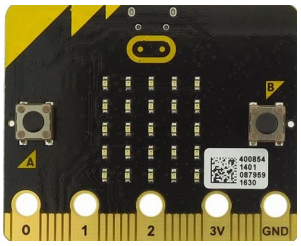
Chap.1 Un système embarqué : la carte micro:bit

1. Introduction



Vous avez normalement été initiés aux systèmes embarqués en seconde.
Nous étudierons particulièrement cette année la carte micro:Bit équipée d'un microcontrôleur ARM Cortex-M0 32 bits conçue spécialement pour l'éducation informatique au Royaume-Uni.

On peut programmer cette carte micro:Bit en javascript, scratch3, LISP, C++, mais aussi en MicroPython un langage très similaire au Python. Pour cette année nous nous limiterons uniquement à la programmation de la carte sous MicroPython.

Nous utiliserons l'éditeur MicroPython **mu** , éditeur libre, gratuit, multi-plateforme, téléchargeable ici.

Mu permet de transférer très facilement son programme sur la carte avec l'icône « **flasher** »
Il permet aussi (voir prochain chapitre) de contrôler directement sa carte avec les icônes « **REPL** » et « **Graphique** ».

Un Tuto Micro:bit en français ici ou là. Les entrées-sorties de la carte ici.
Une très bonne vidéo d'introduction aux systèmes embarqués sous MicroPython ici .

1. Présentation rapide des différents capteurs intégrés et entrées/sorties

La carte BBC micro:Bit est munie :

- d'un accéléromètre 3D
- d'un capteur de température
- d'une boussole numérique 3D associée à un magnétomètre
- d'une matrice de LED 5x5 , LED qui peuvent s'allumer ou servir de détecteur de lumière.
- de 2 boutons programmables.
- d'une connexion Bluetooth
- d'une connexion radio
- d'une connexion USB
- de 20 entrées/sorties analogiques, digitales, Bus etc

Ces différents capteurs, entrées, sorties permettent de nombreuses applications. Nous en verrons quelques-unes dans ce cours.

2. Afficher des images et du texte sur la matrice de LED

La matrice de LED permet d'afficher ou de faire défiler du texte, mais aussi des images de la bibliothèque MicroPython.

Exercice 1. Teste sur MU le code suivant et détermine le rôle des méthodes display.show(), display.clear(), de la fonction sleep(). Identifie sur la micro:Bit les boutons a et b

```
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
        sleep(1000)
        display.clear()
    elif button_b.is_pressed():
        display.show(Image.SAD)
        sleep(1000)
        display.clear()
    else:
        display.show("?")
```

Definition 1.

- **display.scroll('texte')** permet d'afficher un texte lettre par lettre
- **`running_time()`** renvoie le nombre de millisecondes depuis que l'appareil a démarré.
- ***button_a.is_pressed*()** renvoie *True* si le bouton a est effectivement appuyé
- ***button_a.was_pressed*()** renvoie *True* si le bouton a a été appuyé
- ***button_a.get_presses*()** renvoie le nombre de fois que le bouton « a » a été appuyé.

Exercice 2. Modifie le code précédent en changeant les images et en affichant sur la matrice de LED le message (ALORS?) à la place de « ? » si personne n'appuie sur les boutons pendant plus de 5s.

Chaque pixel (LED) de la matrice peut prendre des valeurs d'intensité entre 0 et 9. La méthode **display.set_pixel(x,y,i)**, où (x,y) sont les coordonnées du pixel et i l'intensité permet de les allumer. On peut aussi les contrôler tous ensemble en créant sa propre image avec la fonction image().

Exemple : fleche=Image("00900:00090:99999:00090:00900")

Exercice 3. Faire un code qui affiche une croix et le signe + si on appuie sur un des 2 boutons.

Exercice 4. Faire un code qui affiche le nombre de fois où on a appuyé sur le bouton a.

3. Le capteur de température

La carte micro:Bit intègre un capteur de température. Soyons honnêtes, il n'est pas « TOP » dans la mesure ou il est soudé sur la carte elle-même, à proximité du processeur qui chauffe… Mais bon, il a le mérite d'exister.

Exercice 5. Teste le code suivant. Modifie le en faisant clignoter l'affichage si la température dépasse 30 degrés.

```
from microbit import *
while True:
  geste=accelerometer.current_gesture()
  if geste=="shake":
    display.clear()
    display.show(temperature())
```

4. Le détecteur de luminosité

Peur du noir ? Pas de problème, la matrice de LED devient un capteur de lumière basique très facilement, te permettant de détecter la luminosité ambiante et d'actionner d'autres capteurs si besoin.

