SNT – ELLA

OBJETS CONNECTÉS

TP NOTÉ
UTILISATION DES CAPTEURS SUR CARTE MICRO:BIT

Important :

- Lire **attentivement** les consignes car aucune aide ne pourra être apportée par le professeur.
- Les 6 premiers dépôts sont notés.
- Ce travail est long mais pas difficile. **En cas de blocage, changer de partie.**
- Avant de démarrer, créer un dossier nommé **tachefinaleMB**
- Tous les programmes créés aujourd'hui seront sauvés dans ce dossier.
- Ouvrir le logiciel *Mu* et connecter votre carte Micro:bit sur un port USB.

PARTIE 1 : UTILISATION DES BOUTONS A ET B

RAPPELS :

Il y a **deux boutons A et B** sur la face avant de la carte Micro:bit.
On peut détecter quand ces boutons sont pressés, ce qui permet de déclencher des instructions sur l'appareil.

- `button_a.is_pressed()` : renvoie **True** si le bouton A est actuellement enfoncé et **False** sinon
- `button_a.was_pressed()` : renvoie **True ou False** pour indiquer si le bouton A a été appuyé depuis le démarrage de l'appareil ou la dernière fois que cette méthode a été appelée
- `button_a.get_presses()` : renvoie le nombre de fois où on a appuyé sur le bouton A

EXERCICE : DEUX TEXTES DIFFÉRENTS

1. Compléter ce programme qu'il affiche BONJOUR ou HELLO suivant que le bouton A ou B est actuellement pressé

```
1 from microbit import *
2
3 while True:     #répéter indéfiniment ce qui suit
4   if button_a.is_pressed():
5     display.scroll("BONJOUR")
6     sleep(500)
7   if ..............................  :
8     ...
9     ...
```

2. Flasher le programme sur la carte et vérifier qu'il fonctionne en pressant le bouton A puis quelques secondes plus tard, le bouton B.
3. Sauver ce programme sur le nom **deuxtextes-mb.py**

DEPÔT 1 — **deuxtextes-mb.py sur http://entraide-ella.fr**

PARTIE 2 : CAPTEUR DE LUMIÈRE

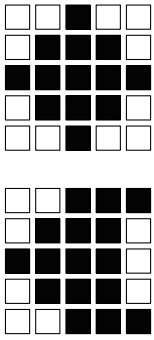
En inversant les LEDs d'un écran pour devenir un point d'entrée, l'**écran LED** devient un capteur de lumière basique, permettant de détecter la luminosité ambiante.

La commande `display.read_light_level()` retourne un entier compris entre 0 et 255 représentant le niveau de lumière.

E EXERCICE : LE SOLEIL A RENDEZ-VOUS AVEC LA LUNE

1. Créer un **nouveau programme** et saisir puis compléter le programme suivant pour afficher une image de lune si on baisse la luminosité (en recouvrant la carte avec sa main par exemple) et un soleil sinon.
2. Sauvegarder ce programme sous le nom **lunesoleil-mb.py**.

```
from microbit import *

soleil = Image("00900:"
               "09990:"
               "99999:"
               "09990:"
               "00900")

lune = Image(...    #code à compléter à l'aide du modèle de lune  ci-contre



while True:
    if display.read_light_level()<...  :   # tâtonner avec un entier entre 0 et 255
        display.show(lune)
    else:
        display.show(...)
    sleep(10)
```



DEPÔT 2 — **lunesoleil-mb.py sur http://entraide-ella.fr**

PARTIE 3 : CAPTEUR DE TEMPÉRATURE

La carte Micro:bit n'a pas un capteur de température dédié. Au lieu de cela, la température fournie est en fait la température de la **puce de silicium du processeur principal**. Comme le processeur chauffe peu en fonctionnement (c'est un processeur ARM à grande efficacité), sa température est une bonne **approximation** de la température ambiante.

L'instruction `temperature()` renvoie la température de la carte Micro:bit en degrés Celsius.

E EXERCICE : THERMOMÈTRE DE SECOURS

Écrire un **nouveau programme** nommé **temperature-mb.py** qui affiche la température en défilement.

```
from microbit import *

while True:
    temp  = ..............    #On recupère la température de la carte
    chaine =  ............    #On convertit l'entier temp en chaine de caractères
    display.scroll(........)  #On affiche la variable temp
    sleep(100)      #On marque une pause
```

DEPÔT 3 — **temperature-mb.py sur http://entraide-ella.fr**

Un **accéléromètre** mesure l'accélération de la carte Micro:bit, ce composant détecte quand la carte est en mouvement. Il peut aussi détecter d'autres actions (gestes), par exemple quand elle est secouée, inclinée ou qu'elle tombe.

