ACT2 Internet Of Thing : Exemple d'un objet connecté

1. Prise en main

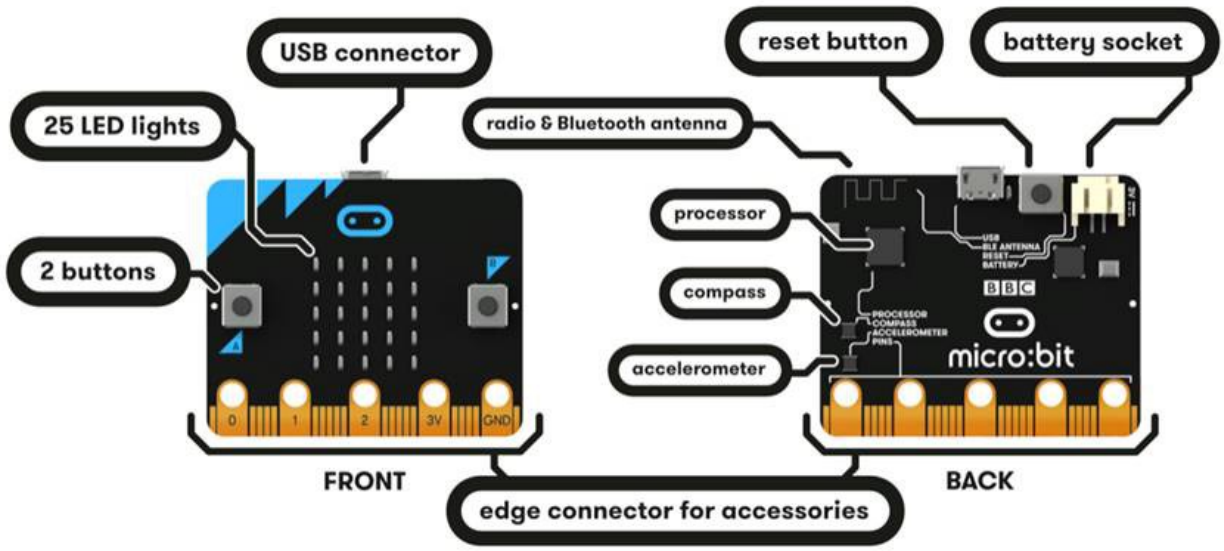
Présentation de la carte utilisée : Micro:Bit

La carte micro:bit est un ordinateur à processeur ARM conçu par la BBC pour l'éducation informatique au Royaume-Uni. Cette carte peut se programmer en utilisant plusieurs langages. Nous nous intéresserons ici uniquement à la programmation de la carte sous MicroPython.
Ce petit ordinateur possède la dernière technologie qui équipe les appareils modernes : téléphones mobiles, réfrigérateurs, montres intelligentes, alarmes antivol, robots, etc...

Ainsi, il s'apparente à ce que l'on nomme l'Internet des objets : *Internet of Things*, abrégé IoT.

Un micro:bit est à la fois autonome et extensible. En plus d'utiliser ses LED intégrées, boutons et capteurs, nous pouvons élargir sa gamme de fonctions en l'insérant dans un connecteur.

On la programme avec MuPython (https://codewith.mu/) ou en ligne sur Vittascience (https://fr.vittascience.com/microbit/).



Programmation

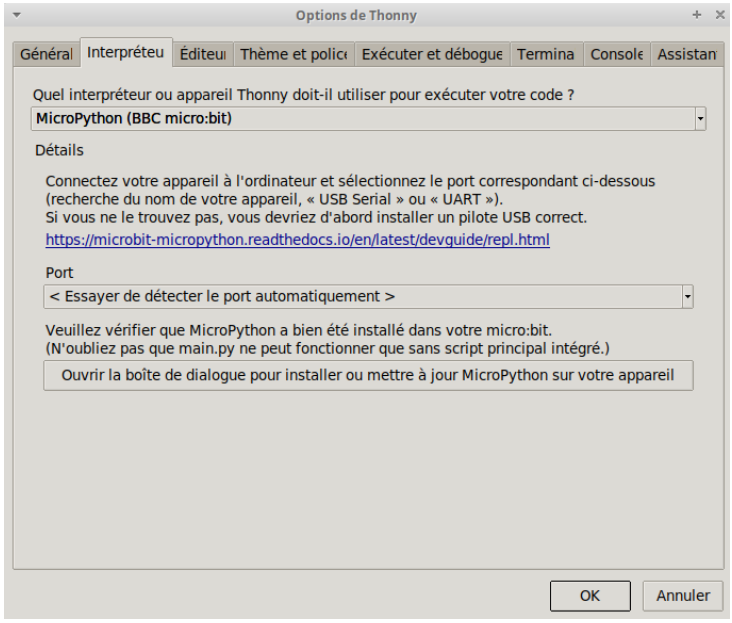
La documentation est disponible ici : https://microbit-micropython.readthedocs.io/fr/latest/

- Ouvrir l'éditeur, et copier-coller le programme ci-dessous :

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

- Connectez la carte à un port USB de votre PC.
- Transférer le code sur la carte :

Sur Thonny → *Options > changer l'interpréteur python*



Sur Vittasciences https://fr.vittascience.com/microbit/

→ *Uploader le programme exécutable (microbit.hex), à transférer dans la carte micro:bit*

Une DEL doit clignoter au dos de celle-ci lors du transfert. Lorsque celui-ci est terminé, le programme est utilisable.