Exercice 6. Teste le code suivant et modifie le pour que la matrice de LED clignote une seconde si la luminosité est faible

```
from microbit import *
while True:
 #detecte le niveau de lumière  reçue par la matrice de LED
 lightLevel = display.read_light_level() #varie entre 0 et 255
 if lightLevel<50 :
   display.clear()
   display.show(Image.ANGRY)
 else :
   display.clear()
   display.show(Image.HAPPY)
 sleep(500)
```

5. Pour ne pas perdre le Nord !

Tu ne trouves plus la salle de cours NSI ! No problemo, la micro:Bit va t'aider à retrouver ton chemin grâce à sa boussole intégrée.

Definition 2.

- `compass.calibrate()` permet de calibrer la boussole sous forme d'un petit jeu où on doit éclairer toute la matrice de LED.
- `compass.heading()` Donne l'angle entre le Nord et le compas (entre 0 et 359°)

Exercice 7. Teste le code suivant et essaie de comprendre la petite astuce mathématique permettant de transformer l'angle avec le nord en une aiguille d'horloge variant de 0h à 12 h. Modifie le pour qu'il alterne toutes les 5 s entre l'affichage de l'aiguille et un affichage des lettres N, S, E, W indiquant la direction générale.

```
from microbit import *
#calibre la boussole
if not(compass.is_calibrated()) :
   compass.calibrate()
#indique le nord
while True:
  aiguille = ((15 - compass.heading()) // 30) % 12
  display.show(Image.ALL_CLOCKS[aiguille])
```

Le magnétomètre offre d'autres possibilités en plus de la boussole que nous ne développerons pas ici.

Definition 3. `compass.get_field_strength()` : méthode qui renvoie l'intensité du champ magnétique. Cela permet de détecter la présence d'un aimant.

Cette force peut être décomposée selon les axes x, y et z : et compass.get_x() compass.get_y() compass.get_z() donnent les valeurs de l'intensité suivant ces directions.

6. Communiquer par radio

Super ! Avec 2 cartes micro:Bit vous pouvez communiquer par radio, vous envoyer des messages, des données… Tout est là.

Exercice 8. Teste le code suivant et écris le code de la carte réceptrice pour envoyer un message de réponse (« Oui ça roule, et toi ?») en utilisant la commande ***radio.receive()***

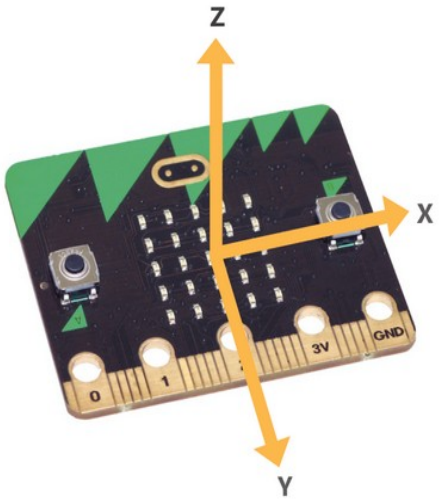
```
from microbit import *

import radio

radio.on() #allume la radio
radio.config(group=1) #sélectionne le canal de communication
while True :
  if button_a.is_pressed():
       radio.send("Salut ça roule ?")
```

La carte micro:Bit via sa connexion Bluetooth lui permet aussi de se connecter à différents appareils (portable, PC etc.). Nous ne développerons pas cette partie dans ce cours.

7. L'accéléromètre

L'accéléromètre de la carte est du même type que celui de votre téléphone portable, celui qui vous permet par exemple d'afficher l'écran dans le bon sens.

Il permet de détecter les mouvements de la carte et son orientation dans l'espace.



Definition 4. **`accelerometer.current_gesture()`** renvoie une variable de type str associée aux gestes suivants :

"up", "down", "left", "right", "face up", "face down", "freefall", "3g", "6g", "8g", "shake".

Exercice 9. Teste le code suivant et modifie le pour afficher un visage triste lorsqu'on secoue la carte.

```
from microbit import *

while True:
    gesture = accelerometer.current_gesture()
    if gesture == "face up":
        display.show(Image.HAPPY)
    else:
        display.show(Image.ANGRY)
```

A. Carte micro:bit au repos

Definition 5. **Au repos : L'accéléromètre** mesure suivant son inclinaison les coordonnées du vecteur correspondant à l'accélération de la pesanteur ($g=9,8ms^{-2}$) en millième de g.

`microbit.accelerometer.get_values()` mesure ses coordonnées (x,y,z) et les renvoie dans un tuple

Exemple : si la carte est horizontale, le vecteur accélération de la pesanteur étant vertical et orienté vers le bas, on aurait (0,0,-1000)

Exercice 10.

En cadeau, un petit niveau à bulles pour vérifier que votre table de travail est bien horizontale.