Des accéléromètres sont présents dans les smartphones et les manettes de jeux afin de pouvoir afficher l'image dans le bon sens, ou bien de changer de direction dans un jeu.

L'accéléromètre mesure le mouvement selon trois axes :

- X - l'inclinaison de gauche à droite
- Y - l'inclinaison d'avant en arrière
- Z - le mouvement haut et bas

L'instruction `accelerometer.get_x()` permet de détecter un mouvement de gauche à droite en renvoyant un nombre compris entre -1023 et 1023 (0 étant la position "d'équilibre").
L'instruction `accelerometer.is_gesture("shake")` renvoie **True** ou **False** selon si la carte est secouée.

E EXERCICE : EN HAUT, EN BAS, À GAUCHE, À DROITE

1. Créer le **nouveau programme** ci-dessous.

```
from microbit import *

while True:
    x = accelerometer.get_x()
    if x > 500:
        display.show(Image.ARROW_E)
    elif x < -500:
        display.show(Image.ARROW_W)
    else:
        display.show("-")
```

2. Flasher ce programme appelé **inclinaison-mb.py** sur la carte. Expliquer ce qu'il fait.
3. Compléter le programme avec une variable **y** pour qu'il affiche une flèche vers le haut si on incline la carte en avant et une flèche vers le bas si on l'incline en arrière.

   (On pourra utiliser **Image.ARROW_N** et **Image.ARROW_S**,)

DEPÔT 4 **inclinaison-mb.py sur http://entraide-ella.fr**

E EXERCICE : LANCER DE DÉS

1. Compléter le programme suivant afin qu'il simule un dé en affichant une face au hasard lorsque la carte Micro:bit est secouée.

```
from microbit import *
from random import randint

while True:
    if accelerometer.is_gesture("shake"):   # On a secoué la carte
        de = randint(1,6)      # On choisit un nombre au hasard entre 1 et 6
        display.show(...)      # On affiche le nombre précédent
        ....... .              # On marque une pause de 500 ms
```

2. Sauvegarder ce programme sous le nom **hasard-mb.py**

DEPÔT 5 **hasard-mb.py sur http://entraide-ella.fr**

FONCTIONNEMENT

L'interaction sans fil n'est que de la physique : les **ondes radio** (une sorte de radiation électromagnétique, un peu comme la lumière visible) ont certaines propriétés (comme l'amplitude, la pulsation ou la période) modulées par un émetteur de façon à ce que cette information puisse être encodée et ainsi diffusée.
Lorsque des ondes radio rencontrent un **conducteur électrique** (c'est-à-dire une antenne), elles provoquent l'apparition d'un **courant alternatif** duquel l'information contenue dans les ondes peut être extraite et retraduite dans sa forme originale.

A RETENIR

Il y a un **émetteur/récepteur radio** sur la carte Micro:Bit. Il permet à deux cartes de communiquer à distance.
Plusieurs commandes sont disponibles :

- `radio.on()` : allume la radio. À écrire **obligatoirement avant** toute utilisation de la radio
- `radio.off()` : éteint la radio
- `radio.send(message)` : envoie la chaîne de caractères (*texte*) "message"
- `radio.receive()` : reçoit le prochain message à être diffusé sur le canal de réception
- `radio.config(channel=7)` : configure le canal radio sur lequel vous allez émettre et recevoir (comme les *Talkie-Walkies* !). Les canaux disponibles vont de 0 à 83 (celui par défaut étant le numéro 7)

Avant de commencer, ouvrir le logiciel *Mu* et connecter votre carte Micro:bit sur un port USB.

EXERCICE : ENVOI/RÉCEPTION D'UN MESSAGE-IMAGE

```
import radio
from microbit import *



radio.on()
radio.config(channel = ...)     #choisis pour canal un entier entre 0 et 83 (le même que ton voisin)

while True:
    #émission d'un message en cas d'appui sur le bouton A
    if button_a.was_pressed():
        message = Image.PACMAN   #ou toute image parmi :  Image.HEART , Image.HAPPY, Image.SMILE...
        radio.send(message)

    #réception d'un message
    messagerecu = radio.receive()
    if messagerecu != None:     #Tu as a reçu un message de ton voisin
        display.show(messagerecu, delay=100, wait=False)
        sleep(1000)
        display.clear()
```

1. Flashe ce programme sur la carte.
2. Demande à ton voisin de ne pas toucher à sa carte et de ton côté presse la bouton A.
3. Dès que ton voisin a reçu l'image, demande-lui à son tour de presser le bouton A.