2. Applications

Exercice 1 : Affichages

- Afficher un message lettre par lettre avec la commande `display.show("Voici un test")` .
- Tester la différence avec `display.scroll("Ceci est du scrolling")` .
  a) Dans la doc, voir les différentes méthods liées à `display`.
  b) Sélectionner la méthode permettant d'effacer la lettre à l'écran (cela éteint l'ensemble des LEDs).
- Les 25 LEDs de la Micro :Bit ont chacune des coordonnées dans un repère.
  a) Voir le détail de la commande `display.set_pixel` avec la fonction : `help(display.set_pixel)` .
  b) Colorer les LEDs situées aux quatre extrémités en testant différentes intensités lumineuses pour chaque coin.
  c) Effacer puis créer une double boucle qui colorie les lignes 1 et 2 en faisant progressivement passer l'intensité lumineuse de 0 à 9.

Exercice 2 : Température

Ici on cherche à utiliser le capteur de température de la carte. Ce capteur mesure la température du microcontroleur principal. Ce type de composant chauffe assez peu et a donc tendance à refléter la température extérieure.

- Créer une boucle infinie qui affiche la température toutes les 3 secondes.

La commande qui récupère la température sous forme de nombre entier de degrés Celsius est `temperature()` . Attention, c'est un nombre, donc pour l'afficher il faut le convertir en string avec `str(nombre)` .

- Recouvrir doucement de son doigt le microcontrôleur central afin de réchauffer le processeur et vérifier que cela a bien un impact après quelques secondes.

- Créer un programme qui affiche un icône de satisfaction (HAPPY) ou de tristesse (SAD) selon que la température dépasse 20°C ou non.

Exercice 3 : Compas

- Afficher la valeur du compas lorsqu'on appuie sur le bouton A à l'aide du programme suivant :

```
from microbit import *
compass.calibrate()
while True:
  if button_a.is_pressed() :
      display.show(str(compass.heading()))
  display.clear()
```

- Que renvoie la valeur ? à quoi correspond le 0 ?
- Créer une boussole : faire en sorte qu'au lieu d'un nombre, l'image affichée soit un trait qui indique le nord.

On pourra utiliser l'image de l'aiguille de l'horloge : `display.show(Image.ALL_CLOCKS[heure])`
(où heure est un nombre entre 0 et 11)

3. Chi-Fu-Mi

Objectif :

MVP (ou Minimum Valuable Product)

Transformer sa carte Micro :Bit en plateforme pour Chi-Fu-mi, afin d'affronter un adversaire.
Les règles sont les suivantes :

- Chaque joueur dispose d'une carte.
- Chacun secoue sa carte et celle-ci affiche un nombre, une lettre ou un symbole correspondant à Pierre, Feuille ou Ciseaux.
- Chaque partie est réinitialisée par le fait de presser A et B simultanément.

En plus

Introduire une 3ème carte qui va recevoir et traiter les résultats de chaque joueur.
Puis, elle renvoie à chacun le résultat.

On utilisera les ressources : https://microbit-micropython.readthedocs.io/fr/latest/

Pour la mesure d'une secousse, on utilise l'accéléromètre, qui est capable de classer certains gestes, notamment `accelerometer.is_gesture('shake')`

EVALUATION :

APP	• Rechercher l'information utile à l'aide de sources fiables
REA	• Mettre en œuvre une solution, par la traduction d'un algorithme ou d'une structure de données dans un langage de programmation. • Imaginer et concevoir une solution, décomposer en blocs, se ramener à des sous-problèmes simples et indépendants, adopter une stratégie appropriée

Fonctionnalités demandées :

• *Réinitialisation par press A et B*	/2	• *Affichage d'un motif aléatoire à la secousse*	**/3**
• *Connexion des deux cartes entres elles*	/3	*Documentation du projet sur les sources utilisées*	**/2**

Fonctionnalités plus : (+1pt par fct)

• *3ème carte qui comptabilise...*		• *… Arbitre...*	
• *… Chronomètre ...*		• Utilisation des classes	

Code :tre

• *Lisibilité du code*	**/2**
• *Variables explicites*	**/2**
• *Pas de répétitions, utilisation de boucles et de fonctions*	**/2**
• *Commentaires pertinents*	**/2**