Testez ce code et comprenez-le.

```
from microbit import *
x,y=2,2
while True:
  lectx,lecty=accelerometer.get_x(),accelerometer.get_y()
  if lectx>20:
    ix=2
  elif lectx<-20:
    ix=-2
  else :
    ix=0
  if lecty>20:
    iy=2
  elif lecty<-20:
    iy=-2
  else :
    iy=0
  display.clear()
  display.set_pixel(x+ix,y+iy,9)
  sleep(100)
```

Exercice 11.

Analyse le code suivant et anticipe ce que son exécution produira sur la carte micro:Bit. Flashe le script sur la carte et vérifie ta prédiction :

```
from microbit import *
sens = 100 #sensibilité du réveil du dormeur
display.show(Image.ASLEEP)
reposX, reposY, reposZ = accelerometer.get_values()
sleep(500)
while True:
  X, Y, Z = accelerometer.get_values()
  display.clear()
    if not(reposX-sens<X<reposX+sens and reposY-sens<Y<reposY+sens and
reposZ-sens<Z<reposZ+sens):
    display.show(Image.ANGRY)
  else:
    display.show(Image.ASLEEP)
  sleep(100)
```

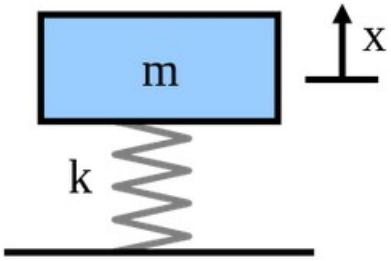
Mini Projet1. En vous inspirant des exercices 10 et 11 :

- écris un programme permettant de piloter une led dans la matrice de la carte micro:Bit avec l'accéléromètre.
- fais évoluer le code pour que la led laisse une traînée évanescente derrière elle.
- laisse libre cours à ton imagination pour développer ce programme...

B. Carte micro:Bit en mouvement

ATTENTION lorsque la micro:Bit est en mouvement le problème est plus compliqué et souvent on trouve tout et n'importe quoi sur les docs en ligne. Pour comprendre ce que mesure un accéléromètre faisons un peu de Physique.

Physique : Ben ? On mesure quoi alors ?



Imaginons le système suivant se déplaçant en accélérant vers le haut. Plus l'accélération est grande, plus la masse sera écrasée sur le ressort (comme en avion lorsqu'on est collé sur son siège au décollage). Du point de vue de la masse (dans son référentiel) les forces qui l'écrasent sur le ressort sont :

Son poids : $\vec{P}=m.\vec{g}$ La force d'inertie : $\vec{F}=-m.\vec{a}$

Et bien au coefficient m près c'est cela que mesure l'accéléromètre et dans les 3 directions x,y,z. Il mesure $\vec{A}=\vec{g}-\vec{a}$

Ce que l'on peut considérer comme une **«pesanteur apparente »** subie par la masse (en multipliant par **m** on

obtient $m(\vec{g}-\vec{a})$ qui correspond **au poids apparent** de la masse). **DONC ATTENTION, contrairement à ce qu'on peut lire ici où là** ce n'est pas son accélération par rapport à la Terre.

- **Exemple 1** : une personne en chute libre a une accélération égale à g vers le bas, donc une action ressentie **A=0** ; elle est en impesanteur. **Elle a l'impression de ne rien peser.**
- **Exemple 2** : Un cosmonaute dans une fusée qui accélère à 2 g vers le haut va être écrasé sur son siège par une pesanteur apparente dirigée vers le bas de **3g. Il a l'impression de peser 3 fois son poids.**

Definition 6. **En mouvement :** L'accéléromètre mesure les coordonnées (x,y,z) de l'intensité de la **pesanteur apparente** subie par la micro:Bit. On peut calculer cette grandeur (qui **n'est pas sa réelle accélération par rapport au sol)** par la formule :

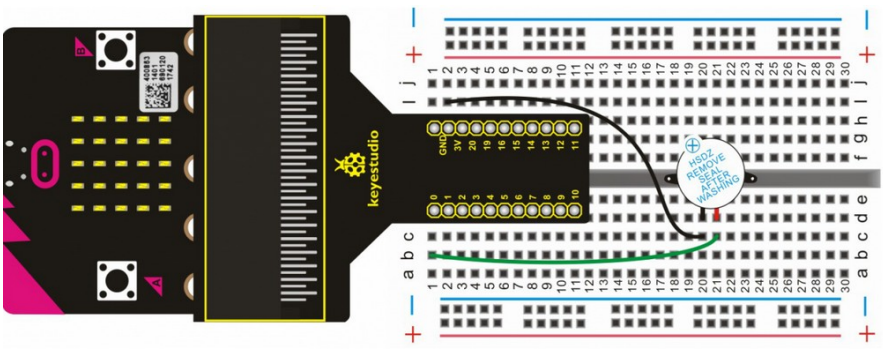
$$A=\sqrt{x^2+y^2+z^2}$$

8. Les entrées/Sorties

La carte dispose de 20 entrées/sorties digitales (prenant uniquement comme valeur 0 ou1) ou analogiques (valeurs continues entre 0V et 3V) qui vous permettent de faire des tas de montages intéressants.
Toutes les entrées sorties ainsi que les commandes qui leur sont associées se trouvent ici.