E EXERCICE : ENVOI/RECEPTION D'UN TEXTO

```
import radio
from microbit import *

radio.on()
radio.config(channel=...)     # de  0 à 83

while True:
    if button_a.was_pressed():
        radio.send("...................")

    signalrecu = radio.receive()
    if signalrecu != None:
        display.scroll(signalrecu)
```

Indications :

- « ! = » correspond au symbole mathématique « $\neq$ »
- le qualificatif `None` (aucun) signifie que la variable **signalrecu** ne possède encore aucune valeur

1. Former un binôme avec un autre élève et choisir un canal radio commun autre que 50 (compris entre 0 et 83).
2. Compléter la commande `radio.send` pour envoyer son prénom au binôme.
3. Flasher le programme sur sa carte et tester tour à tour l'envoi de message radio.
4. Sauvegarde ce programme sous le nom **message-mb.py**

DEPÔT 6 **message-mb.py sur http://entraide-ella.fr**

E EXERCICE : DOMOTIQUE EN MINIATURE (PARTIE OPTIONNELLE)

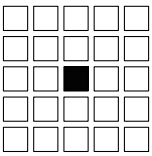
L'objectif est de simuler un système de domotique en miniature : une carte va simuler les capteurs de la maison et l'autre carte va simuler l'interface de l'utilisateur.

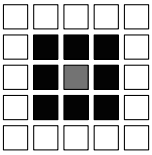
```
import radio
from microbit import *

radio.on()
radio.config(channel=50)      # canal à modifier avec son binôme

while True:
    if button_a.is_pressed():
        radio.send("A")
    if button_b.is_pressed():
        radio.send("B")

    signalrecu = radio.receive()
    if signalrecu != None:
        display.show(signalrecu)
        sleep(500)
        display.clear()
        if signalrecu == "A":
            radio.send("a")
        elif signalrecu == "B":  #sinon si
            radio.send("b")
```

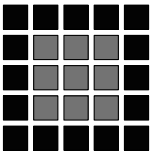
1. Se placer en binôme et choisir un canal.
2. Modifier ce programme afin qu'il affiche la température de la maison (relevée sur le capteur du binôme) lorsque l'utilisateur appuie sur le bouton A.
3. Modifier ce programme afin qu'il affiche la luminosité de la maison (relevée sur le capteur du binôme) lorsque l'utilisateur appuie sur le bouton B.
4. Sauvegarder ce programme sous le nom **domotique-mb.py**

MP PROJET 1 : CARRÉS EN BOUCLE

Création d' une animation avec un carré qui s'agrandit encore et encore.







```
1  from microbit import *
2
3  carre1 = Image(...)
4
5  carre2 = Image(...)
6
7  carre3 = Image(...)
8
9  while True:
10     ...
```

1. Compléter le programme microPython afin de répondre à l'objectif en s'inspirant des trois images ci-dessous pour les carrés.
2. [**bonus**] Modifier le programme pour que l'affichage se déclenche lorsque le bouton A est appuyé et s'arrête lorsque le bouton A est relâché.

DEPÔT 7 **projet1-mb.py sur http://entraide-ella.fr**

MP PROJET 2 : LA FLÈCHE TOURNANTE

Affichage d' une flèche tournante sur la matrice LED lorsque le bouton A est pressé et qui va s'arrêter sur sa dernière position lorsque le bouton B est pressé.

1. Écrire un programme microPython qui réponde à l'objectif affiché.
2. [**bonus**] Modifier ce programme pour que la flèche tourne de plus en plus vite !

DEPÔT 8 **projet2-mb.py sur http://entraide-ella.fr**

MP PROJET 3 : MOTUS

Créer un programme qui simule le tirage au sort d'une boule avec des numéros entre 1 et 9.
L'urne contient aussi deux boules noires, on affichera par exemple une tête de mort si jamais l'une d'entre elles est piochée.
Le tirage d'une boule se lancera en secouant la carte.

1. Écrire un programme microPython qui réponde à l'objectif affiché.

DEPÔT 9 **projet3-mb.py sur http://entraide-ella.fr**