microbit import \* [i]: display.scroll(str() button\_a.get\_presses() ))

Une carte  $\mathtt{Micro:bit}$  est un objet connecté qui, entres autres, est à l'écoute de certains **évènements**. Pour programmer cette capacité d'écoute, il est possible d'appliquer le principe fondamental des objets connectés : créer une **boucle infinie**  $\mathtt{while}$   $\mathtt{True}$ :

A chaque tour de cette boucle, le programme doit vérifier la réalisation de l'évènement attendu.

Lorsque l'événement se réalise, il est possible d'utiliser le mot clé break qui per-met alors de quitter la boucle infinie.

## Exemple

Le programme ci-dessous tourne en boucle:

- Lorsque aucun évènement n'est détecté, c'est l'image triste

## 4.2.7 — Le mouvement

L'accéléromètre de la carte Micro:bit est accessible par le module accelerometer.

Il est alors possible de récupérer une des coordonnées du vecteur accélération. Par exemple avec un appel à la fonction .  $\operatorname{get}_{-x}()$ .

## Exemple

Le programme ci-dessous affiche une flèche en fonction de l'inclinaison

**de la carte** Micro:bit.

Le bouton B permet de quitter la boucle infinie.

```
from microbit import *
button_b.was_pressed()
while True:
    capteur = accelerometer.get_x()
    if capteur > 40:
        display.show(Image.ARROW_E)
    elif capteur < -40:
        display.show(Image.ARROW_W)
else:
        display.show(Image.ARROW_W)

the button_b.was_pressed():
        display.clear():
        display.clear():
        break</pre>
```



- Image . SAD qui s'affiche.
- Pendant que le bouton A est pressé, l'image joyeuse Image . HAPPY s'affiche.
- Pendant que la broche pin1 est touchée (pour cela, pincer en même temps que la broche GND avec la main droite et la broche pin1 avec la main gauche), l'image endormie Image. ASLEEP s'affiche.
- Enfin, lorsque le bouton B est pressé, l'écran s'efface et le programme quitte la boucle.

```
from microbit import *
while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    elif pin1.is_touched():
        display.show(Image.ASLEEP)
    elif button_b.is_pressed():
        display.clear()
        break
    else:
        display.show(Image.SAD)
```

## **REMARQUE**

Les broches pin0, pin1 et pin2 peuvent aussi servir de bouton. Ainsi la méthode .is\_touched()} permet de renvoyer une valeur booléenne qui vaut True lorsqu'une personne en contact avec la masse (broche GND) touche puis relâche la broche en question.

En effet quand l'instruction est exécutée, la carte mesure la résistance entre la broche à laquelle l'instruction s'applique et la masse (broche



GND). Si cette dernière a varié et est passée d'une valeur quasi infinie à une valeur faible, le test devient vrai (True). Cet évènement arrive lorsqu'une personne en contact avec GND touche la broche puis la relâche.

#### 4.2.6 Le hasard

La bibliothèque random est utilisable avec Micro:bit. Grâce à cette bibliothèque, il est très simple de générer des nombres aléatoires avec les fonctions:

- random() pour tirer un nombre décimal (float) compris entre 0 (inclus) et 1 (exclu)
- randint(a,b) pour tirer un nombre entier (int) appartenant à a..b (inclus).

#### Exemple

Le programme ci-dessous affiche très rapidement 50 nombres aléatoires tirés entre 1 et 6. Le dernier nombre tiré est affiché pendant une seconde puis effacé.

```
from microbit import *
from random import randint
for i in range(50):
    display.show(random.randint(1, 6))
    sleep(20)
sleep(1000)
display.clear()
```