Exercice 12. Faire sonner un buzzer toutes les demi-secondes. Réalise le montage ci-dessous et flashe le code sur la carte. Si tu as le système de branchements Grove, adapte ce montage.

```
from microbit import *
while True:
  pin0.write_digital(1)
  sleep(20)
  pin0.write_digital(0)
  sleep(480)
```

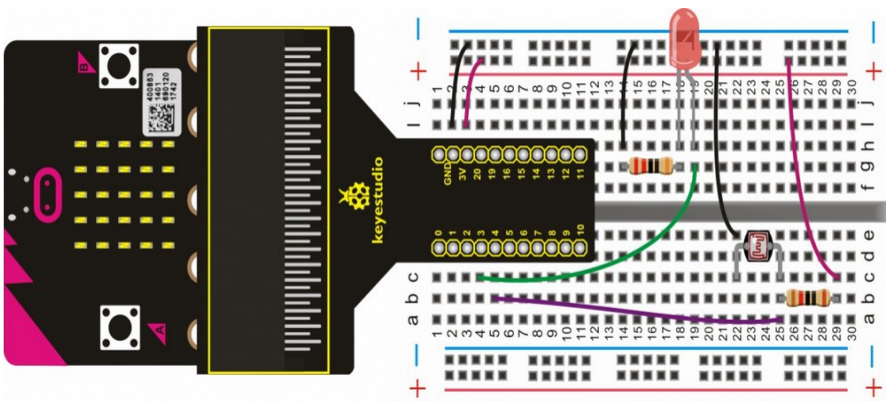


- Que fait la ligne de code `pin0.write_digital(1)` ?
  Que fait la ligne de code `pin0.write_digital(0)` ?
  Modifie les durées (20 ms et 480 ms) pour tester tes réponses

Suivant le matériel que tu as sur ta table fais l'exercice 12 ou l'exercice 13

Exercice 13. Détecteur de lumière avec une photorésistance. Réalise le montage ci-dessous et flashe le code sur la carte. Couve la photorésistance avec ta main puis éclaire là. Observe.

```
from microbit import *
display.off()
while True:
  lum=pin4.read_analog()
  pin3.write_analog(lum)
  print(lumiere)
  sleep(100)
```

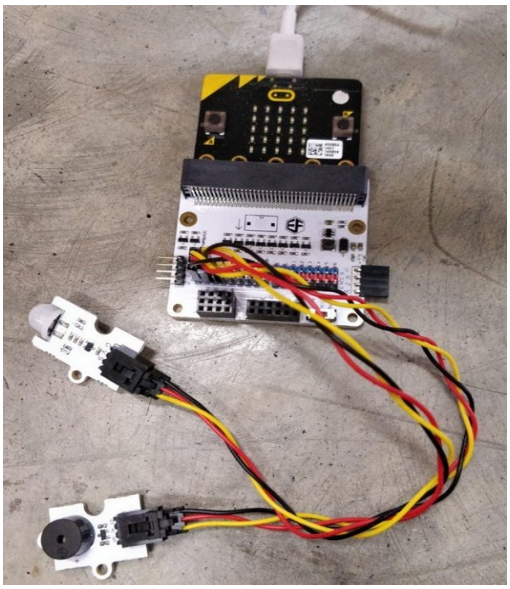


Que fait la ligne de code `pin4.read_analog()` ?
Que fait la la ligne de code `pin3.write_analog(lum)` ?

ATTENTION : Les pin 4 et 3 contrôlent aussi 2 colonnes de la matrice LED. Si on veut s'en servir en entrées/sorties on doit désactiver la matrice avec l'instruction `display.off()`

Exercice 14. Tu veux protéger ton bureau avec une alarme hyper moderne ? No problemo, la micro:Bit et son capteur détecteur de mouvement sont là pour toi. Branche le buzzer sur le pin 0 du Grove et le détecteur sur le pin 1 et flashe ce code.

```
from microbit import *
import music
buzzer_pin= pin0
PIRSensor_pin = pin1
while True:
  if PIRSensor_pin.read_digital() == 0:
    music.stop()
  else:
    music.play("c5:4", wait=False)
```



En t'aidant du module music, modifie ce code pour simuler en cas d'intrusion la sirène de la police (les notes sont RE LA et durent une seconde)